Using HomeSite+ for Dreamweaver MX

HomeSite+ for Dreamweaver MX for Windows 98, Windows ME, Windows NT 4.0, Windows 2000, and Windows XP

Macromedia® Incorporated

Copyright Notice

Part number: ZCS50M100

Contents

About This Book xi

Intended audience xii

Developer resources xii
- Macromedia website xii
- ColdFusion web resources xiii
- Tutorials xiii

About the documentation xiv
- Documentation conventions xiv
- Viewing online documentation xiv
- Printing online documentation xv

Getting answers xvi

Contacting Macromedia xvi

Copyright information xvii

Chapter 1 Setting Up the Product 1

System requirements 2

Installing or upgrading 2

Completing the setup 3
- About the Initial Configuration Wizard 3
- Using the Initial Configuration Wizard 4

Monitoring system resources 6
- Releasing system resources taken at startup 6
- About the Resource Level Monitor 6
- Using the Resource Level Monitor 7

Chapter 2 Configuring Browsers and Servers 9

Configuring the internal browser 10

Configuring an external browser 12

Working with files on remote servers 14
- Using the SNE node 14
- Connecting to a server 15

Managing servers 19
About server mappings 20
Adding a server mapping 23

Chapter 3 Exploring the Workspace 25

The workspace areas. 26
Setting options 27

Working in the Resources window. 28

Creating and browsing files in the Document window. 29
About the Edit tab 29
About the Browse tab 29
About the Help tab 29

Tracking your work in the Results window 30

Customizing the workspace 31
Managing application toolbars and the QuickBar 31
Customizing toolbars 32

Getting the most from the online Help system. 35
Opening Help in tag editors and Tag Chooser 35
Editing Help in tag editors and Tag Chooser 35
Accessing online Help 35
Printing Help 35
Bookmarking Help 36
Searching the online Help 36
Extending the Help system 38
Adding media content 40

Chapter 4 Managing Files 41

About the Files tabs. 42

About file encoding. 43

Working with files 44
Setting startup options 44
Opening a file 44
Adding a link to an open file 45
Saving a file 45
Backing up files 46
Changing the file list display 48
Dragging a file from Windows Explorer 48
Building a Favorite Folders list 48
Downloading a web page 49

Chapter 5 Writing Code and Web Content 51

Inserting code 52
Inserting a tag from the QuickBar 52

Selecting a tag from Tag Chooser 53
Completing a tag with a tag editor 54

Using inline tools to enter code 55
Using Tag Insight 55
Using Function Insight 56
Using Tag Completion 56
Using Auto Completion 57
Using Code Templates 57
Using the Extended and Special Characters palette 58

Using code generating tools 58

Adding document content 59

Using keyboard shortcuts 61

Saving a code block as a snippet 62
Sharing snippets 63
Assigning a shortcut key to a snippet 63

Resources for website accessibility 64

Tips for visually impaired users 65
Assigning keyboard shortcuts 65
Working with user interface elements 66

Chapter 6 Editing Pages 67

Setting editor options 68
Using the Editor toolbar 68
Settings editor options 68

Selecting a code or text block 69

Saving text to the multiple-entry Clipboard 69
Using the Clipboard 69
Setting the Clipboard entry limit 69

Collapsing text 70

Editing a referenced file 71
Editing an image file 71
Editing an included file 72

Using tag editors 73
About VTML tag editors 73
Editing a tag with a tag editor 73

Navigating the structure of a document 74
About outline profiles 74
Setting the Tag Tree display 75

Editing code in the Tag Inspector 76
Setting the Tag Inspector display 76
Creating and editing an event handler script block 77

Formatting pages with Cascading Style Sheets 78
About Cascading Style Sheets (CSS) 78
About the integrated style editor 79

Chapter 7 Using Web Development Languages 81

About language support 82
Supported languages 82
How a language is detected 83

Setting options for markup languages 83

Coding in XHTML 85
What is XHTML? 86
Setting options for XHTML 87
Using coding tools that support XHTML 87
Setting color coding for XHTML 88
Using CodeSweepers to convert your code to XHTML 89
Validating XHTML code 89

Using regular expressions 91
About regular expressions 91
Writing regular expressions 91
Using a special character 92
Creating a single-character regular expression 92
Using a character class 93
Creating a multi-character regular expression 94
Using a back reference 94
Anchoring a regular expression to a string 95
Regular expression examples 95

Using color coding schemes 96
Setting the supported file types for a scheme 96
Setting the display of tags in the Editor 97

Formatting code with CodeSweepers 98
Running a CodeSweeper 99
Managing CodeSweepers 100
Setting Macromedia CodeSweeper options 102

Validating code 104
Using the default Validator 104
Using the CSE HTML Validator 108

Working with tag definitions 109
About namespace precedence 109
Setting namespace precedence 109
Editing tag definitions 110

Chapter 8 Accessing Data Sources 113

Introduction to database tools 114

Working with data sources 114

Configuring a ColdFusion data source 114
Connecting to a data source 114
Viewing a data source 115

Using SQL Builder for database queries 116
The SQL Builder user interface 117
Writing an SQL statement 117
Building a SELECT statement 118
Inserting SQL into a page 119
Testing and editing a query 120

Chapter 9 Managing Projects 121

Understanding projects 122
What is a project? 122
Why use a project? 122
About project folders 122
About the project file 123

Creating a project 124
Setting project options 124
Creating a project 125
Populating a project 125

Working with a project 128
Using the Projects tab 128
Managing project files 129
Managing project resources 130
Performing other project-level tasks 131

Adding a project to source control 132
Why use source control? 132
Supported source control systems 132
Setting up a project in source control 132
Using source control in HomeSite+ for Dreamweaver MX 134
Displaying the Source Control toolbar 134
Sharing project files in Visual SourceSafe 134

Chapter 10 Debugging Application Code 135

Overview of the Interactive Debugger 136

Setting up a debugging session 137

Using the Debugger 140
About the Debugger toolbar 140
Running the Debugger 140
About the Debug window 141
Debugging across multiple pages 141
Stepping through code 142
Evaluating an expression and setting a watch 142

Chapter 11 Deploying Files 143

Setting default deployment options 144

Deploying a single file or folder 145

Performing a custom deployment 146
Selecting folders and files to deploy 146
Adding a deployment server 148
Running the Deployment Wizard 149

Saving deployment results 153

Chapter 12 Testing and Maintaining Web Pages 155

Working in the Results window 156
Opening the Results window 156
Saving results 157

Searching documents 158
Selecting search text 158
Saving search text 158
Using basic search commands 159
Using extended search commands 160
Searching with regular expressions 164

Checking spelling 165
Configuring the spelling checker 165
Using the spelling checker 168

Verifying links 169

Using Site View to check page links 171

Testing page download times 172

Chapter 13 Customizing the Development Environment . . 175

About Visual Tools Markup Language (VTML) 176

Using Tag Chooser 177

Exploring the new VTML structure 178

About dialog definition files 179
Category tag 179
Element tag 179

Creating a tag definition file 180
Tag definition file structure 180
Defining attributes 181
Defining attribute categories 181

Building a tag editor 182

Defining controls 182
Populating dialog boxes with tag data 183
Generating a tag 183

Adding tag Help 186

Container and Control examples 187
TabDialog 187
TabPage 187
Panel 187
Label 188
DropDown 188
ListBox 188
FontPicker 189
ColorPicker 189
Checkbox 189
RadioGroup 189
TextArea 190
SQLTextArea 190
FileBrowser 191
Image 191
StyleTextBox 191
ActiveX 191

Building a custom wizard 192

Creating a wizard definition page 193
Dynamic expressions in tags 193
Bound controls 193
Wizard definition page example 193

Creating a wizard output template 196
Parameters 196
Expressions and functions 196
WIZ Tags 197
Special considerations 197

Wizard definition page library 198
Examples 198
SelectDataSource 198

Chapter 14 Scripting the Visual Tools Object Model 201

Writing and executing scripts 202
The VTOM hierarchy 202
Writing a script 202
Executing a script 203
Creating a custom toolbutton or toolbar 204

Application object 205
Properties 205
Methods 212
Toolbar and toolbutton methods 230

ActiveDocument object 236
Properties 236
Methods 240

DocumentCache object 247
Properties 247

Project object 250
Properties 250
Methods 250

ProjectManager object 252
Properties 252
Methods 252
Folder methods 254
Deployment methods 256

DeploymentManager object 258
Properties 258
Methods 259
Project folder names 263

HTTPProvider object 266
Properties 266
Methods 271

ZIPProvider object. 276
Properties 276
Methods 277

ActiveScripting examples 282
JScript 282
VBScript 284

Third-party add-ins 286
Running a script at startup 286
Sample startup script 286

Table of CommandID values 288

Table of SettingID values 292

Glossary 301

About This Book

Using HomeSite+ for Dreamweaver MX is designed to familiarize you with the product's user interface and the productivity tools that you can use to quickly develop high-quality applications and web content.

Contents

- Intended audience xii
- Developer resources xii
- About the documentation xiv
- Getting answers xvi
- Contacting Macromedia xvi
- Copyright information xvii

Intended audience

This book is intended for professional web developers who have a working knowledge of HTML and web server environments. It introduces you to the user interface and development tools, and provides instructions for installing, configuring, and using the product.

Developer resources

Macromedia Corporation is committed to setting the standard for customer support in developer education, technical support, and professional services. Therefore, the Macromedia web site provides a wealth of online resources.

This section also describes other resources available for HomeSite+ for Dreamweaver MX developers.

Macromedia website

The Macromedia website is designed to give you quick access to the entire range of online resources, as described in the following table:

Resource	Description	URL
Macromedia website	General information about Macromedia products and services	www.macromedia.com/
Product information	Detailed product information on Macromedia products and related topics	www.macromedia.com/software/
Technical support	Macromedia HomeSite Support Center with links to many support services Search the repository of technical articles on Macromedia products	www.macromedia.com/support/homesite
Installation support	Support for installation-related issues for all Macromedia products	www.macromedia.com/support/homesite/installation.html
Support forums	Online access to experienced developers and Macromedia support staff, where you can post messages and read replies on subjects relating to HomeSite+ for Dreamweaver MX features	http://webforums.macromedia.com/homesite/
Developer Center	All the resources that you need to stay current in your skills, including online discussion groups, Knowledge Base, technical papers and more	www.macromedia.com/desdev/developer/
Browser testing sites	Links to websites that provide online testing services, as browser compatibility is still an important issue for website developers.	http://builder.cnet.com/webbuilding/0-7255-8-5801921-1.html

Resource	Description	URL
Professional education	Information about developer certification and the classes, on-site training, and online courses offered by Macromedia	www.macromedia.com/support/training/
Macromedia alliances	Connection with the growing network of solution providers, application developers, resellers, and hosting services that create solutions with Macromedia products	www.macromedia.com/partners/

ColdFusion web resources

Following are just a few of the many sites dedicated to ColdFusion Markup Language (CFML) development:

- CF Advisor Online at http://www.cfadvisor.com/api-shl/engine.cfmwww.cfadvisor.com/api-shl/engine.cfm/a.
- Haznet's Fallout shelter, a CF portal, at http://www.hudziak.com/haznet/cfml.htmlwww.hudziak.com/haznet/cfml.html/a.
- ColdFusion Developer's Journal, an online version of the popular print journal, at http://www.sys-con.com/coldfusion/index2.cfmwww.sys-con.com/coldfusion/index2.cfm/a.

Tutorials

The World Wide Web Consortium (W3C) at www.W3Schools.com offers free web tutorials.

About the documentation

The documentation is designed to provide support for the complete spectrum of participants. The print and online versions are organized to allow you to quickly locate the information that you need. The online documentation is provided in HTML and Adobe Acrobat formats.

You can also access release notes, product support information, and several developer resources from the Help menu.

The rest of this section describes the conventions used in the documentation and the contents of the HomeSite+ for Dreamweaver MX documentation set.

Documentation conventions

When reading the documentation, note formatting cues, as described in the following table:

Type of information	Convention
Code sample	Set in a `monospaced` font
Levels to access a dialog box or pane	Separated by the greater than sign (>), and the path is set in bold. Following are two examples: • "Select **File > New**" means "Select New from the File menu." • "Select **Options > Settings > Editor > Auto Completion**" means "Select Settings from the Options menu. In the Settings dialog box, expand the Editor node and select Auto Completion."
Book titles and emphasized text	Set in *italics*
New terms	Set in **boldface**

Viewing online documentation

You can view Help for a specific tag, Help topics in HTML format, or online documentation in Adobe Acrobat (PDF) format.

To view Help for a tag:

- Position the cursor in a tag and press F1 or right-click and select Edit Tag. Help for the selected tag appears, including syntax and usage information.

F1 Help is available for all supported languages. To view a list of the supported languages, open Tag Chooser (Ctrl+E).

To view the online HTML Help:

1 Select **Help > Open Help References Window**.

The Help tab of the Resources window appears, displaying a tree of online books that includes several language references, as well as the documentation.

2 Find the Help topic you need, using the tree, search engine, index, or bookmarks.

For details, see "Getting the most from the online Help system" on page 35.

3 Double-click a topic to display it in the Document window.

To view the PDF documentation:

Open the PDF documentation from the product CD-ROM or download it from the HomeSite+ for Dreamweaver MX section of the Macromedia website.

Printing online documentation

You can print one Help topic at a time in the HTML Help, or print as many pages of the Help as you need from the Adobe Acrobat (PDF) format of the Help.

To print a single Help topic in the HTML Help:

1 Display the Help topic that you want to print.

For instructions, see "Viewing online documentation" on page xiv.

2 Right-click the topic in the Document window and select your browser's command to print.

To print several pages from the PDF documentation:

1 Open the PDF documentation from the product CD-ROM or download it from the HomeSite+ for Dreamweaver MX section of the Macromedia website.

2 Print as many pages of the documentation as you need.

For more information, see the Adobe Acrobat Reader online Help.

Getting answers

One of the best ways to solve particular programming problems is to tap into the vast expertise of the HomeSite+ for Dreamweaver MX developer communities which you can by following the Community links on http://www.macromedia.com/software/homesite/.

Other developers on the forum can help you figure out how to do just about anything with http://www.macromedia.com/software/homesite/. The search facility can also help you search messages from the previous 12 months, allowing you to learn how others have solved a problem that you might be facing. The forum is a great resource for learning HomeSite+ for Dreamweaver MX, and it is also a great place to see the HomeSite+ for Dreamweaver MX developer community in action.

Contacting Macromedia

Corporate headquarters

Macromedia, Inc.
600 Townsend Street
San Francisco, CA 94103

Tel: 415.252.2000
Fax: 415.626.0554

Web: http://www.macromedia.commacromedia.com/a

Technical support

Macromedia offers a range of telephone and web-based support options. Go to http://www.macromedia.com/support/www.macromedia.com/support//a for a complete description of technical support services.

All user forums are listed on the Macromedia Forums home page at http://www.macromedia.com/support/forums/ http://www.macromedia.com/support/forums//a. You can post an entry anytime.

Sales

Toll-free: 888.939.2545

Tel: 617.219.2100
Fax: 617.219.2101

E-mail: mailto:sales@macromedia.comsales@macromedia.com/a

Online store: http://dynamic.macromedia.com/bin/MM/store/US/home.jsphttp://dynamic.macromedia.com/bin/MM/store/US/home.jsp/a.

Copyright information

Chapter 1

Setting Up the Product

This chapter describes how to install or upgrade HomeSite+ for Dreamweaver MX and how to initially configure it.

Contents

- System requirements 2
- Installing or upgrading 2
- Completing the setup 3
- Monitoring system resources 6

System requirements

The minimum installation requirements are as follows:

- Pentium-compatible processor (Pentium II or higher recommended)
- Microsoft Windows® 98, Windows ME, Windows NT® 4.0, Windows 2000, or Windows XP
- Internet Explorer 4.5 or later
- 128 MB available RAM
- 800 x 600 pixel screen resolution with 256-color display
- 200 MB of hard disk space
- ColdFusion® Server for debugging, database access, and deployment
- CD-ROM drive for packaged (not trial) version

Installing or upgrading

This section contains instructions for installing and upgrading HomeSite+ for Dreamweaver MX. The installation also installs TopStyle 2.1 Lite, its integrated style sheet editor, unless your computer already has TopStyle version 2.1 or 2.5 installed.

Do not install an earlier version of HomeSite+ for Dreamweaver MX over a later version; however, later versions are backward-compatible with earlier versions. If both this version and a previous version are installed on your computer, you can run either version, but the earlier version will reflect changes made in the current version to searching, source control and projects, debugging, FTP, RDS, and validation.

HomeSite+ for Dreamweaver MX records any errors that are encountered during the installation to install.log in the root directory. Please be ready to send this file to Technical Support if you need help with the installation.

To install or upgrade:

1 If you are upgrading, you can uninstall the previous version before installing the new version, but do *not* delete the UserData and AutoBackup folders from the previous installation. If you delete them, you cannot import your customizations, or the files that were created by Auto Backup, into the new version.

2 Close all open applications and insert the HomeSite+ for Dreamweaver MX CD-ROM.

3 When the initial window displays, click Install.

4 Follow the instructions in the Installation Wizard.

5 After the program files are copied to your computer, select the option to restart your computer and click Finish.

Note
If you would like to configure Microsoft Internet Explorer so that it opens this product as the default HTML editor (instead of Microsoft Frontpage, for example), see Knowledge Base article http://www.allaire.com/handlers/index.cfm?ID=10425&Method=Full10425/a on the Macromedia website.

Completing the setup

This section explains how to initially set up HomeSite+ for Dreamweaver MX if you have installed it for the first time, and how to set up HomeSite+ for Dreamweaver MX based on the options that were set in a previous version. The Initial Configuration Wizard (ICW) guides you through both of these processes.

Note
The English language version of HomeSite+ for Dreamweaver MX is installed with support for Chinese (traditional and simplified), Korean, and Japanese character sets on localized Windows systems. To enter characters from these sets, use the Microsoft input method. This support depends on your system language, and does not work if the Western language option is set. In Windows, you can reset your system language from the Control Panel, in Regional Settings or Regional Options.

About the Initial Configuration Wizard

The Initial Configuration Wizard (ICW) lets you carry over options from a previous version and also set new options.

The first time you run the current version of HomeSite+ for Dreamweaver MX, the ICW automatically starts, and it checks for a previous version:

- If one is found, you can selectively import options from the previous version.
- If one is *not* found, the Wizard prompts you to set a few initial options.

When you are finished, HomeSite+ for Dreamweaver MX starts.

You can run the ICW again at any time by selecting **File > New** and then selecting Initial Configuration Wizard from the Custom tab.

Using the Initial Configuration Wizard

This section explains how to complete the ICW after a first-time installation on a computer, and after an upgrade.

The ICW automatically starts when you run HomeSite+ for Dreamweaver MX for the first time.

To complete the ICW after a first-time installation:

1 In the initial Welcome dialog box of the Wizard, click Next.

2 In the Debug Settings dialog box, add, modify, or delete RDS server configurations. Click Next.

3 In the Debug Mappings dialog box, add, modify, or delete mappings to RDS servers. Click Next.

You can view or modify this information later using the Debug menu commands.

4 In the Development Style dialog box, select the language toolbars to include in the QuickBar, and select or browse to the template that you want to use as your default for all HomeSite+ for Dreamweaver MX documents.

5 In the Perform Upgrade dialog box, confirm your selections (you can click Back and make changes if necessary), and then click Finish.

The initial configuration is complete and you can start using HomeSite+ for Dreamweaver MX.

To complete the ICW after an upgrade:

1 In the initial Welcome dialog box of the Wizard, click Next.

2 In the Upgrade Product Settings dialog box, note the items that you can upgrade, based on the changes made in the previous version. Click Next.

What appears next depends on the changes made in the previous version:

- If remote servers were defined in the previous version, the Upgrade Remote Servers dialog box appears.

  Select every server that you need for the current version or any previous version. (If you do not select a server in this list, it will no longer work in the previous version either.) Click Next.

- If toolbars were added or modified, the Upgrade Toolbars dialog box appears.

  Select every toolbar change that you want to carry over to the current version. Click Next.

- If the previous version contained customizations such as shortcuts, snippets, code templates, and modifications to Tag Completion and Tag Insight, then the Upgrade Feature Customizations dialog box appears. Select every customization that you want to carry over to the current version. Click Next.

3 In the Preserve Editor Settings dialog box, select the options to carry over to the current version. Click Next.

   You can change these options later by selecting **Options > Settings** and making changes in the Editor pane, Validation pane, and the panes underneath Editor.

4 In the Preserve General Settings dialog box, select the customizations to carry over to the current version. (Only nondefault options appear.) Click Next.

   You can change these options later by selecting **Options > Settings** and making changes in the following panes: General, Markup Languages, Startup, Locations, File Settings, Browse, Spelling, Dreamweaver/UltraDev, Projects, and Tag Definitions Library.

5 In the Miscellaneous Settings dialog box, select the items whose options you want to carry over to the current version. For example, to carry over color coding customizations, select Color Settings. Or if Auto Completion was enabled in the previous version, and you want to keep it turned on in the current version, select Auto Completion. Click Next.

6 In the Debug Settings dialog box, add, modify, or delete RDS server configurations. Click Next.

7 In the Debug Mappings dialog box, add, modify, or delete mapping to RDS servers. Click Next.

   You can view or modify this information later using the Debug menu commands.

   Note
   To import a localhost RDS development mapping from a previous version, complete the Wizard, open HomeSite+ for Dreamweaver MX, expand the Macromedia FTP & RDS node on one of the Files tabs, right-click the localhost server and select Delete server. Run the Wizard again and import the previous version of the localhost RDS development mapping.

8 In the Development Style dialog box, select the language toolbars to include in the QuickBar, and select or browse to the template that you want to use as your default for all HomeSite+ for Dreamweaver MX documents.

9 In the Perform Upgrade dialog box, confirm your selections (you can click Back and make changes if necessary), and then click Finish.

The initial configuration is complete and you can start using HomeSite+ for Dreamweaver MX.

Monitoring system resources

On Windows 98 and Windows ME platforms, HomeSite+ for Dreamweaver MX uses a great amount of Windows Graphics Device Interface (GDI) and user resources. This section describes the resource problem and how to mitigate it.

For more information, also see http://www.allaire.com/Handlers/index.cfm?ID=21011&Method=FullKnowledge Base article 21011/a on the Macromedia website.

Releasing system resources taken at startup

You can free up Windows 98/ME resources by preventing unnecessary programs and processes from running at startup.

To release resources that Windows 98/ME takes at startup:

1 In Windows, select **Start > Run**.

2 In the Run dialog box, enter `msconfig` and click OK.

3 Clear the options that you are absolutely certain that you do not need, use, or have. Record your changes as you proceed, in case you need to change back.

4 Click Apply.

For a useful description of memory usage on Windows 98/ME systems, see the *Windows 9.x System Resources* article on http://www.windows-help.net/techfiles/win-resources.htmlInfiniSource/a.

After reclaiming these resources, if you still encounter difficulties with system resources, you might benefit from configuring the Resource Level Monitor.

About the Resource Level Monitor

The Resource Level Monitor runs in the background when you start HomeSite+ for Dreamweaver MX and warns you when resources reach a critical level, enabling them to save documents and close the application before a crash occurs. However, the warning dialog box was not labeled with the application name, and users had to edit the Windows registry to configure the monitor.

As a result, the Resource Level Monitor has been enhanced for this release so that the warning dialog box includes the application name in the window title and in the message text, to indicate that it is generated by HomeSite+ for Dreamweaver MX. Also, users can now configure the monitor in a graphical user interface.

Note
You must have rsrc32.dll installed on your computer to use the Resource Monitor. This DLL file comes from the Windows System Resource Monitor.

Using the Resource Level Monitor

This section explains how to respond to warnings and how to configure the monitor.

Responding to warnings

A warning dialog box appears if your available Windows Graphics Device (GDI) or user resources drop below your default warning thresholds. For example, it appears if your default warning threshold for GDI resources is 15% and your available GDI resources drop to 14%.

For best results, save your work, even if you choose to continue. If you continue, the warning dialog box closes and does not appear again unless your system resources drop another 5%; for example, when your GDI resources drop to 9%.

Configuring the monitor

You should configure the Resource Level Monitor if your system crashes without having been warned, or if you are being warned frequently or unnecessarily. You can adjust monitor options until you find a stable level for your computer.

To configure the Resource Level Monitor:

1 In the **Options > Settings > Resource Monitoring** pane, specify any of the following options:

- **Enable resource monitoring** Clear this to disable resource monitoring.

  GDI level threshold (%) Select the minimum percentage of remaining Windows Graphics Device Interface (GDI) resources before the system issues a warning. The default is 15%.
- **User level threshold (%)** Select the minimum percentage of remaining user level resources before the system issues a warning. The default is 15%.
- **Monitoring interval (minutes)** Select a time interval in minutes for the system to monitor your available resources.

2 Click Apply.

Chapter 2

Configuring Browsers and Servers

This chapter contains instructions for setting up browsers and servers.

Contents

- Configuring the internal browser 10
- Configuring an external browser 12
- Working with files on remote servers 14

Configuring the internal browser

You can use the internal browser to browse documents and application pages from within HomeSite+ for Dreamweaver MX.

With a server mapping, you can also preview server-side processes in your page; for example, server-side includes, results from submitting a form to a CGI program, and the results of server-side code. For more information, see “About server mappings” on page 20.

You can also use the Browse toolbar to view pages in external browsers, test the page in different screen resolutions, and more. For more information, see “About the Browse tab” on page 29.

To set internal browser options:

1 In the **Options > Settings > Browse** pane, select one of these browser options:

- **Use Microsoft Internet Explorer** Uses Microsoft Internet Explorer (version 3.01 or later) as the internal browser. To download the latest version, see http://www.microsoft.com/downloads/http://www.microsoft.com/downloads//a.
- **Use Netscape** Uses Netscape (version 6 or later) as the internal browser. For this option you must correctly install and configure the Mozilla browser NGLayout/Gecko control. (Please note that the Mozilla control is under continual development, and could lack stability.)

  For setup instructions, “To install and configure Mozilla:” on page 10.
- **Use the built-in browser** Uses the built-in browser as the internal browser. Please note that this browser has only limited support of HTML and browser extensions.

You can only use one internal browser at a time, but you can change this setting as needed. Restart the product for the change to take effect.

2 To have the same file saving behavior when viewing pages in the internal browser as when viewing pages in the external browser, select Use External Browser Configuration for Internal Browser.

For more information, see “Configuring an external browser” on page 12.

To install and configure Mozilla:

1 Download a Mozilla build from http://ftp.mozilla.org/pub/mozilla/releases/http://ftp.mozilla.org/pub/mozilla/releases//a.

For best results, download the Mozilla 0.8.1 build created on 8/28/2001, from http://ftp.mozilla.org/pub/mozilla/releases/mozilla0.8.1/mozilla-win32-0.8.1.ziphttp://ftp.mozilla.org/pub/mozilla/releases/mozilla0.8.1/mozilla-win32-0.8.1.zip/a.

2 Unzip the Mozilla binary files into a new directory on your computer.

3 Add the Mozilla bin directory to your `PATH` environment variable. Reboot if necessary. If you are using Windows 95/98, you may have to specify the short name version of the path; for example, c:\mozil~1\bin.

4 If your Mozilla bin directory does not have component.reg, or it is small (< 5k), then delete it and run mozilla.exe. This generates a new component.reg file.

   This file is the database that XPCOM uses to create new objects. If it is not there or it is empty, the Mozilla control will not be able to create Gecko components.

5 Open a Command prompt and change to the Mozilla bin directory. For example, if the bin directory is in D:\mozilla, enter `d:` and then enter `cd mozilla\bin`.

6 Enter `regsvr32 mozctlx.dll`.

   If this does not work, the directory containing regsvr32.exe is not in your `PATH` variable. Use the Windows Find Files utility to locate the program and then run it using its full path; for example `c:\winnt\system32\regsvr32.exe mozctl.dll`.

7 Run regedit.exe and, under HKEY_LOCAL_MACHINE\Software, create a new key called Mozilla; for example, HKEY_LOCAL_MACHINE\Software\Mozilla.

8 Create a string value under this key called BinDirectoryPath, with a value of the path to the Mozilla bin directory; for example, c:\mozilla\bin.

For more information, see Knowledge Base article http://www.allaire.com/Handlers/index.cfm?ID=9927&Method=Full9927/a on the Macromedia website.

Configuring an external browser

As part of the installation process, the program compiles a list of the web browsers it detects on your computer. You can view, add, edit, and remove browser configurations; you can change the default browser; and you can set the file saving behavior for your browsers.

The following procedures describe how to configure the external browsers.

To view the list of browsers:

- Select **Options > Configure External Browsers**.

The External Browsers dialog box appears.

To add a browser to the list:

1 Select **Options > Configure External Browsers**.

2 In the External Browsers dialog box, verify that the browser is installed on your computer, and then click Add.

3 Complete the Browser dialog box, as follows:

- **Name** Enter a name for the new browser.
- **Use DDE** Select this if the browser uses Dynamic Data Exchange (DDE) for object linking and embedding. (Most browsers use Object Linking and Embedding (OLE) instead.)
- **Location** Click the file icon, find the browser program file, and click Open.

4 Click OK.

The External Browsers dialog box displays the new browser configuration.

5 Click OK.

HomeSite+ for Dreamweaver MX saves your changes.

To edit a browser in the list:

1 Select **Options > Configure External Browsers**.

2 In the External Browsers dialog box, select a browser and click Edit.

3 (Optional) In the Browser dialog box, change information for the browser:

- **Name** Enter a different name for the browser.
- **Use DDE** Select this if the browser uses Dynamic Data Exchange (DDE) for object linking and embedding. (Most browsers use OLE instead.)
- **Location** Select a different program file for the browser (for example, select a later version of the browser).

4 Click OK.

5 In the External Browsers dialog box, click OK.

HomeSite+ for Dreamweaver MX saves your changes.

To remove a browser from the list:

1 Select **Options > Configure External Browsers**.

2 In the External Browsers dialog box, select a browser and click Delete.

3 Click Yes to confirm.

4 Click OK.

HomeSite+ for Dreamweaver MX saves your changes.

To change the default browser:

1 Select **Options > Configure External Browsers**.

2 In the External Browsers dialog box, select the appropriate browser, and click the up arrow until it is the first item in the list.

3 Click OK.

HomeSite+ for Dreamweaver MX saves your changes.

To set the file saving behavior for your browsers:

1 Select **Options > Configure External Browsers**.

2 In the External Browsers dialog box, select one of the following options:

- **Prompt to save changes to the current document** Asks you whether to save the current document before opening it in the external browser.
- **Automatically save changes to the current document** Saves the current document before opening it in the external browser.
- **Browse using a temporary copy (no need to save)** Copies the current document and opens this copy in the external browser. This option is useful when you are making many changes to a page, but it requires more system resources.

3 Click OK.

HomeSite+ for Dreamweaver MX saves your changes.

Note
If you select the Use External Browser Configuration for Internal Browser option, the file saving behavior that you specify here also applies to the internal browser.

Working with files on remote servers

Working with directories and files on remote servers is similar to working with them on local or network drives. For example, when you save files, changes are saved to the remote server. The primary difference is that you must establish a connection to a remote server before you can work with its files.

Using the SNE node

You can use the Macromedia FTP & RDS node, called the **Shell Namespace Extension** (SNE) node, to add FTP and RDS servers. Then you can work with files on configured remote servers from within HomeSite+ for Dreamweaver MX and in Windows Explorer, as the following describes:

- **I**n HomeSite+ for Dreamweaver MX Display a Files resource tab and, in the top pane, select *My Computer* from the drop-down list.

  The top pane displays all of your drives and SNE node servers.

- **In Windows Explorer** Find the SNE node under *My Computer*, and access the files under the SNE node in the same way as you access files in a folder.

  If you cannot find the SNE node, open the **Options > Settings > File Settings > FTP & RDS** pane and select the **Enable Explorer shell integration** option.

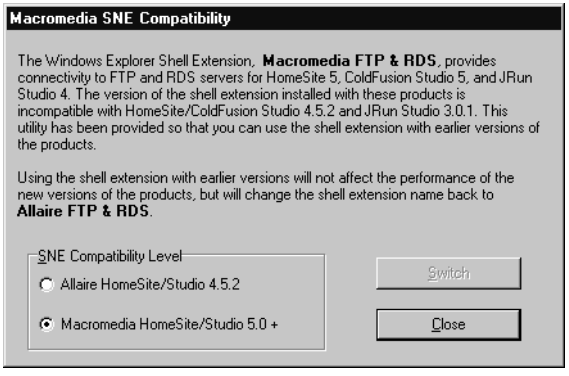
  You must also select this option to debug pages and access RDS servers.

In previous releases, the SNE node was called *Allaire FTP & RDS*. In this version, the node is called *Macromedia FTP & RDS*. If you install this version on a computer that has a previous version of the product installed, the previous version no longer recognizes the node. To fix this, you must modify the SNE node.

To use the SNE node with earlier versions:

1 After completing the installation, run the SNECompatibility.exe program in the program directory; for example, C:\Program Files\Macromedia\HomeSite+\SNECompatibility.exe.

  The Macromedia SNE Compatibility dialog box appears:



2 Select the appropriate compatibility option:
   - **Allaire HomeSite/Studio 4.5.2** Enables the remote server functionality in both this version and in previous versions, and changes the name of the SNE node to "Allaire FTP & RDS."
   - **Macromedia HomeSite/Studio 5.0+** Enables the remote server functionality only in this version.
3 Click Switch, then click Close.

Connecting to a server

HomeSite+ for Dreamweaver MX provides FTP server access, and secure HTTP access through Remote Development Services (RDS).

RDS lets you communicate using HTTP with ColdFusion on your local computer and on any configured remote host. RDS features include:

- Secure server access
- Remote file access
- Data source browsing and SQL query building
- CFML debugging

You must have an RDS server connection to access data sources and to debug pages.

Required server information

You must have specific information about a remote server to connect to it, such as its host name and user access requirements. You can obtain this information from the owner of the server, whether it is your IT group, an ISP, or another provider.

Connecting to an FTP server

You can remotely access FTP servers for file transfers and other site management tasks using the Macromedia FTP & RDS node. After the connection is established, you can access files on remote servers and maintain websites located anywhere on the Internet.

FTP server connection requirements vary greatly. The following instructions work in the majority of cases.

To connect to an FTP server:

1 On the Files resource tab, in the top pane, select Macromedia FTP & RDS from the drop-down list.

- If you see *Allaire FTP & RDS* in the list, you can select this instead. For more information, see "Using the SNE node" on page 14.
- If you do not see any FTP & RDS node in the list, select **Options > Settings > File Settings > FTP & RDS > Enable Explorer shell integration**.

2 In the list of servers in the top pane, right-click the top-level node labeled *Macromedia FTP & RDS* and select Add FTP Server.

3 Complete the Configure FTP Server dialog box, as follows:

Field	Description	Comment
Description	Name for the FTP server to display in the Files tab and in Windows Explorer.	
Host Name	Server's IP address or domain name, such as macromedia.com.	Servers with ftp as part of the domain name require you to enter the complete name; for example, ftp.somesite.com.
Initial Directory	Top-level directory for the account.	This is optional for accounts that default to the root directory of the FTP server, but it is required if the account specifies an initial directory.
Relative from server-assigned directory	Option to specify if the initial directory should be set from the server-assigned user directory or from the server root.	Clear this option if the Initial Directory should be set from the server root.
Username	Login name for the account used to access the remote server, or "anonymous" for anonymous FTP servers.	If you leave this field blank, you must provide a username when you log in.
Password	Password for the account used to access the remote server.	If this field is left blank, you are asked for a password when you log in.
Root URL	The http:// address of the site.	This setting allows you to browse files that were opened from the remote server, edited, and saved.
Remote Port	Port used by the FTP server.	Use the default 21, unless the server administrator or ISP specifies another port.

Field	Description	Comment
Request Timeout	Maximum number of seconds to wait for a server connection to complete.	
Disable passive mode	Option to select if the server does not use passive connections.	
File time offset	The number of hours ahead or behind the current local time to use in the date and time information for files on the FTP server.	If you set this value, you cannot select the Assume UTC file times option.
Assume UTC file times	Option to adjust date/time information on the FTP server for servers using the Universal Coordinated Time format.	Select this if you see incorrect date/time information for files on the FTP server.

4 Click OK.

You can now work with files on the server.

To do a file size check after transferring a file:

1 Open a local file.

2 Select **File > Save As**, and save the local file to the remote server.

3 If nothing happens, the file transfer was successful.

Otherwise, if there is a discrepancy between the byte counts of the local copy and the copy saved to the remote server, an error message appears.

Enabling secure FTP

You can transfer data using Secure Sockets Layer (SSL) to servers that support 40-bit encryption and decryption. HomeSite+ for Dreamweaver MX installs this FTP technology, from Ipswitch, as well as a default certificate and key.

Note
For this release, you cannot successfully transfer Unicode and Unicode Big Endian files through FTP using SSL. Therefore, disable the Secure (SSL) feature in the Configure FTP Server dialog box before transferring these types of files.

To enable SSL for an FTP connection:

1 On the Files resource tab, in the top pane, with Macromedia FTP & RDS selected in the drop-down box, right-click the FTP server and select Properties.

2 In the Configure FTP Server dialog box, select the Secure (SSL) option.

3 Click Configure SSL and enter the appropriate certificate information.

For more information, click Help.

4 Click OK.

The following table summarizes the error messages that you could receive:

Error message	Cause
Unable to connect to the FTP server.Success Remote Server Operation Failure: Winsock Error: Connection aborted.	SSL has not been enabled in the Configure FTP Server dialog box, and you attempted to connect to a server requiring SSL.
Secure Connection Error - you have requested a secure connection to the server but either the server does not support SSL or WS_FTP could not negotiate a secure connection. This connection is NOT secure. Do you wish to continue?	An SSL-enabled connection is selected at startup, but SSL is disabled on the server.

To revert to the installed certificate and key:

1 In the **Options > Settings > File Settings > FTP & RDS** pane, select the **Reset FTP SSL Certificate to Default** option.

2 Click Apply.

Connecting to an RDS server

Complete the following configuration procedure for each ColdFusion Server that you want to access.

To connect to an RDS server:

1 On the Files resource tab, in the top pane, select Macromedia FTP & RDS from the drop-down list.

- If you see *Allaire FTP & RDS* in the list, you can select this instead. For more information, see "Using the SNE node" on page 14.
- If you do not see any FTP & RDS node in the list, select **Options > Settings > File Settings > FTP & RDS > Enable Explorer shell integration**.

2 In the list of servers in the top pane, right-click the top-level node labeled *Macromedia FTP & RDS* and select Add RDS Server.

3. Complete the Configure RDS Server dialog box, as follows:

Field	Description
Description	Enter a label for the ColdFusion Server connection.
Host name	Enter localhost or an IP address.

Field	Description
Port	Accept the default Port or enter your ColdFusion Server port number.
Use Secure Sockets Layer	Select this option if needed for SSL security.
Remaining RDS security fields	Complete these fields based on the settings in the ColdFusion Administrator. For details, see “Managing ColdFusion security for RDS” on page 19.

4 Click OK.

The ColdFusion Server appears in the Macromedia FTP & RDS list on the Files resource tab and in Windows Explorer.

Managing ColdFusion security for RDS

The ColdFusion installation configures basic security for the server and, by default, requires a password for the ColdFusion Administrator and for HomeSite+ for Dreamweaver MX.

To change the default security settings:

1 Open ColdFusion Administrator.

2 Select Basic Security from the ColdFusion Server link list.

3 Change the password security settings.

For more information, see the ColdFusion documentation.

4 Click Apply.

Managing servers

You can view, change, and delete server configurations.

To view and edit the configuration for a server:

1 On the Files resource tab, in the top pane, with Macromedia FTP & RDS selected in the drop-down box, right-click the server and select Properties.

2 (Optional) Make changes in the server configuration.

3 Click OK.

HomeSite+ for Dreamweaver MX saves the changes to the server configuration.

To delete a server:

1 On the Files resource tab, in the top pane, with Macromedia FTP & RDS selected in the drop-down box, right-click the server and select Delete Server.

2 In the confirmation dialog box, click Yes.

HomeSite+ for Dreamweaver MX removes the remote server from the Macromedia FTP & RDS list.

About server mappings

A development mapping is essential for working with files using Remote Development Services (RDS). With mappings, you can process pages on a server from within HomeSite+ for Dreamweaver MX, and you can debug application code on a remote server using RDS.

To add a server mapping, see "Adding a server mapping" on page 23.

Mapping for page processing

By default, when you browse a page in the internal or external browser, HomeSite+ for Dreamweaver MX opens the page from the local file system, or returns it using FTP from a remote server. That is adequate for checking page content and formatting. For developing a website, however, you must see dynamic pages as visitors to your site will experience them.

To do this, you can route the documents through a web server. The server software can be on your local computer, a network server, or a remote system. So, instead of opening the files, an HTTP request for the page is sent to the server. If any server-side processing is required, such as for CFML, JavaServer Pages (JSP), or other scripting languages, the web server transfers the page to the appropriate server for further processing, and then returns it to the browser. This is valuable for previewing applications and site elements in a test environment before deploying the site.

You establish this routing by associating the physical directories where your files are stored with the server that processes the files. This association is called a **mapping**. A wide range of web servers is supported, so you can create multiple mappings and select which server to use for processing. Consult your server documentation or provider for the specifics of accessing server directories.

To use server mappings, you must configure Microsoft Internet Explorer or Netscape Mozilla NGLayout as the internal browser. For more information, see "Configuring the internal browser" on page 10.

Mapping for debugging

If you have ColdFusion and HomeSite+ for Dreamweaver MX on the same computer, you can run the debugger against files opened from the file system, including mapped network drives. But if you are working with a remote ColdFusion Server using RDS, you must set a mapping to that server to run the debugger.

To debug applications in HomeSite+ for Dreamweaver MX, you also must complete the procedures in "Debugging Application Code" on page 135.

Understanding RDS mappings

A file mapping ensures that HomeSite+ for Dreamweaver MX, the RDS server, and your browser can resolve a local path into a server path and URL. The path fields in the Configure RDS Server dialog box specify how each — HomeSite+ for Dreamweaver MX, the RDS server, and your browser—can find the directory that you are mapping.

Note
An error occurs if you try to browse a file when the HomeSite+ Path set in an RDS mapping does not match the path of the active file. For example, you cannot browse a file opened from a mapped drive unless that drive path matches the mapping path.

The next section presents the most common RDS mapping scenarios.

RDS file mapping examples

The following scenarios show how file mappings work when you have local or remote files matched with either local or remote servers:

- HomeSite+ for Dreamweaver MX and ColdFusion on the same computer
- Debugging in HomeSite+ for Dreamweaver MX on a remote ColdFusion Server using one of the following:
  - Drive mappings
  - Network Neighborhood
  - RDS file access

Using local paths

Debugging against a locally installed version of ColdFusion Server (localhost) is the most common scenario. In most cases, this arrangement allows both the ColdFusion Server and HomeSite+ for Dreamweaver MX to see the directories in the same way.

In this scenario, you use mappings to resolve URL paths. The URL part of the mapping determines how HomeSite+ for Dreamweaver MX displays a physical file in a browser.

For example, both HomeSite+ for Dreamweaver MX and the ColdFusion Server might see a file as C:\webprojects\App1\index.cfm, and the browser as http://215.180.21.1/index.cfm.

To resolve the URL paths, you must create a mapping for the App1 directory, as in the following table:

ColdFusion Server and HomeSite+ for Dreamweaver MX on same computer	
HomeSite+ path	C:\webprojects\App1\
ColdFusion Server path	C:\webprojects\App1\
Browser/URL path	http://215.180.21.1/

Using drive mappings

Developers often debug against a remote server across an internal network. In many cases, they use a network drive mapping.

For example, a developer might have a remote drive X mapped to a network shared directory \\MyServer\webprojects\ where webprojects is the name of the shared directory in the network server MyServer.

In this scenario, HomeSite+ for Dreamweaver MX might see a file as X:\App1\index.cfm, the ColdFusion Server as C:\webprojects\App1\index.cfm, and the browser as http://215.180.21.1/App1/index.cfm.

To resolve communication between HomeSite+ for Dreamweaver MX and the ColdFusion Server, you must create a mapping for the App1 directory, as in the following table:

HomeSite+ access to a remote server using drive mappings	
HomeSite+ path	X:\App1\
ColdFusion Server path	C:\webprojects\App1\
Browser/URL path	http://215.180.21.1/App1/

Using UNC paths and the Network Neighborhood

Developers can debug code against remote ColdFusion Servers across an internal network using UNC paths. They often use the Network Neighborhood to access a file on a remote server. For example, a developer might access a file on \\MyServer\webprojects\, where webprojects is the name of the shared directory in the network server MyServer.

In this scenario, HomeSite+ for Dreamweaver MX might see a file as \\MyServer\webprojects\App1\index.cfm, the ColdFusion Server as C:\webprojects\App1\index.cfm, and the browser as http://215.180.21.1/App1/index.cfm.

HomeSite+ for Dreamweaver MX and the ColdFusion Server must understand how a file location appears to the parties involved, so you must create a mapping for the App1 directory, as in the following table:

HomeSite+ accesses files on remote server using UNC paths/Network Neighborhood	
HomeSite+ path	\\MyServer\webprojects\App1\
ColdFusion Server path	C:\webprojects\App1\
Browser/URL path	http://215.180.21.1/App1/

Using Remote Development Services

When developing outside local area networks, you access files on a ColdFusion Server across the Internet using Remote Development Services (RDS).

In this scenario, HomeSite+ for Dreamweaver MX might see a file as RDS://MyRDSserver/C:/webprojects/App1/index.cfm, the ColdFusion Server as C:\webprojects\App1\index.cfm, and the browser as http://215.180.21.1/App1/index.cfm.

Although the server path can be inferred from the local RDS path, you still must create a mapping. In special scenarios, the path resolution from the ColdFusion Server to HomeSite+ for Dreamweaver MX can be ambiguous. Therefore, you must create a mapping for the App1 directory, as in the following table:

HomeSite+ accesses files on remote server using RDS remote file access	
HomeSite+ path	RDS://MyRDSserver/C:/webprojects/App1/
ColdFusion Server path	C:\webprojects\App1\
Browser/URL path	http://215.180.21.1/App1/

Adding a server mapping

This section provides instructions for adding a server mapping in HomeSite+ for Dreamweaver MX, and how to set a default mapping.

For more information on setting up server mappings, see the following articles on the Macromedia website:

- http://www.allaire.com/Handlers/index.cfm?ID=8347&Method=FullHow to set up a server mapping in HomeSite 4.0/a
- http://www.allaire.com/handlers/index.cfm?ID=8500&Method=FullSetting up Server Mappings in HomeSite/a

Note
If you use Personal Home Page Tools (PHP) on an Apache web server, see Dave Alder's article, http://www.roundridge.com/hs/index.phpSetting up server mappings with HomeSite/Apache/PHP/a.

To add a mapping:

1 In the **Options > Settings > Browse** pane, select Microsoft Internet Explorer, Netscape Mozilla NGLayout, or the built-in browser as the internal browser.

2 Click Development Mappings.

3 In the Remote Development Settings dialog box, on the Mapping tab, select a server from the drop-down list of configured servers.

If a server is not on the list, add it. For more information, see "Adding a server mapping" on page 23.

4 Enter path information for the mapping:

- **HomeSite+ Path** Path for HomeSite+ to access a directory
- **ColdFusion Server Path** Path for the ColdFusion Server to access the same directory
- **Browser Path** Path for the internal browser to access the same directory

For page browsing on localhost, only HomeSite+ Path and Browser Path are required.

5 Click Add, then click OK.

6 Click Apply.

HomeSite+ for Dreamweaver MX saves the mapping. You can now browse your documents in the internal or external browser.

To edit a mapping:

1 In the **Options > Settings > Browse** pane, click Development Mappings.

2 In the Remote Development Settings dialog box, on the Mapping tab, select a mapping and click Edit.

3 (Optional) Edit the information for the mapping.

4 Click Update, then click Apply.

To change to the mappings for a different server:

1 In the **Options > Settings > Browse** pane, click Development Mappings.

2 In the Remote Development Settings dialog box, on the Mapping tab, select the server for the mapping that you want to use.

3 Click OK, then click Apply.

To set a mapping as the default:

1 In the **Options > Settings > Browse** pane, click Development Mappings.

2 In the Remote Development Settings dialog box, on the Mapping tab, select the mapping that you want to move.

3 Click the up arrow to move the mapping to the top of the list.

4 Click OK.

Chapter 3

Exploring the Workspace

This chapter acquaints you with the main areas of the user interface. It also guides you in customizing the workspace to make your development work as productive as possible.

Contents

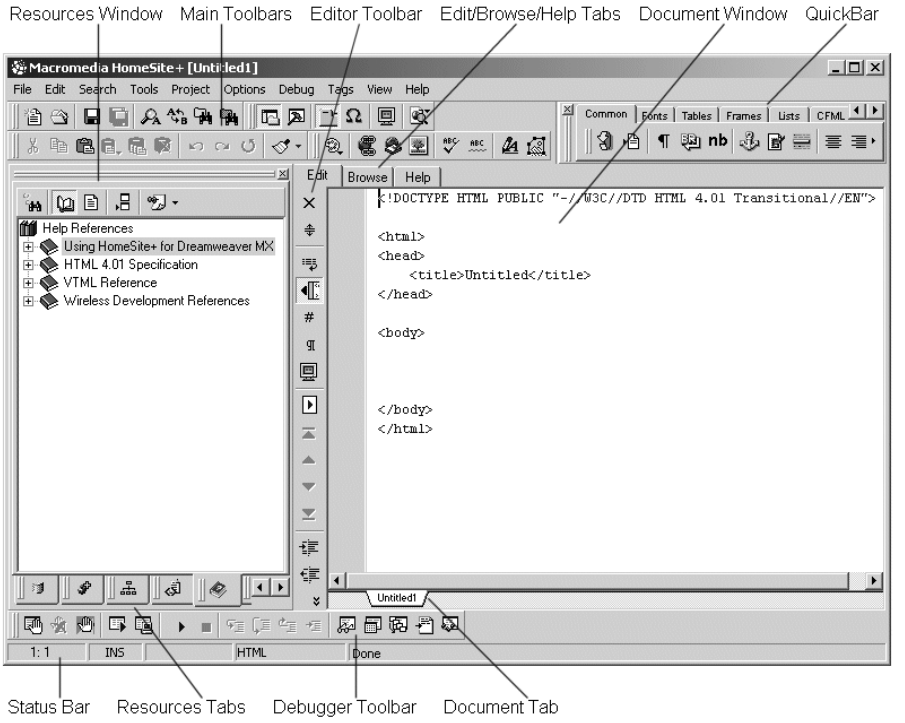
- The workspace areas 26
- Working in the Resources window 28
- Creating and browsing files in the Document window 29
- Tracking your work in the Results window 30
- Customizing the workspace 31
- Getting the most from the online Help system 35

The workspace areas

The term **workspace** describes the user interface that you see when you first load the program. The workspace has four principal areas:

- **Command area** At the top of the workspace is the title bar, which displays the file path of the current document. Below that is the menu bar, which contains standard Windows commands plus development menus. Below the menus are toolbars that provide one-click access to commands and application tools. To the right is the QuickBar, a tabbed toolbar for inserting JSP, HTML, and other web language elements.
- **Resources window** Contains tabs for file management, data sources, projects, code snippets, online Help, and Tag Inspector.
- **Document window** Contains tabs for writing and browsing pages.
- **Results window** Contains tabs to track search and replace operations, code validation, link verification, images, project deployment, and compilation.

The following figure shows the main workspace areas:

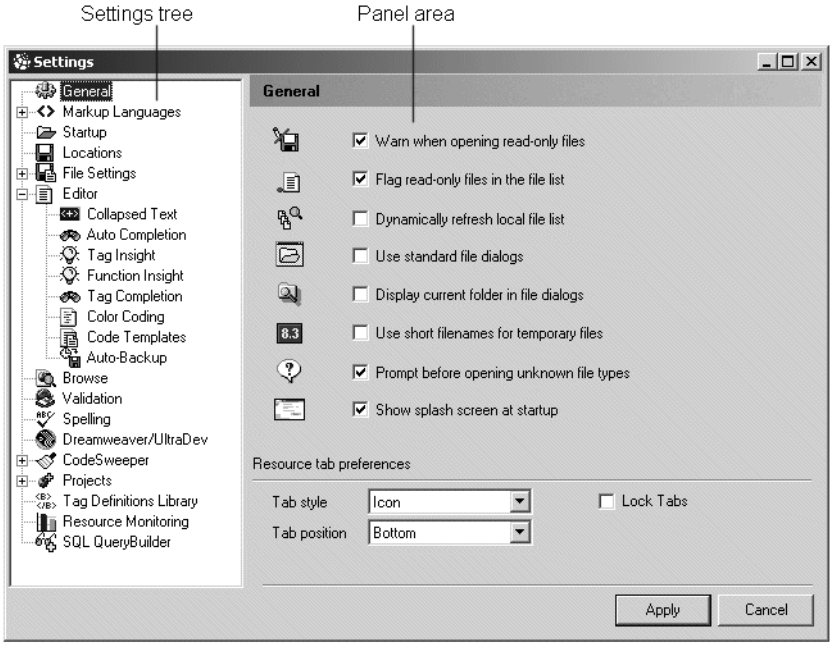


Setting options

The Settings dialog box lets you configure many HomeSite+ for Dreamweaver MX options to fit your work style, and you can easily change settings for different tasks.

To set options:

1 Select **Options > Settings** (F8) to open the dialog box.



2 Click a feature category in the left pane to display options in the right pane. Expand a category to view further options.

3 Click Apply to save the settings.

To customize the workspace, see “Customizing the workspace” on page 31.

Working in the Resources window

The Resources window is a tabbed user interface that gives you quick access to file management, development, and Help resources. Select a tab at the bottom of the window to open the pane for each resource. Context menus and specialized toolbars provide the set of commands for each pane.

The following table describes each tab in the Resources window:

Resources tab	Description
Files 1 and Files 2	Lets you manage files on local and network drives and on remote servers. The drop-down list at the top allows you to pick a drive or server; the bottom two panes display directories and files for the selected drive or server. The two Files tabs, labeled **Files 1** and **Files 2**, facilitate working with files in multiple locations, including directories, drives, and remote systems. For information about how HomeSite+ for Dreamweaver MX determines the current directory with two Files tabs, see "About the Files tabs" on page 42.
Database	Shows your application data sources. When you add a data source, it appears in the drop-down list at the top. Just select a data source to view its schema. You can write and test database queries in the SQL Builder.
Projects	Helps you manage site content by organizing pages and supporting files. You have the option of adding a project to your version source control system. The drop-down list at the top displays recent projects. The bottom two panes display the folders and files for the selected project.
Site View	Provides a visual representation of each link in the current document. Right-click in the pane to select a view type and display options. Right-click a link to expand the view.
Snippets	Provides a convenient place to store code blocks and content for reuse. You can also share snippets with other users.
Help	Contains product documentation and other online resources. You can customize the Help by adding and editing Help files. For more information, see "Extending the Help system" on page 38.
Tag Inspector	Lets you edit in an interactive property sheet user interface. The top pane is the **Tag Tree**, a customizable view of the document hierarchy. The bottom pane is the **Tag Inspector**, where you can edit code instead of working directly in the Document window.

Creating and browsing files in the Document window

You work primarily in the Document window as you develop code and content. The three tabs at the top of the give you access to the following activities:

- **Edit** Develop code and content.
- **Browse** Preview your work in the internal browser.
- **Help** View Help. This tab appears the first time you open a file from the Help tab at the bottom of the Resources window.

The following sections explain how to use each of these tabs.

About the Edit tab

The Edit tab opens the **Editor**, which can be used for everything from simple text editing to developing application code and content.

The Edit toolbar extends vertically to the left of the Document window below the Edit tab when you select the Edit tab. You can position the cursor over each toolbutton to display a descriptive tooltip.

About the Browse tab

The Browse tab displays the current document in the internal browser. This is helpful when you are making many changes to a page that do not require processing by a server; for example, previewing formatting changes. For information on configuring the internal browser, see "Configuring the internal browser" on page 10.

The Browse toolbar displays at the top of the Document window when you select the Browse tab. You can position the cursor over each toolbutton to display a descriptive tooltip. You can use the toolbar to browse a page, refresh the display, and test a page in different screen resolutions. Use Toggle Rulers to frame your page with rulers measuring in pixels, with the 640x480 markers highlighted in red. Use Browser Size to preview a page in a 640x480 or 800x600 screen resolution.

Press F12 to display the current document in the internal browser, or press F11 to view the document in the default external browser.

About the Help tab

This tab displays tag-specific Help and pages from the Help. This tab becomes visible in the workspace upon opening it for the first time.

To open the Help tab, do one of the following:

- Press F1 in a tag editor
- Select a tag in Tag Chooser and press F1
- In the Editor, position the cursor in a tag and press F1
- In the Resources window, on the Help tab, display a Help page

Note
You can cycle through all of the pages that have been opened in the Help tab during the current session. To do this, click in a page and then hold down the Alt key while pressing the left or right arrow key.

Tracking your work in the Results window

The Results window displays the results of document operations in individual panels. It opens automatically when you run an extended search (Extended Find or Extended Replace), Code Validation, Link Verification, Image Thumbnails, or Project Deployment.

Select **View > Results** to display the Results window above the Status bar. You can use the drag bar at the top of the window to resize the view. Right-click any tab to open a menu of commands that apply to the tab.

Customizing the workspace

If you are interested in configuring the workspace to more readily support the way you work, you can start by viewing a list of the functions that are available in HomeSite+ for Dreamweaver MX, to see if you are using the program to its full potential.

To view a list of HomeSite+ for Dreamweaver MX functions:

1 Select **Options > Customize**.

2 In the Customize dialog box, on the Toolbars tab, review the contents of the Toolbuttons list.

3 (Optional) Add a toolbutton for a function to a toolbar. For more information, see "Customizing toolbars" on page 32.

4 Click Close.

There are many ways to customize the look and functionality of your workspace, including setting the display and positioning of toolbars and tabs in the Resources window; assigning shortcut keys for commands, code snippets, and scripts; and creating custom toolbuttons.

For advanced customizations, such as editing tag editors and creating new ones, use the Visual Tools Markup Language (VTML). For more information, see "Customizing the Development Environment" on page 175.

You can extend or change the program's functionality by scripting the Visual Tools Object Model (VTOM). For more information, see "Scripting the Visual Tools Object Model" on page 201.

Managing application toolbars and the QuickBar

The application toolbars and the QuickBar are divided by location, function, and the level of customization available.

The application toolbars, on the left of the workspace, provide standard Windows commands, and access to tools such as external browsers, the Style Editor, Code Validation, and the Link Checker.

The QuickBar, on the right of the workspace, lets you insert tags from any of the supported languages.

You can change the order of tabs on the QuickBar, hide a default toolbar, show a toolbar that is hidden by default, or move a toolbar.

To change the order of tabs in the QuickBar:

1 In the workspace, right-click in any toolbar or the QuickBar, and select Organize QuickBar.

2 In the Organize QuickBar dialog box, select names of tabs and press the Up and Down arrow keys until you are satisfied with the order of the tabs.

The tab on the top of the list appears on the far left side of the QuickBar.

3 Click Close.

To hide or show a toolbar:

1 Select **Options > Customize**.

2 In the Customize dialog box, in the Visible Toolbars box, clear the check box next to every toolbar that you want to hide, and select the check box next to every toolbar that you want to show.

3 If you want to use the default settings for toolbars—which toolbars display in the workspace and in what position—click Reset to Defaults.

This does not override any changes that you have made to toolbuttons.

4 Click Close.

To move a toolbar or QuickBar tab:

1 For a QuickBar tab, verify that the tab is active.

2 Drag a toolbar or QuickBar tab by its handle (the two vertical bars on its left edge) to a new position.

3 To move every toolbar back to its default position in the workspace, select **Options > Customize**. In the Customize dialog box, click Reset to Defaults.

This also resets the default settings for which toolbars are shown or hidden. It does not override any changes that you have made to toolbuttons.

4 To move every floating toolbar and QuickBar tab to its default position in the workspace, right-click in a toolbar or QuickBar and select Dock floating toolbars.

Customizing toolbars

You can add standard and custom toolbuttons to a toolbar, change the order of toolbuttons in a toolbar, remove toolbuttons from a toolbar, and add and remove a custom toolbar. You cannot delete a standard toolbar, but you can hide it.

To add a standard toolbutton:

1 Select **Options > Customize**.

2 In the Customize dialog box, in the Visible Toolbars box, select the toolbar to which you want to add a toolbutton.

If you are using the default workspace, you might want to select the View toolbar, since it has room for a few more toolbuttons.

A picture of the selected toolbar appears near the top of the dialog box.

3 In the Toolbuttons box, scroll until you see the toolbutton that you want to add.

4 Drag the toolbutton to an empty space in the picture of the toolbar.

5 Click Close.

The toolbutton is added to the toolbar in the workspace.

To add a custom toolbutton to a toolbar:

1 Select **Options > Customize**.

2 In the Customize dialog box, in the Visible Toolbars box, select a toolbar.

3 Click Add Custom Button.

4 In the Custom Toolbutton dialog box, specify what you want the custom toolbutton to do, as described in the following table:

For the custom button to do this...	Do this...
Insert start and end tags into the current document	Select Insert custom start and end tags into the current document, and complete the Start Tag and End Tag boxes.
Display a custom dialog box	Select Display a custom dialog, and complete the Dialog File box with the full path of the VTML file.
Launch an application besides Dreamweaver MX, which already has a toolbutton in the Editor toolbar	Select Launch external application, complete the Filename box with the absolute path of the program's executable file, and complete the Command Line box with the necessary commands; for example, `%CURRENT%` passes the current document's name to the external program.
Run an ActiveScript file	Select Execute an ActiveScript file, and complete the Script File box with the absolute path of the script file.

5 Click Browse and select an image file, or enter the absolute path of the image for the toolbutton in the Button Image box.

This is not applicable when launching an external application.

6 In the Caption box, enter a text label to appear under the toolbutton when a user displays both toolbuttons and their captions.

This is not applicable when launching an external application.

7 In the Button Hint box, enter text for the toolbutton's tooltip.

8 Click OK.

9 In the Customize dialog box, click Close.

The toolbutton is added to the toolbar in the workspace.

To change the order of toolbuttons:

1 Select **Options > Customize**.

2 In the Customize dialog box, in the Visible Toolbars box, select a toolbar.

3 In the picture of the toolbar, drag each toolbutton that you want to move to its new position.

4 Click Close.

To remove a toolbutton:

1 Select **Options > Customize**.

2 In the Customize dialog box, in the Visible Toolbars box, select a toolbar.

3 In the picture of the toolbar, drag the toolbutton that you want to remove to a position outside of the toolbar.

4 Click Close.

To add a custom toolbar:

1 Select **Options > Customize**.

2 In the Customize dialog box, to the right of the Visible Toolbars box, click Add Toolbar.

3 In the Add Toolbar dialog box, enter a name for the toolbar and click OK.

4 (Optional) Add toolbuttons and separators to the new toolbar.

5 (Optional) To make the new toolbar visible in the workspace, select the check box next to it.

6 Click Close.

To delete a custom toolbar:

1 Select **Options > Customize**.

2 In the Customize dialog box, in the Visible Toolbars box, select a toolbar.

3 Click Delete Toolbar.

4 Click Yes to confirm.

The custom toolbar is deleted.

Getting the most from the online Help system

The online Help system provides you with several types of information:

- Pop-up Help for basic tag syntax (press F2)
- Comprehensive context-sensitive tag Help (press F1)
- Embedded Help in tag editors, Tag Chooser, and Expression Builder (press F1 or Ctrl+F1)
- An extensive collection of searchable references in the Help tab

Opening Help in tag editors and Tag Chooser

Help text in tag editors and in Tag Chooser provides context-sensitive syntax and usage information.

To view Help in a tag editor or Tag Chooser, do *one* of the following:

- Click Toggle Embedded Help. Help text displays in a pane at the bottom of the dialog box.
- Click Browse help in separate panel. Help text displays in a new browser window.
- Press F1. Help text displays on the Help tab in the Document window.

  For more information, see "About the Help tab" on page 29.

Editing Help in tag editors and Tag Chooser

Using the Visual Tools Markup Language (VTML), you can edit Help text in dialog boxes or add your own text. For details, see "Customizing the Development Environment" on page 175, and particularly "Adding tag Help" on page 186.

Accessing online Help

The online Help tab contains the printed product documentation in HTML format and additional online references. They are an extensible resource for learning about product features, technology extensions, and other development topics.

In the Resources window, click the Help tab to display the Help. You can access other options for viewing Help from the Help toolbar, located at the top of the Help tab.

Printing Help

If you are using Internet Explorer as the internal browser, you can print a Help topic by displaying the document on the Browse tab, then right-clicking the document and selecting Print.

The default internal browser does not support printing.

Bookmarking Help

You can quickly access your favorite Help documents by bookmarking them. You can assign bookmarks to documents in the Help, file system, or the Internet.

To bookmark a document in the Help Tree:

- Right-click the document title and select Add bookmark.

  If the Help document is open, it automatically displays in the Bookmarks list. If the document is not open, right-click again and select Refresh.

To bookmark an external document:

1 On the Help toolbar, click the down arrow next to the Bookmarks toolbutton and select Organize Bookmarks.

2 In the Help Bookmarks dialog box, click Add.

3 In the Add Bookmark dialog box, enter a name for the bookmark.

4 Click Browse and find the document to bookmark, or enter the appropriate URL or file path.

5 Click OK.

  The bookmark is added to the Help Bookmarks list.

6 Click Close.

To open a bookmarked Help document:

- On the Help tab toolbar, click the down arrow next to the Bookmarks toolbutton and select a bookmark from the list.

To maintain bookmarks:

On the Help tab toolbar, click the Down arrow key next to the Bookmarks toolbutton and select Organize Bookmarks to open the Help Bookmarks dialog box. you can then perform any of the following tasks:

- To add a bookmark, click Add, enter a name, select Browse, find the file or web page, click Open, and click OK.

  The bookmark appears in the Help Bookmarks dialog box.

- To remove a bookmark, click the bookmark and click Remove.

  The bookmark is removed from the list.

- To rename a bookmark, select the bookmark, click Rename, enter a new name, and press Enter.

Searching the online Help

You can access a book's online index in the Help tree, or you can search the entire set of Help references using basic to complex search criteria.

Using an online index

With the exception of language references, each book in the Help has an online index.

To search a book using an index:

1 Decide on the term or terms that you want to find in the index.

2 Do one of the following:

- On the Help tab, in the Help toolbar, click Help tree, and open the book that you want to search. Double-click the Index page.
- In the top right corner of a topic, click the **i** toolbutton to open the index for that topic's book (and make sure that you search the correct book).

The Index for the selected book appears with a link for each letter of the alphabet.

3 Click the link for the first letter of your search term, and scroll the list of terms as necessary to find the term. Repeat this step for every search term you need.

Using the full-text search engine

To search the Help for the first time on your computer, allow the program to **index** the files; that is, to generate a full-text search database from the files in the Help. This greatly improves the quality of search results.

After the initial indexing, each time you open the Help tab the program checks the Help for changes. If Help files have been added or deleted, the program automatically indexes the Help files again.

If you have problems with Help Search, re-index (regenerate the search database for) the contents of the Help folder. To do so, close the program, delete the Verity/Collections folder in your installation directory, open the program again, and run a search. Click Yes to allow Verity to re-index the Help files.

To run a full-text search of every Help reference:

1 On the Help toolbar, click Search.

2 If this is the first time that you are running a search on your computer, you are prompted to index the Help text. Click Yes.

3 In the Search Help References dialog box, in the Enter the word(s) to find box, enter a word or phrase.

For information on using search operators, click Search Tips.

4 To only search the titles in Help pages, select Search titles only (faster).

This is fast but not complete. If you select this and your search does not have the desired results, try searching again without selecting this option.

5 To limit the references searched, select the Search only in selected references option and select the check box next to one or more references.

6 Click Search.

The search results appear in the Help pane.

7 To open a document from the results list, double-click it.

The results list is saved until a new search is run.

To return to the search results:

- On the Help toolbar, click Search results.

To redo a recent search:

1 On the Help toolbar, click Search.

2 In the Search Help References dialog box, in the Enter the word(s) to find box, click the down arrow and select the search string that you previously used.

Using advanced search operators

You can use a variety of advanced search operators to make your searching more precise. For information on using advanced search operators, click Search on the Help toolbar and, in the Search Help References dialog box, click Search Tips.

To print this list of search tips, open the file \Help\VeritySearchTips.htm and use your browser's print command.

Extending the Help system

Like the product, the Help system is fully extensible. You can add documents to the Help system by dragging and dropping them into the Help folder, either from the Files tab or Windows Explorer.

Help document files are generally in HTML, but you can include text files. The Help browser will use the filename as the title. You can also create a link from within an HTML file to files with TXT, DTD, and PDF extensions if you use Internet Explorer as the internal browser. You can view web graphics files (GIF, JPG, PNG) in Help files.

This extensibility enables you to do the following:

- Write and install your own custom Help files
- Download documents from the web and add them to the Help system
- Edit existing Help topics
- Distribute Help files; this can be particularly helpful for a development team

Documents are not visible in the Help tree—only folders are. Therefore you must place files in an existing folder or a new Help folder.

Adding content to the Help tree

You can add files or whole references (folders) to the Help tree.

To create a new folder in the Help tree:

1 In a Files tab, navigate to the Help folder in your installation.

2 Right-click in the Files pane and select Create Folder.

A new folder with a highlighted name box appears.

3 Enter a name for the folder in the box, and press Enter.

4 If necessary, press F5 to refresh the Help display.

The new folder appears at the bottom of the Help tree.

To add documents to the Help tree, do one of the following:

- Write an HTML Help file and save it to a folder in the Help directory.
- Use standard Windows commands to paste a file into a folder in the Help directory.
- Display the document to add in a web browser, and then select the Save As command and save it to a folder in the Help directory.

Changing the order of items in the Help tree

If you add a folder to the Help directory, the folder appears at the bottom of the Help tree. For most users, this is all you need to do.

If, however, you want to control the order in which the Help references appear, you can edit the booktree.xml file in the Help root directory. The structure of the booktree file uses a basic tag set to configure how the Help displays on the book, chapter, and page levels.

The parser reads the booktree tags from top to bottom, so you can insert folder and file tags where you want the references to appear in the Help tree. You can edit this file to rearrange existing Help content as well as to add custom folders and files to a specific location in the Help tree. Alternatively, you can use the `path` attribute in the book and chapter tags to specify the folder's location on your computer.

To add a folder to a specific location in the Help tree:

1 In Windows Explorer, open the Help root directory, make a backup copy of the booktree.xml file, and copy the new folder into the Help root directory.

2 In HomeSite+ for Dreamweaver MX, open the new Help folder.

3 In the file list, double-click the booktree.xml file.

The booktree file opens in the Editor.

4 Add a book tag for the new folder and save the file.

Example: `<help_book path="folder name">`Custom Help Files

The new folder appears in the order that you set, but the files in the folder appear in alphabetical order. If you want to control the order of chapters and pages, add tags in the booktree file for these entries, and rearrange them.

Displaying text files in the Help tree

You can add text files to the Help tree. The Help browser default file type is HTML, so it looks for a `title` tag in each file. If it does not find one, or if the tag is empty, it displays the filename as the document's title in the Help tree.

To change an HTML file's title in the Help tree:

- Edit the HTML file to include its new name in the `<title>` tag.

To change a text file's title in the Help tree:

1 Make a backup copy of the booktree.xml file in the Help root folder of your installation directory.

2 Open the booktree.xml file.

3 Find the `help_page` tag for the text file, and add a `title` attribute for the text file; for example, `<help_page title="My Readme">.`

Adding media content

You have considerable flexibility in adding supporting files (such as graphics, animations, video, and sound) to Help documents. The following are two ways to do this:

- Create a file structure in the Help tree that conforms to the media file references in your documents. For example, you can copy an Images folder into your custom Help folder.
- In your custom documents' references to media content, supply a URL or accessible file location for remote files. For example, you might reference a remote site to display a chart of the NASDAQ, so the chart remains up-to-date. Generally, this approach is not optimal for adding static information, since external servers can go down and/or your computer can slow down, you use it with caution.

Note
If you are using the default internal browser or an older external browser, proper playback of media files is limited.

Chapter 4

Managing Files

This chapter describes how to perform file management tasks such as opening, closing, saving, and automatically backing up files. This chapter also describes how to work with Unicode and DBCS (Double-Byte Character Set) files.

The tasks described in this chapter do not replace the need for creating a project to organize the files in a website or web application. For instructions on how to use projects, see “Managing Projects” on page 121.

To manage remote files, see “Working with files on remote servers” on page 14.

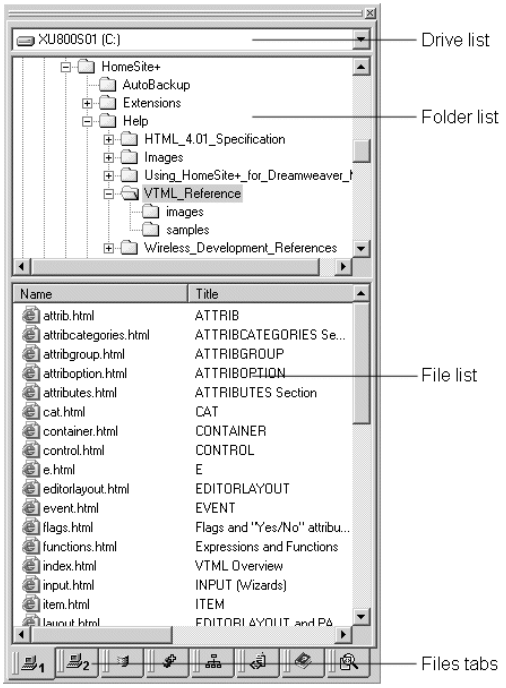
Contents

- About the Files tabs 42
- About file encoding 43
- Working with files 44

About the Files tabs

You can select the Files tabs in the Resources window to access servers, drives, directories, and files.

Having two Files tabs reduces the amount of file system navigation needed to work across multiple directories and drives. The following screenshot shows the main areas of the Files tabs.



Following are some tips on how the two Files tabs work together:

- Set the Drive list to My Computer to easily access local and network drives and Macromedia FTP & RDS.
- You can use the standard copy, move, and paste commands between tabs.
- If you undock one or both of the tabs, you can drag files and folders between them. To move them, drag them. To copy them, drag them while holding down the Ctrl key.
- When you select Filter or View from the right-click menu, the setting is only applied to the active tab; HomeSite+ for Dreamweaver MX maintains separate settings for each tab.
- Both Files tabs share the same Favorite Folders settings.
- If both File tabs point to the same location, HomeSite+ for Dreamweaver MX refreshes both tabs to display changes to files and folders in that location.

Several operations—such as opening and saving files, backing up before replacing text in files, displaying thumbnails for the images in a folder, browsing to files in tag editors, and creating an image map—refer to the current directory.

The Files tabs use the following rules to determine which directory is the current directory:

- When browsing to a file in a tag editor, the current directory is the folder that the current document is in.

  Options > Settings > General > Display current folder in file dialogs must be set.
- If only one Files tab is visible, the current directory is the one displayed in the visible File tab.
- If both Files tabs are visible, or neither Files tab is visible, then the current directory is the one displayed in the primary Files tab (*Files 1* tooltip).

You can move and dock tabs, for example to display or hide both Files tabs.

About file encoding

If you work with non-ANSI–encoded documents, you can open encoded files and save files with character encoding.

The following encoding formats are supported:

- ANSI (Current system code page)
- Unicode
- Unicode Big Endian
- UTF-8

Processing files from Unicode encoding formats involves codepage checking, detection of file encoding, and format conversions. Therefore, enabling non-ANSI file encoding slows document handling operations in the Editor. You can also work with ANSI files while working with Unicode files, but for optimal performance, only enable non-ANSI file encoding when you must open or save Unicode files.

By default, files are not handled as Unicode.

Note
You cannot transfer a Unicode file successfully using a Secure Sockets Layer (SSL) enabled FTP server. Disable the SSL feature in the Configure FTP Server dialog box before transferring these types of files.

To enable non-ANSI file encoding:

1 In the **Options > Settings > File Settings** pane, select Enable non-ANSI file encoding.

2 (Optional) Select the option to display encoding information on the Document tab in the Editor window.

3 Click Apply.

Working with files

You can use the following procedures to work with files, regardless of their location (local drive, network drive, or remote server).

Opening and saving files refer to the current directory. For information about how HomeSite+ for Dreamweaver MX determines the current directory with two Files tabs, see "About the Files tabs" on page 42.

Setting startup options

You have a number of choices in how files are folders are displayed when you start HomeSite+.

To set startup options:

1 In the **Options > Settings > Startup** pane, select the files and folders to open when you open HomeSite+ for Dreamweaver MX.

2 Under Startup, select what you want to open.

3 Under Startup folder, select the folder you want to display in the Files tab.

4 Click Apply.

Opening a file

Use the following procedures to open a file in the file list.

To open a standard file:

1 On the Files tab, use the drive drop-down list and the directory tree to navigate to the directory that contains the file.

2 Double-click the file in the file list.

Note
Read-only files are marked with a red dot in the file list. You can right-click a file and select Properties to set read and write access. This is not recommended, however, if you are using a source control application to manage read and write privileges.

To open an encoded file:

1 Verify that you have Unicode formatting enabled.

For more information, see "About file encoding" on page 43.

2 Open the file.

If you attempt to open a Unicode file without having selected the Enable non-ANSI file encoding option, the encoding format of the file cannot be detected or handled properly. The file is loaded as an ANSI string of the same type specified in your system language settings.

To open a recently used file:

1 Select **File > Recent Files**.

A drop-down list appears, listing the last ten files that were closed.

2 Select a file from the list.

3 (Optional) To clear the list of recently used files, select **File > Recent Files > Clear MRU List**.

4 (Optional) To remove files from the list, select **File >Recent Files > Remove Obsolete**.

Adding a link to an open file

To create a link to a file:

1 Open the file in which you want to add a link.

2 From the file list, drag a file or image to the Editor, to the desired cursor position on the page.

Note
Alternatively, you can right-click a file and select Insert as Link. The link is inserted in the current page at the cursor position.

Saving a file

In the Document tab at the bottom of the Editor, an **X** next to a filename indicates that there are unsaved changes in the file.

This section describes standard file saving behavior and the file saving behavior of files with encoding.

Saving a standard file

Use the Save commands from the File menu to save a file: Save, Save As, Save All, and Save as Template. You can specify a format for saved files in the **Options > Settings > File Settings** pane. The default format is PC, but UNIX and Macintosh formats are available. You can also set a default file extension in this pane.

When you add a link or image to a new file, you are prompted to save the file. This ensures that the relative path to each of these page elements resolves correctly.

Saving a file with encoding

If you work with non-ANSI–encoded documents, select character encoding from the Save As dialog box.

For this release, do not save a file as Unicode or Unicode Big Endian on an SSL-enabled FTP server, or else the files are not created and saved correctly.

Also, saving Unicode files might result in zero-byte files. This is a known problem that tends to occur when changing Unicode encoding selections.

To save a file with encoding enabled:

1 Select **File > Save As**.

2 In the Save As dialog box, in the Encoding drop-down list, select a format.

3 (Optional) Select Check the document character set.

This displays a warning message if the Save As encoding selection conflicts with the file's document character set statement (the `<meta>` charset statement in HTML), or with the encoding attributes in an XML processing instruction. You can cancel the Save As operation and reconcile the encoding formats.

4 Complete the other fields as you would for a standard file and click Save.

HomeSite+ for Dreamweaver MX saves the file with the encoding that you specified.

Backing up files

This section describes the Auto-Backup feature and how to use it.

Auto-Backup operates much like the backup systems in many desktop applications. It is a convenient way to save files while working, but more importantly, it is a safeguard against the loss of local and remote files caused by program or computer crashes, network disruptions, or power outages.

You can configure Auto-Backup to do any of the following tasks:

- Create a backup before the original file is saved.
- Create a backup of all open, modified files at a timed interval.
- Create a backup of files modified by search and replace operations.

In addition, you can back up files that are affected by a Replace operation in the **Search > Extended Replace** dialog box.

How Auto-Backup works

Following are the steps that HomeSite+ for Dreamweaver MX takes to back up files:

1 During installation, HomeSite+ for Dreamweaver MX creates a default Auto-Backup folder under the product root directory, and creates a control filenamed Auto-Backup.ini to track the files in the Auto-Backup folder.

You can change the backup location. If you do, HomeSite+ for Dreamweaver MX creates a new INI file in the new location.

2 While you are working on a file, Auto-Backup creates a backup file with the following name convention: *filename + an incremented 3 digit number + the file extension*; for example, myfile000.htm.

3 If the application closes abnormally, when you restart the application, Auto-Backup opens all of the open files that were saved by Timed Backup.

   If Timed Backup is not enabled and the application closes abnormally, you can open the last backup version of a lost file from the backup location.

4 When the original file is saved, or when the application closes normally, Auto-Backup deletes a Timed Backup file.

Configuring Auto-Backup

You can configure Auto-Backup to best meet your needs.

To set Auto-Backup options:

1 In the **Options > Settings > Editor > Auto-Backup** pane, enable or disable Auto-Backup.

2 Accept the default backup directory or select a different one.

3 (Optional) For Auto-Backup on Save, set the following options:

   – Use the backup directory for local and network files, or save the backup with the original file. Note that the backup directory is always used for remote files.

   – Set a time interval (in days) after which backup files are deleted.

4 For Timed Auto-Backup, set a time interval (in minutes) after which HomeSite+ for Dreamweaver MX saves all open modified files.

Using Auto-Backup

You can use Auto-Backup to manage backups or to recover a lost file.

To manage backups created by Auto-Backup:

1 Select **Options > Auto-Backup File Maintenance**.

2 In the Auto-Backup File Maintenance dialog box, manage backups created during Save and Extended Search and Replace operations.

3 Click Close.

To recover unsaved files after the application closes abnormally:

1 Open HomeSite+ for Dreamweaver MX again.

2 If you have set options for Timed Auto-Backup, the Timed Auto-Backup File Recovery dialog box automatically appears. Recover each lost file by selecting it and clicking OK.

3 Otherwise, select **Options > Auto-Backup File Maintenance** and, in the Auto-Backup File Maintenance dialog box, check the files that you must recover and select **File > Open Selected Files for Edit**.

The files open in the Editor.

Changing the file list display

You can arrange the file list to display exactly what you want.

To filter the file list:

1 Right-click in the file list, select Filter, and select one of the following options:

- **Web Documents** displays web documents only.
- **Web Images** displays web images only (JPEG, JPG, PNG, and GIF).

  You can edit the list of file extensions for Web Documents and Web Images in the **Options > Settings > File Settings** pane.
- **All Web Files** displays both web documents and web images.
- **All Files** displays every file in the selected directory.

2 To refresh the file list display, press F5.

To change the information listed for each file:

1 Right-click in the file list and select View.

2 From the View submenu, show or hide the document title, document size, modified date/time, and document type.

3 To refresh the file list display, press F5.

Dragging a file from Windows Explorer

You can drag a file or image from Windows Explorer into the current document in the Editor to create a link to the file.

If you hold down the Ctrl key while dragging the file into the Editor, the file opens in a new document. If the file is not a recognized file type, you are prompted to open it in the associated program for that file type.

Building a Favorite Folders list

You can build a list of favorite folders to quickly access their files.

To add a folder to the favorites list:

1 On a Files tab, in the directory tree, select a folder.

2 In the file list, right-click and select **Favorite Folders > Add Current Folder to Favorites**.

To view a favorite folder:

1 On a Files tab, right-click in the file list and select Favorite Folders.

2 From the Favorite Folders submenu, select the folder.

The folder opens in the directory tree and its files appear in the file list.

To organize your favorite folders:

1 On a Files tab, right-click in the file list and select **Favorite Folders > Organize Favorites**.

2 In the Favorite Folders dialog box, select a folder and click the up and down arrows to move its location in the favorite folders list.

3 Select a folder and click Remove.

4 When you are prompted to confirm the deletion, click Yes.

5 Click OK.

Downloading a web page

You can download any HTML page and open it in the Editor.

Note
These files cannot be saved back to the server.

To open a page from a website:

1 Select **File > Open from Web**.

2 Enter the URL for the page, or select from your Bookmarks or Favorites list.

3 If the site is accessed using a proxy server, click Proxy and enter the server name and port number.

4 (Optional) Set a time-out limit for the connection.

5 Click OK.

The file opens as an untitled document.

To copy web page content, do one of the following:

- Select a block of text on a web page, use the Windows copy commands, and paste the text into a document. Note that page formatting is not preserved.
- Use the browser's command to view the web page source code, then copy and paste the source code that you want into a document.

Note
Please honor any copyright and other restrictions on web document content.

Chapter 5

Writing Code and Web Content

HomeSite+ for Dreamweaver MX manages many different web development tasks, from writing basic HTML pages to designing, testing, and deploying complex, dynamic sites.

This chapter describes basic techniques for creating web content and application code.

Contents

- Inserting code 52
- Using inline tools to enter code 55
- Using code generating tools 58
- Adding document content 59
- Using keyboard shortcuts 61
- Saving a code block as a snippet 62
- Resources for website accessibility 64
- Tips for visually impaired users 65

Inserting code

HomeSite+ for Dreamweaver MX coding tools support a range of writing styles, from typing to point-and-click, and you can set the level of support you want: for repetitive tasks, you can use its productivity tools to enter repeated code blocks and required text; when coding tags that you are not familiar with, you can display pop-up tips and lists of language elements.

Note
Some operations, such as browsing to files in tag editors and creating an image map, refer to the current directory. For information about how HomeSite+ for Dreamweaver MX determines the current directory with two Files tabs, see "About the Files tabs" on page 42.

Inserting a tag from the QuickBar

The QuickBar is a customizable development toolbar. When you click a QuickBar toolbutton, it inserts code directly into the document or, for tags that require attributes, opens a tag editor.

In addition to toolbars for product tools and standard Windows commands, you can display toolbars for HTML, CFML, JSP, scripting, and ASP code.

Customizing a toolbar

You can customize the QuickBar and toolbars in these ways:.

To do this	Do this
Change tab order in QuickBar	Right-click QuickBar; select Organize QuickBar; use the arrows to set the tab order
Move QuickBar	Drag the double lines at the left end of the toolbar
Reset QuickBar configuration	In Customize dialog box, Toolbars tab, click Reset to Defaults. This does not remove toolbuttons that you have added.

You can customize a toolbar in these ways:

To do this	Do this
Display Customize dialog box	Select **Options > Customize**.
Change order of items in toolbar	In Customize dialog box, Visible Toolbars list, click a toolbar. Drag toolbuttons and separators.
Add toolbutton	In Customize dialog box, Visible Toolbars list, click a toolbar. Drag a toolbutton from the list to the toolbar.

To do this	Do this
Add custom toolbutton	Click Add Custom Button. Custom Toolbutton dialog box displays. Select an action; complete entries for action; click OK.
Add separator	In Customize dialog box, Visible Toolbars list, click a toolbar. Click Add Separator. Separator is added. Drag separator.
Remove toolbutton or separator	In Customize dialog box, Visible Toolbars list, click a toolbar. Drag item from toolbar to outside of toolbar.
Add custom toolbar	In Customize dialog box, click Add Toolbar; enter a name; click OK. Select new toolbar; add toolbuttons.
Delete custom toolbar	In Customize dialog box, select a toolbar, and click Delete Toolbar.

Selecting a tag from Tag Chooser

Tag Chooser provides access to the tag set in all supported languages. For details, see "About language support" on page 82.

To use Tag Chooser:

1 In the Editor, right-click and select Insert Tag.

The left pane in Tag Chooser displays the supported languages. The right pane displays the tags for the selected language.

2 Do either of the following:

- Click a language folder. Its elements display in the right pane.
- Expand a language folder. Its elements display in functional categories. Click a category. Its tags display in the right pane.

3 Click a tag in the right pane.

If you want to view syntax and usage information for the tag, click one of the Help icons below the right pane. The Help pane opens to display tag-specific Help.

4 To insert a tag, click Select for the highlighted tag or double-click it.

Tags that are inserted directly into a document are listed with brackets in the right pane, such as `<HTML></HTML>`. All other tags have individual editors that open when the tag is selected.

5 If a tag editor opens for the selected tag, complete the entries as needed and click OK to insert the tag.

You can open the Help pane in a tag editor for syntax and usage information.

You can resize Tag Chooser and keep it open while you work.

Completing a tag with a tag editor

You can use a tag editor to add a lot of content within an existing tag, for example in the body tag.

To complete the details for a tag:

1 In the Editor, position the cursor in the tag to complete.

2 Right-click, and select Edit Current Tag from the pop-up menu.

3 Complete the tag editor dialog box.

4 Click OK.

The tag editor closes, adding the attributes and values that you specified to the tag.

Using inline tools to enter code

You have a number of tools to help you insert new code and to edit existing code as you type. Each of these tools supports a distinct language element, such as tags, objects, and functions.

You can set the options for these inline tools in the Settings dialog box (F8).

Using Tag Insight

Tag Insight lets you insert tag names, attributes, and values as you type, after typing a start bracket (<) for a tag.

To enable and configure Tag Insight:

1 Open **Options > Settings > Editor > Tag Insight** and select the Enable tag insight and Enable tag insight tag List options.

2 Set how many seconds Tag Insight waits before displaying a list of options.

3 To add an item to the drop-down list of options that appears after typing < in the Editor: Click Add, enter the item, and click OK.

4 To delete an item from the drop-down list: Click the item in the list and click Delete. The item is immediately deleted from the list.

5 Click Apply.

To view a Tag Tip:

1 Position the cursor inside a tag and press F2 to see the attributes and values for the tag.

2 Press the Esc key to close the pop-up.

To insert a tag with Tag Insight:

1 In the Editor, enter a start bracket (<) to display a drop-down list of tags.

Press the Esc key to close a drop-down list.

2 Scroll down the list, select a tag, and press Enter to insert the tag.

3 Press the spacebar to display the drop-down list of attributes for the tag, then select an attribute and press Enter.

4 Press the spacebar again to display a drop-down list of known values for the attribute, then select a value and press Enter.

5 Repeat the last two steps until you have entered every element for the tag.

To edit a tag using Tag Insight, do any of the following:

- **To add an attribute** — Position the cursor to the left of a tag end bracket (>) and press the spacebar to display a drop-down list of attributes for the tag. Select an attribute and press Enter.

Press the Esc key to close a drop-down list.

- **To change an attribute** — Delete the attribute and add a new attribute.
- **To add a value** — Press the spacebar after the attribute to display a drop-down list of known values for the attribute. Select a value and press Enter.
- **To change a value** — Delete the value and add a new value.

Using Function Insight

Function Insight lets you insert function argument syntax as you type.

To enable and configure Function Insight:

1 In the **Options > Settings > Editor > Function Insight** pane, select the Enable function insight option.

2 Set how many seconds Function Insight waits before displaying a list of options.

3 To change the list of available functions, modify the Expression Builder definition file. The Expression Builder function library determines the entries in the function list. For more information, see "Customizing the Development Environment" on page 175.

4 Click Apply.

To insert arguments using Function Insight:

1 Enter a function name followed by a left parenthesis, (, to display a drop-down list of recognized arguments.

2 Select an argument from the list and press Enter to insert it.

Arguments must be separated by commas. If there is more than one possible argument for a function, a comma is automatically inserted after the argument.

3 To add another argument, press the spacebar after the previous argument and its comma. Select an argument from the list and press Enter to insert it.

4 Repeat the previous step as needed.

5 To complete the function call, delete the comma after the last argument, if applicable, and add a closing right parenthesis.

Using Tag Completion

When Tag Completion is turned on, the end tag is automatically inserted after you finish typing the start tag.

You can set options for this feature in the **Options > Settings > Editor > Tag Completion** pane.

The default tag completion set does not include every container tag necessary for coding in XHTML; for example, it does not include < />.

Using Auto Completion

Auto Completion completes a code block by inserting the appropriate code when you enter the opening code fragment (trigger string).

You can set options for this feature in the **Options > Settings > Editor > Auto Completion** pane.

A few strings are listed by default; such as, to insert a Comment end tag (-->) after you enter the start tag (<!--). You can also specify the cursor position after insertion; for example, before --> in the previous example of the code comment. You can insert any user-defined string, even blocks of text or code; however, you might want to use code templates for whole blocks of text and code snippets for whole blocks of code.

Using Code Templates

A code template provides a shortcut for entering static text blocks. As with Auto Completion, this feature is intended for user-defined strings.

To define a code template:

1 Open **Options > Settings > Editor > Code Templates**.

2 In the Code Templates pane, click Add.

3 In the Add Code Template dialog box, enter a keyword, a description of the code template, and a value to be inserted in place of the keyword. Click OK.

For example, you could enter *dt4* for the keyword, *HTML 4.0 Doctype* for the description, and for the value, `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">`.

If the value includes a tab, it is inserted as an ASCII #9 character, unless the **Options > Settings > Editor > Insert tabs as spaces** option is selected.

4 The code template displays in the list. Click Apply to save your changes.

To edit a code template:

1 Open **Options > Settings > Editor > Code Templates**.

2 In the Code Templates pane, click Edit.

3 In the Edit Code Template dialog box, change values as needed and click OK.

4 Click Apply to save your changes.

To insert code template text:

1 In the Editor, enter the keyword.

2 Press Ctrl+J to expand the text.

For example, the keyword *dt4* expands to `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">`.

Using the Extended and Special Characters palette

To enter special characters and Latin-1 extended characters, you either can enter the ASCII code or select from a special characters palette.

To display the special characters palette:

- Select **View > Extended and Special Characters**.
  The palette appears at the bottom of the window.

To change the character to its entity name:

- Select **Options > Settings > Editor > Automatically convert extended characters**.
  This converts every character higher than ASCII 127 to its entity name so that it will be properly rendered in browsers.

Using code generating tools

Following are some tools that you can use to speed up your web development:

- Templates (**File > New**) are useful for creating many pages with the same standard content, for example the same header and footer or the same application logic. To use your current document as a template for future documents, select **File > Save As Template**. Then, to create new files from this template, select **File > New** and, in the New Document dialog box, select the template from the Custom tab.
- Wizards (**File > New**) generate HTML, DHTML, CFML, Java, and JSP code. The Wizards for Deployment Scripts and Initial Configuration are in the Custom tab.
- Expression Builder provides an expandable tree of all supported expression elements, including functions, constants, operators, and variables. It fully supports ColdFusion 5. Expression Builder lets you insert expressions into a working area, where you can edit or expand your expression before inserting it into your code.
- The JavaScript Tree provides quick access to an expandable tree of JavaScript elements, allowing quick insertion of a JavaScript element. There is also a JavaScript Wizard available to generate and insert several JavaScript functions.
  Both the JavaScript Tree and JavaScript Wizard are in the list of toolbuttons that can be added to your workspace. For instructions on adding a toolbutton to a toolbar, see "Customizing toolbars" on page 32.
- TopStyle Lite enables you to design, preview, and reference CSS files.
- If Macromedia Dreamweaver MX is installed on your computer, you can use it for visual page editing. It lets you to prototype page layouts, create HTML tables and forms, set font and image formatting, generate DHTML elements, and more.
  To open your page in Dreamweaver MX, select **View > Open in Macromedia Dreamweaver/UltraDev**.
- The Image Map Editor lets you to add more than one link to the same web image (GIF, JPG). To get started, select **Tools > New Image Map**.

Adding document content

The default template contains just the code required for a valid HTML document. This default template is stored as \Wizards\HTML\Default Template.htm below your root directory. You can edit this file and save it to change its content. Or you can save a document as a template by choosing **File > Save as Template**.

To open a new document, do either of the following:

- Press Ctrl+N to open the default document template.
- Select **File > New** to select a different template or a wizard.

You can add a link from an HTML, CFML, JSP or other HTML embedded document to a URL.

To add a link, do either of the following:

- Drag a file from a Files tab to insert a link into the current HTML document, in an image tag, or in an audio or video clip.
- Click the Anchor toolbutton on the Common QuickBar tab to define internal and external links.

For other supported languages, such as WML and SMIL, use the appropriate link syntax.

To insert a file into a document:

1 Select **File > Insert File**.

2 Open the drop-down Files of type list to filter file extensions.

3 Locate the file and click OK to insert the file contents in the current document.

To convert a text file to HTML:

1 Select **File > Convert Text File** and locate a local or remote file with a .txt extension.

2 Click OK to open the file in the Editor, surrounded by basic HTML tags.

3 Add formatting or link to a style sheet.

To copy text from a browser:

1 To strip the HTML formatting code from copied text, select the Treat HTML dropped from external application as plain text option in the **Options > Settings > Editor** pane.

2 In the browser, select the text that you want to copy.

3 Use Windows copy and paste commands to enter the text block in the Editor.

To insert Microsoft Office content, do either of the following:

- To convert Microsoft Office content, such as lists, tables, and worksheet cells, to plain text, select the content and copy it to the Editor.

- To insert content with the equivalent HTML formatting, open Dreamweaver MX, copy the content into Dreamweaver MX, and save.

  You can edit the content visually in Dreamweaver MX, or close it and return to working directly in the code.

To preview a page in a browser, do one of the following:

- To view the current document in the internal browser, press F12.

  For more information, see "About the Browse tab" on page 29.
- To view the current document in the default external browser, press F11.
- To switch to a different external browser, click the View External Browser List toolbutton on the View toolbar, and select a browser from the list.

To add content in a visual WYSIWYG editor:

1 Select **View > Open in Macromedia Dreamweaver/UltraDev**.

- If this command does not appear in the View menu, select **Options > Settings**. In the Settings dialog box, select **Dreamweaver/UltraDev > Enable Dreamweaver/UltraDev integration**.
- If the menu command appears but does not work, install Macromedia Dreamweaver MX.

2 Add content in Dreamweaver MX.

For more information, see the Dreamweaver MX tutorials and online Help.

3 Save your work and close Dreamweaver MX.

The page displays in HomeSite+ for Dreamweaver MX again, with the changes you made in Dreamweaver MX.

Using keyboard shortcuts

Keyboard shortcuts are active for many file management, editing, and debugging commands.

To see the current keyboard assignments:

1 Select **Options > Customize**.

2 Click the **Keyboard Shortcuts** tab.

To print the list of shortcut keys:

1 In the **Options > Customize > Keyboard Shortcuts** tab, right-click the list and select Browse.

2 Use the browser's print command to print the document.

To assign a key combination:

1 In the **Options > Customize > Keyboard Shortcuts** tab, select a command from the list.

2 Click in the text box beside the Apply toolbutton and press the key combination you want to use.

If the key combination is already in use, the command that is currently assigned to the key combination appears below the text box. If you click Apply, you can no longer use the key combination for that command.

3 Click Apply.

The list automatically refreshes and displays the change.

Saving a code block as a snippet

A code snippet is a block of code or content that you store for reuse. You can comment your snippets just as you would any code block.

To create a code snippet:

1 Before adding a snippet, you must create a folder. To do so, click the Snippets tab in the Resources window, right-click in the Snippets panel, and select Create Folder.

You can create additional folders as needed. The default location for snippets is the \UserData\Snippets folder below your root directory.

2 Enter a name for the folder.

3 Right-click in the Snippets pane and select Add Snippet to open the Snippet dialog box.

4 Enter a name for the snippet in the Description box. Note that snippet names cannot contain characters that are illegal in filenames, such as slashes, special characters, or double quotes.

5 In the Start Text window, enter or paste an opening code block. You can set a default spacing between the blocks by pressing the Enter key at the end of the start text and at the beginning of the end text.

6 In the End Text window, enter a closing code block.

7 Click OK.

Because snippets can be created as start and end blocks, you can use them to surround other tags and content. This is handy for inserting special formatting, navigation elements, and script blocks. Simply highlight the content you want to surround, then insert the snippet.

To insert a code snippet, do either of the following:

- Double-click an entry in the Snippets panel.
- Right-click and select Insert into document from the popup menu.

To edit or delete a code snippet:

- Right-click the snippet and choose the appropriate command (Edit or Delete) from the snippets shortcut menus.

To manage code snippets and snippet folders:

- Use the snippets shortcut (right-click) menus.

Sharing snippets

You can share code snippets with other developers by giving them access to a shared folder on your computer or on another network computer.

To create a shared snippets folder:

1 Open **Options > Settings > Locations**.

2 In the Shared Snippets box, click Browse and navigate to the shared folder location and click OK.

3 In the Snippets panel, right-click, and select Create Shared Folder.

The folder toolbutton changes color to show that it is shared.

Anyone with access to the shared folder can now add, edit, and delete snippets.

Assigning a shortcut key to a snippet

Open the **Options > Customize** dialog box and select the Snippet Shortcuts tab to enable a keyboard shortcuts for a snippet. Select a snippet and press keys to enter the combination in the box at the bottom.

If a combination is in use, the current command appears below the shortcut box. To overwrite a combination, click Apply.

Resources for website accessibility

The W3C Web Accessibility Initiative (WAI) at http://www.w3.org/WAI/ provides a great deal of practical information about designing applications and interfaces for the broadest range of users. Macromedia supports user accessibility in its product interfaces and is striving to fully implement the WAI recommendations.

Basic steps, such as providing a text alternative for images, graphics, and animations, and using descriptive text for page elements and navigation, can make a world of difference to a visually impaired user's experience of your site. Validation tools are available on the site to check compliance with established standards.

Standards may change more quickly than the product, but you can script the Visual Tools Object Model (VTOM) to extend or manipulate its functionality. For details, please see "Customizing the Development Environment" on page 175 and "Scripting the Visual Tools Object Model" on page 201.

Tips for visually impaired users

Macromedia is committed to enabling a high level of accessibility in our products. HomeSite+ for Dreamweaver MX includes support for customizable keyboard shortcuts and browser-based HTML online documentation, including Alt text for all images.

This section offers suggestions for ways to work productively in HomeSite+ for Dreamweaver MX.

Assigning keyboard shortcuts

Keyboard shortcuts are enabled for many file management, editing, and debugging commands. The shortcut list also contains many commands and language elements that are unassigned.

To open the shortcut key list:

1 Select **Options > Customize** (Shift+F8).

The Customize dialog box appears.

2 Press Ctrl+Tab to switch to the Keyboard Shortcut tab.

3 Press Tab to highlight the first key assignment in the list.

4 Use the arrow keys to scroll the list

5 To open the list in the external browser, press Shift+F10 to open the context menu, then select the Browse command.

You can change any of the current key combinations and you can also enter new key combinations to unassigned items in the list.

To assign a key combination:

1 In the Customize dialog box, on the Keyboard Shortcut tab, select a command from the list.

2 Press Tab to place the cursor in the key assignment box and then press the key combination you want to assign.

3 Select Apply to save the assignment.

If the key combination is already in use, a message box displays. You can then choose to overwrite the current assignment or cancel the dialog and press Shift+Tab to place the cursor in the key assignment box again.

The list automatically refreshes when changes are made.

Creating additional key combinations

The Snippets Shortcuts and Scripts Shortcuts tabs in the Customize dialog box also contain a key assignment box that you can use to enter key combinations for those code components.

Using shortcut keys for common tasks

Here are a few handy shortcuts:

- Press F11 opens the current document in the default external browser, then use Alt+Tab to move between the applications.
- F9 toggles the Resources window display.
- Shift+F9 changes the focus between the Document Window and the Resources Window.
- In the Document Window, use Ctrl+Tab to move between open documents.
- In the Resources Window, use Ctrl+Tab to move between Resources tabs.
- In the Help tab, use the arrow keys and the plus and minus keys to navigate the Help tree and press Enter to open a Help topic.
- F12 toggles the Edit and Browse modes.
- Shift+F12 toggles the Edit and Help modes.

Working with user interface elements

The following list describes some ways to work productively in the user interface.

- You can use Windows Explorer instead of the Files tabs for working with the file system, network drives, and remote servers.
- Tag Chooser (Ctrl+E) gives you access to the tag sets of all supported languages. Selecting a tag from Tag Chooser opens the tag editor or inserts the tag directly if the tag does not have attributes. This is an alternative to using the tag toolbuttons on the QuickBar.
- The Results window opens to display output from the Search, Code Validation, Link Tester, and Project Deployment operations as well as showing Image Thumbnails. You can hide or display the Results window by pressing Ctrl+Shift+L. The Keyboard Shortcuts list contains key combinations to focus the view on the various Results tabs.
- Enable the **Options > Settings > General > Use standard file dialogs** option. It is off by default. The standard dialogs are accessible by screen readers.

Chapter 6

Editing Pages

HomeSite+ for Dreamweaver MX has a wealth of tools for updating code and content. This chapter explains how you can use these tools to your best advantage.

Contents

- Setting editor options 68
- Selecting a code or text block 69
- Saving text to the multiple-entry Clipboard 69
- Collapsing text 70
- Editing a referenced file 71
- Using tag editors 73
- Navigating the structure of a document 74
- Editing code in the Tag Inspector 76
- Formatting pages with Cascading Style Sheets 78

Setting editor options

HomeSite+ for Dreamweaver MX features a flexible editing environment that gives you control over all aspects of web maintenance.

Using the Editor toolbar

The column between the Document window and the Resources window contains the Editor toolbar. This toolbar gives you quick access to formatting, document management, and coding options.

Settings editor options

You can configure the following Editor display and coding tool options in the **Options > Settings > Editor** pane and its sub-panes:

- Default font, color, and text formatting
- Collapsed text lets you close a block of selected text while editing other parts of a document.
- Auto Completion saves typing by completing the string when a trigger string is entered.
- Tag Insight gives you pop-up access to tags, attributes, and values as you type. After settings options for this feature, you can toggle it on the Edit toolbar.
- Function Insight supports typing of function syntax.
- Tag Completion inserts the end tag when you type the start tag. You can edit tag syntax and add tags.
- Color Coding gives you a customizable set of colors for code, based on the file extension of the document.
- Code Templates insert a text block when enter a keyword and press the shortcut keys. Templates can be edited and you can your own.
- Auto-Backup lets you create a backup file when saving and at timed intervals.

Selecting a code or text block

You can use the mouse and standard Windows keyboard commands to highlight sections of a document, or use the following shortcuts:

- To select a tag, press Ctrl and double-click.
- To select an entire tag block, press Shift+Ctrl and double-click in either the start or end tag.
- To select a range of code and text, click at the start of the selection, then press Shift and click at the end of the selection.

Saving text to the multiple-entry Clipboard

The multiple-entry clipboard lets you store multiple text blocks in the clipboard and selectively paste items from it. You can use the Edit toolbar to complete these tasks, and you can set the maximum number of text blocks that you can store.

Using the Clipboard

To use the multiple-entry clipboard, access the following toolbuttons on the Edit toolbar:

- **Show Clipboard** Displays a drop-down window of copied items. You can position the mouse over an item to display a tooltip of the item's contents.
  Click an item to paste it into the current document at the current cursor position.
- **Paste All** Pastes all the current items into the current document at the current position. Items are pasted in the same order in which they were copied.
- **Clear Clipboard** Deletes all copied items from the clipboard.

These toolbuttons are not available until you have copied some code or text.

Setting the Clipboard entry limit

By default, the clipboard window stores a maximum of 36 entries. When the maximum number of clipboard entries is reached, the next copy operation deletes the oldest clipboard entry and adds the new copied text to the bottom of the clipboard entry window.

To change the maximum number of entries:

1 In the **Options > Settings > Editor** pane, in the Maximum clipboard entries box, select a new number.

2 Click Apply.

Collapsing text

When editing long documents or complex applications, you can hide text and code blocks so that you can focus on just a portion of the content. A customizable marker displays the first few characters of the collapsed selection.

You can drag collapsed text in the same way as text blocks. Also, you can nest collapsed text blocks. When you run a search operation or check spelling, and a match or misspelling is found in the collapsed section, the collapsed text expands.

To set collapsing text options:

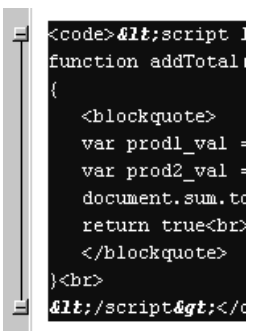
1 In the **Options > Settings > Editor > Collapsed Text** pane, do any of the following:

- Customize the appearance of the collapsed text marker, including the font and size of the text on the marker, and the foreground and background colors for the marker.
- In the Length (characters) box, specify the width of the marker—that is, how many monospaced characters of the collapsed text can fit on the marker.
- Enable a tooltip to display the text when you position the mouse over a collapsed text marker, and set the maximum number of lines to display.
- Enable the automatic selection of a collapsed text section when it is expanded. This lets you preview and re-collapse the section quickly.
- To keep text collapsed when saving and opening a file, select Preserve collapsed text on file open/save. Also, specify the number of days to keep the text collapsed (enter 0 for no limit).

2 Click Apply.

To collapse text, do any of the following:

- Select text, then click one of the collapse buttons (–) in the document gutter:



- To collapse the entire contents of a tag, right-click a tag and select Collapse Tag.
- To collapse a specific tag throughout a document, for example to collapse every `script` tag, right-click one of the tags and select Collapse All Identical Tags. This also collapses the tags that are nested inside the collapsed tags.
- To collapse text based on the current Tag Inspector outline, display the Tag Inspector resource tab and, in the top toolbar, click the blue plus sign <+>.

To expand collapsed text:

- To expand a single block of collapsed text, double-click a collapsed text marker.
- To expand all collapsed text, right-click anywhere in the document and select Expand All Collapsed Sections.

Editing a referenced file

You can edit image files and included files from HomeSite+ for Dreamweaver MX.

Editing an image file

If Macromedia Fireworks MX is installed on your computer, you can open your GIF, JPG, and PNG image files for editing in Fireworks MX. You can also view a thumbnail for an image, to preview it.

To display thumbnails for the images in a folder:

1 In one of the Files tabs, select the directory with the images.

2 Press Shift+Ctrl+F4.

3 In the Results window, right-click on the Thumbnails tab and select Reset to current folder.

For information about how the current folder is determined between the two files tabs, see "About the Files tabs" on page 42.

The Results window displays thumbnails for the images in the current folder.

To edit an image:

1 Select the image to edit, by doing one of the following:

- Right-click an image file in the Project tab or one of the Files tabs, and then select Edit in Macromedia Fireworks.
- In the Results window, on the Thumbnails tab, right-click a thumbnail for an image and select Edit in Macromedia Fireworks.
- In the Editor, right-click an `<img>` tag and select Edit in Macromedia Fireworks. The image tag must be complete and the file path and filename for the `src` attribute must refer to an actual file.

2 If the image you selected is in PNG format, skip this step. Otherwise, in the dialog box that appears, specify whether to open the source PNG file for the image:

- If you *never* create your images in PNG format and export them to GIF or JPG format, then select Never Use Source PNG in the drop-down list and click No.

  The image that you selected opens in Fireworks MX and, when you edit an image in the future, the image opens in Fireworks MX without displaying this dialog box.

- If you *always* maintain your images in PNG format and export them to GIF or JPG format, then select Always Use Source PNG in the drop-down list and click Yes. Browse to the source PNG file and click Open.

  The image you selected opens in Fireworks MX and, when you edit an image in the future, you can immediately browse to the source PNG file without seeing this dialog box again.
- Otherwise, select Ask When Launching in the drop-down list. Click No if you want to open the selected image for editing, or click Yes to browse for and open the source PNG file.

Fireworks MX opens.

3 (Optional) Edit the image.

4 Save your changes and close Fireworks MX.

Editing an included file

You can edit included files in certain tags. However, the referenced file must have an absolute or relative path. You cannot open an included file from a logical file reference, that is, a virtual mapping on the server.

To open an included file for editing:

1 In the Editor, right-click in one of these tags:

- `<cfinclude>`
- `<cfmodule>`
- `<script>`
- `<jsp:include>`
- `<%@ include>`
- `<jsp:forward>`
- `<vtinclude>`
- `<wizinclude>`
- `<!--#include>`

2 Select Edit Include file.

The file, `src`, template, or page that is referenced in the `include` tag opens.

3 Edit the file, save your changes, and close the file.

Using tag editors

A **tag editor** is a tag-specific dialog box that lets you enter information for a specific tag, and then inserts the tag with the attributes and values that you specified. Tag editors fully support ColdFusion Server. Each tag editor has its own online Help for tag syntax and usage information. You can also customize a tag editor; for more information, see "Working with tag definitions" on page 109.

About VTML tag editors

You can choose from two different tag editors for three tags: Anchor <a>, Body <body>, and Image <img>. The standard Image tag editor calculates the height and width of an image and provides an image preview, unlike the VTML Image tag editor. The standard Body tag editor has a preview for the colors of the page background, text, and each type of link, unlike the VTML Body tag editor.

Otherwise, the tag editors that are written in Visual Tools Markup Language (VTML) are the same, except the VTML tag editors support multiple languages, browsers, event code, and coding preferences. The VTML Anchor tag editor supports multiple protocols and accessibility options, unlike the standard Anchor tag editor.

For more information about VTML, see "Customizing the Development Environment" on page 175.

To use a VTML tag editor instead of a standard tag editor:

1 On the **Options > Settings > Markup Languages > HTML/XHTML** pane, select the Anchor - <A>, Body - <BODY>, or Image - <IMG> options as desired.

2 Click Apply.

Editing a tag with a tag editor

You can open a tag editor and access Help for completing the tag editor dialog box.

Note
If the tag editor is labeled with "NOTE: This is a write-only Tag Editor," then the tag editor is only intended for inserting a tag, not for editing the content of the tag.

To open the editor for a tag, do one of the following:

- Right-click in a tag and select Edit Tag.
- Position the cursor in a tag and press Ctrl+F4.
- Position the cursor in a tag and select **Tags > Edit Current Tag**.

To view syntax and usage Help for the tag:

- Press F1.

Tag-specific Help appears at the bottom of the tag editor dialog box.

Navigating the structure of a document

You can inspect and navigate the structure of a document in the Tag Tree. The Tag Tree is in the Resources window, in the top pane of the Tag Inspector tab. You can configure the Tag Tree to display only the tags you want.

Using the Tag Tree and Tag Inspector together, you can edit the code in a document without having to work directly in the Editor. For information about Tag Inspector, see "Editing code in the Tag Inspector" on page 76.

To use the Tag Tree:

1 On the Tag Inspector resource tab, in the top pane, select an outline profile from the drop-down list.

For more information, see "About outline profiles" on page 74.

2 Click a tag in the tree.

In the Editor, the cursor moves to that tag in the current document.

3 In the Editor, press Shift+Ctrl and double-click the tag.

The entire tag block is selected and, if the selected tag has attributes, they appear with their values on the Tag Inspector resource tab.

4 To update the Tag Tree with your edits in the document, in the Tag Tree toolbar, click Refresh.

5 To display the contents of the selected tag in a single node on the Tag Tree, in the Tag Tree toolbar, click Collapse document based on outline.

About outline profiles

Outline profiles let you restrict the Tag Tree display to specific tag sets. This is useful when you must work with mixed language elements and language versions in your documents. The default set of outline profiles provides many options for filtering the Tag Tree display, but you can also customize profiles.

Outline profiles are written in the Visual Tools Markup Language (VTML), and they fully support ColdFusion Server. They are stored in the \Extensions\Outline Profiles folder. You can open a profile in this folder to examine its syntax, to learn how to create or customize an outline profile.

To create or customize an outline profile:

1 In the Editor, open an outline profile in the \Extensions\Outline Profiles folder and examine its syntax.

The VTML markup identifies the tags that are included in the outline profile.

2 If you are editing an outline profile, make a backup copy of the VTML (.vtm) file and then edit the original VTML file.

3 If you are creating a new outline profile, save the file with a new name and then edit the new file.

Alternatively, you can create or edit an outline profile in the Outline Profile Editor dialog box. To open this dialog box, display the Tag Inspector resource tab and, in the toolbar in the top pane, click Configure Outline Profiles.

Setting the Tag Tree display

You can edit the list of outline profiles in the Tag Tree, and import an outline profile from a VTML (.vtm) file or a Document Type Definition (DTD) file.

To modify the list of outline profiles:

1 On the Tag Inspector resource tab, in the top pane, in the Tag Tree toolbar, click Configure Outline Profiles.

2 In the Outline Profiles dialog box, in the Outline Profiles box, select a profile and click Add or Remove, as desired.

3 Repeat the previous step until you are satisfied with the list of outline profiles.

4 In the Recognized Tags frame, do any of the following:

- Select a tag in the list and click Remove.
- Click Add Tag and enter the name of the tag to add to the list.

5 Repeat the previous step until you are satisfied with the list of recognized tags for the selected outline profile.

6 To customize a tag, select a tag in the list and complete the Tag Settings frame:

- **Caption** Select a caption for the tag, to display in the Tag Tree.
- **Icon** Enter or select an image file for the tag, to display in the Tag Tree.
- **Unenclosed tag warning** Select this to display a warning icon next to the tag in the Tag Tree if it does not have a closing tag `</..>`.
- **Warning icon** Enter or select an image file for the warning icon.

7 Click Done. (Note that there is no Cancel button.)

The outline profiles are updated for the Tag Tree.

To import an outline profile:

1 On the Tag Inspector resource tab, in the top pane, in the Tag Tree toolbar, click Configure Outline Profiles.

2 In the Outline Profiles dialog box, click Import.

3 In the Outline Profile Import Wizard dialog box, select one of the following:

- **Import an existing Outline Profile (VTM)** Imports a VTM outline profile.
- **Import from a DTD file** Creates a VTM outline profile from a DTD file.

4 Click Browse and select a VTML or DTD file.

5 Click Next.

The Outline Profile Import Wizard reports the status of the import process.

6 Click Finish.

If you imported a DTD file, HomeSite+ for Dreamweaver MX stores its information in a VTML file in the /Extensions/Outline Profiles folder.

Otherwise, the selected VTML file is copied to /Extensions/Outline Profiles.

7 In the Outline Profiles dialog box, click Done. (There is no Cancel button.)

To use the Tag Tree with imported DTD information:

1 Open a document that adheres to a DTD that you imported.

If you imported multiple DTDs, HomeSite+ for Dreamweaver MX automatically accesses the correct VTML information file.

2 On the Tag Inspector resource tab, in the top pane, select DTD Elements and Entities from the drop-down list.

3 Select entities and elements in the Tag Tree.

Whatever you select in the Tag Tree is highlighted in the document.

Editing code in the Tag Inspector

Tag Inspector lets you edit code in a property sheet user interface that is similar to the property sheets in Visual Basic and Delphi. Tag Inspector fully supports ColdFusion Server.

You can also edit tag definitions from Tag Inspector; for example, to change the valid attributes for a tag. For details, see "Working with tag definitions" on page 109.

Setting the Tag Inspector display

To control the order in which Tag Inspector displays attributes and scripting events, select from the following toolbuttons in the Tag Inspector toolbar:

- **Version Specific** Sorts by language and browser version.
- **Categorized** Sorts by type and browser version.
- **Alphabetically Z-to-A** Sorts by name in descending alphabetical order.
- **Alphabetically A-to-Z** Sorts by name in ascending alphabetical order.

To resolve cross-browser and language issues, select Version Specific or Categorized.

To use Tag Inspector:

1 Click a tag in the Tag Tree or in the Editor.

The Tag Inspector tab displays a list of the attributes and values for the tag.

2 Click an attribute name, and enter or select a value.

3 Click outside the field for the attribute.

The new attribute or updated value appears in your code in the Editor.

Creating and editing an event handler script block

You can add and modify script blocks for events such as `onClick` from Tag Inspector. If the selected tag supports an event as an attribute, you can edit the event in its attribute field. Otherwise, you can edit the event directly under the Events list.

To add an event handler script block:

1 On the Tag Inspector resources tab, in the Tag Tree pane, select a tag.

A list of the tag's supported attributes and events, with their values, appears in the Tag Inspector pane.

2 Click the event in the Attributes list or in the Events list.

3 In the drop-down list, select a script language for the event handler; for example, select Create JavaScript Event.

4 In the Specify Event Handler Name dialog box, select the default entry or enter a name, then click OK.

The event handler is added as an attribute to the tag, and a script block is created in the `head` section of the document.

5 Complete the script block for the handler.

6 Select **File > Save**.

To find an event handler function, do one of the following:

- On the Tag Inspector tab, under Attributes or Events, double-click the event.
- From a tag with an attribute for the event, right-click the event attribute; for example, right-click `onClick="_onclick()"` in `<table width="640" border="0" onClick="_onclick()">`. Then select Navigate to `<function name>` Function.

The cursor moves to the selected event handler function.

Formatting pages with Cascading Style Sheets

This section describes Cascading Style Sheets (CSS), and provides information for using the integrated TopStyle Lite Style Editor and for coding styles by hand.

About Cascading Style Sheets (CSS)

CSS (also known as **styles**) lets you specify a format for a page element, such as a paragraph or table, and then specify where to apply this format.

You can create a style block (also called **embedded styles**) in a single HTML page to apply the formats to a single block of HTML code; or you can create a separate CSS file and apply its formats across one or more pages in a website.

At whatever level you specify styles, the styles "cascade" down to apply to the lower levels as well. For example, a style block in the `<head>` section applies to the entire HTML document, and a style block in a `<table>` tag applies to the entire table. When there is more than one style block, the "local" style rules; in the preceding example, the table style overrides the head style in the table.

About classes

A class is just like a style, except that it is not automatically applied to a page element, but only to a page element whose tag contains the attribute `class="<class name>"`; for example, `<p class="note">`.

This means that you can define an unlimited number of styles for a single page element. For example, you can define classes for a paragraph tag to apply different formatting to introductory text, important notes, and contact information.

Benefits of using CSS

Styles provide the following benefits:

- Lets you quickly preview text, color, and layout designs. This is a great help when you are prototyping a new or redesigned website. Imagine the alternative: changing every tag with formatting—for example, `<font="Arial" size="10" color="black">` and `<p align="center">`—in every page!
- Lets you make rapid and dramatic changes to all HTML elements in a website; you can reference a style sheet by adding a single attribute to the `<head>` tag and, by doing so, immediately change the look of the entire page.
- Ensures a consistent look and feel throughout the website, even with multiple contributors. This makes it easier for visitors to find what they need.
- Reduces the amount of code required to format a page, so a browser can download, process, and display pages more quickly.
- Lets you manipulate styles and other document objects, when used in conjunction with scripts and Dynamic HTML (DHTML).
- Lets you focus on developing content, and solving any organizational or navigation issues in your site, instead of maintaining formatting code.

Note
Browser inconsistencies currently limit the full implementation of styles. To develop a cross-browser design strategy, you must research these inconsistencies, and select the browsers and browser versions that you want to support.

Managing a website or application with CSS

CSS separates the tasks of *formatting* and *developing* content. Therefore, the optimal time to plan and implement styles is in the design phase of a web project. You can add styles later, but this requires that you delete the formatting code within tags; otherwise, the styles have no effect.

As your site or application evolves, you can update, add, and delete styles globally. This gives you more time to test and compare styles, to achieve your design goals.

Style syntax

If you edit styles manually, you must adhere to their syntax requirements. The following are the basic coding requirements for styles:

- Properties and their values must be inside curly braces ({ }).
- Each property/value pair must be separated by a colon.

  Putting a space on both sides of the colon is a convention for better readability, but it is not required.
- Each property/value pair must end with a semicolon.

These requirements apply to both embedded styles and stylesheets. To use a stylesheet, create a CSS file and then, right before the `</head>` tag, add the following:

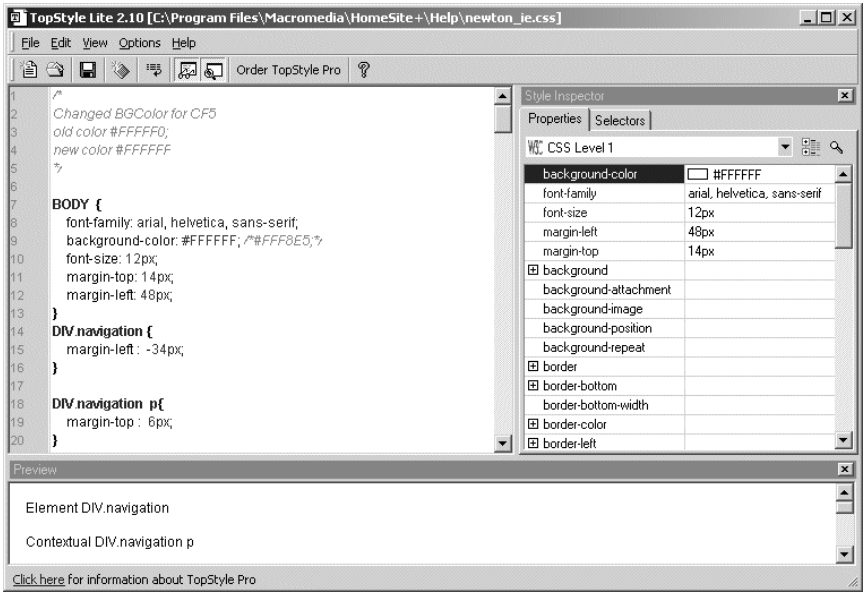
```
<link rel="stylesheet" type="text/css" href="[path to stylesheet]">
```

Example: `<link rel="stylesheet" type="text/css" href="/css/news.css">`

About the integrated style editor

TopStyle Lite provides an integrated user interface for designing and deploying Cascading Style Sheets (CSS). TopStyle lets you define, preview, and apply styles to web content. It has color-coding like HomeSite+ for Dreamweaver MX, so you can identify the various style elements.

The following figure shows the main components of the TopStyle user interface:



The TopStyle online Help includes a tutorial, a CSS reference, procedures for using its features, and information for upgrading to TopStyle Pro.

Chapter 7

Using Web Development Languages

This chapter describes language support in HomeSite+ for Dreamweaver MX and provides specific information for setting language-specific options and color coding schemes, coding in XHTML, validating against different language specifications., and working with tag definitions.

For information about building SQL statements, see “Using SQL Builder for database queries” on page 116.

Contents

- About language support 82
- Setting options for markup languages..... 83
- Coding in XHTML 85
- Using regular expressions..... 91
- Using color coding schemes..... 96
- Formatting code with CodeSweepers..... 98
- Validating code 104
- Working with tag definitions 109

About language support

This section describes what languages are supported, how to get around limitations, and provides an overview of how HomeSite+ for Dreamweaver MX detects and handles a language.

Supported languages

HomeSite+ for Dreamweaver MX has significant support for current standards in several web development languages. For example, the following languages all have their own tag definitions in HomeSite+ for Dreamweaver MX.

- Hypertext Markup Language (HTML)
- Extensible Hypertext Markup Language (XHTML)
- ColdFusion Markup Language (CFML)
- Visual Tools Markup Language (VTML)
- Java
- JavaServer Pages (JSP)
- JRun tag set
- Wireless Markup Language (WML)
- Handheld Device Markup Language (HDML)
- Synchronized Multimedia Integration Language (SMIL)
- Interactive Fiction or Framework Markup Language (IMFL), partial support
- Real-time Markup Language (RTML), partial support

Coding tools, controls, options, wizards, color coding schemes, CodeSweepers, and validation support are also available for many other languages, including ActiveServer Pages (ASP), ActiveX, ActiveScript, Cascading Style Sheets (CSS), Dynamic HTML (DHTML), JavaScript, JScript, Perl, Personal Home Page (PHP), Structured Query Language (SQL), VBScript, and WDDX.

HomeSite+ for Dreamweaver MX does not support every language, but the program is highly flexible and extensible. You can extend its language support with the Tag Definitions Library and VTML. For details, see “Working with tag definitions” on page 109 and “Customizing the Development Environment” on page 175.

However, before you begin creating VTML files, see if someone else has already done the work for you. Some excellent resources include the http://devex.allaire.com/developer/gallery/index.cfmDeveloper’s Exchange/a, http://www.allaire.com/handlers/index.cfm?id=21790HomeSite Community Resources/a, http://hotscripts.com/HotScripts.com/a, and, for ASP and PHP coders, http://www.wilk4.com/asp4hs/ASP4HS/a and http://www.wilk4.com/asp4hs/php4hs.htmPHP4HS/a.

How a language is detected

On a document level, HomeSite+ for Dreamweaver MX determines the language based on the document's DOCTYPE declaration. If there is no declaration, the program treats the document content as HTML. In the case of XHTML, when you enable **Options > Set Document as XHTML**, the program treats the document as XHTML, even if there is no change to the DOCTYPE declaration.

On a character level, supported languages have their own start and end tags to alert the program that the tag content is written in a specific language.

The status bar displays the language of the code in the current document.

Setting options for markup languages

This section provides instructions for setting language-specific options. For setting specific options for XHTML, also see "Setting options for XHTML" on page 87.

To set language options:

1 Open the **Options > Settings > Markup Languages** pane.

2 Select the options you need as described in the following table.

Option	Result when selected
Lowercase all inserted tags	Tag editors and the Tag Inspector insert all lowercase code in the Editor. To convert tags to lowercase in existing documents, you can open the document, select **Edit > Convert Tag Case**, and select the lowercase option.
Always insert colors as hexadecimal values	Tag editors and Tag Inspector insert a RGB value like C0C0C0 and 008080 instead of "Silver" and "Teal". Different browsers interpret the names of colors differently, but if you use RGB values, the same color appears in every browser.
Force filenames to lowercase when inserting links	When dragging a file to the Editor, the inserted link contains a lowercase filename. For example, if you drag ABCs.htm to the Editor, the following code is inserted: ABC's Note that this option can cause links to break on UNIX servers.
When editing tags, return the output on a single line	Tag editors insert the code on a single, non-wrapping line in the Editor.

Include closing when inserting paragraph tag	The Paragraph toolbutton on the Common QuickBar inserts instead of just . Note that even if you clear this option, by default, Tag Completion inserts after you type . To disable this too, open the **Options > Settings > Editor > Tag Completion** pane and delete the P tag from the list.
Insert numeric values surrounded by quotes (Tag Inspector)	When an attribute has a numeric value, Tag Inspector inserts it surrounded by quotes, as tag editors do. For example, `width="20" height="20"`, not `width=20 height=20`.
Option	**Result when selected**
Minimize empty tags (for example)	If the current document is set or recognized to be an XHTML document, the toolbuttons on the QuickBar for empty tags insert minimized empty tags, for example and .
Enforce required attribute validation (tag editors)	The tag editor OK toolbutton is disabled until a value is entered for every required attribute. To require an attribute in a tag editor: 1 In HomeSite+ for Dreamweaver MX open the VTML file for the tag editor, in the <installation directory>\Extensions\TagDefs folder, in the folder for the applicable markup language. 2 Locate the `control` tag for the field to require. (Fields that are required in the W3C specification are visually marked with an asterisk). 3 Add the attribute `REQUIRED="YES"` to the tag. 4 Save the file. 5 Press Ctrl+Alt+Shift+C to clear the VTML cache. For more information, see ../VTML_Reference/control.html`control`/a in the VTML Reference.
Use tag in place of	The Italic toolbutton on the Font QuickBar inserts instead of .
Use tag in place of	The Bold toolbutton on the Font QuickBar inserts instead of .

6 In the Insert special characters as box, select how characters should be represented in the code when you insert them from the **View > Extended and Special Characters** palette. Following is an example for each option:

- **Character entities** Insert the ampersand (&) as `&`
- **Decimal references** Insert ampersand (&) as `&#38;`
- **Hexadecimal references** Insert ampersand (&) as `&#x26;` (a RGB value)

7 Open the HTML/XHTML pane under the Markup Languages pane and select from the options in the following table.

Option	Result when selected
Use title of dragged documents as the link description	When you drag a file to the Editor to insert a link, the programs checks the file for a title and uses any title that it finds as the link text.
Align Center or Align Right	Specify tags you want to insert when clicking these toolbuttons on the Common QuickBar.
Use the VTML Tag Dialog for:	Select tags for which you want to use the Visual Tool Markup Language (VTML) tag editor. For more information, see "About VTML tag editors" on page 73.

8 Click Apply.

9 Open the **Options > Customize** dialog box.

10 On the toolbars tab, in the Visible Toolbars list, select the toolbar for every language that you use. The toolbars are added to the QuickBar, providing you with a quick way to insert code for those languages.

For more information about displaying toolbars and adding toolbuttons, see "Customizing toolbars" on page 32.

11 If you script in ActiveX, add the ActiveX toolbutton to a toolbar.

12 If you process documents as ActiveScript, add the Execute Current Document as ActiveScript toolbutton to a toolbar.

13 Click Close to save your changes.

Coding in XHTML

Within HomeSite+ for Dreamweaver MX, you can set XHTML-specific options, use coding tools that support the XHTML 1.0 specification, reformat your code using a CodeSweeper or HTML Tidy, and validate against the XHTML 1.0 specification.

You can also use the W3C XHTML validator sites for http://validator.w3.org/ web-based files/a or http://validator.w3.org/file-upload.htmllocal files/a.

The rest of this section defines XHTML, describes its benefits, and explains how to take advantage of the XHTML support and features in HomeSite+ for Dreamweaver MX.

Note
To specify the priority of XHTML in the Tag Definitions Library, see "Working with tag definitions" on page 109.

What is XHTML?

XHTML (Extensible Hypertext Markup Language) is a reformulation of HTML as an XML language. It is almost identical to HTML 4.01, but it is more strict and clean.

Using XHTML lets you reap the benefits of XML, while ensuring the backward and future compatibility of your web documents. Following are some specific reasons for using XHTML:

- It is designed to replace HTML. If you want your website to be a visible part of the Internet in the future (and for it to be rendered properly), it is a sound investment to begin coding in XHTML now.
- It is designed to be operable across devices, not just on PCs.
- XHTML is an XML language, so it offers the potential for extensibility. It also lets you view, edit, and validate XHTML documents with standard XML tools.
- XHTML documents can use applications such as scripts and applets that rely on the HTML Document Object Model or the XML Document Object Model.
- It requires quality code. This reduces the discrepancies between how different user agents such as web browsers render a web document, because most of these discrepancies are caused by incorrect or poorly formatted code.

XHTML works on HTML 4-compliant user agents and XML user agents, so you can switch to XHTML without excluding anyone or waiting for XML-based browsers or other user agents to become more prevalent. Because XHTML is so similar to HTML, it is not difficult to switch from HTML to XHTML.

For more information, see the W3C specification for http://www.w3.org/TR/xhtml11/XHTML 1.1 - Module-Based XHTML/a or http://www.w3c.org/TR/xhtml1/XHTML 1.0/a. You can also search http://www.tutorialfind.comtutorialfind/a for XHTML. Other good resources are *Beginning XHTML*, published 2000 by Wrox Press, and http://www.w3schools.com/xhtml/default.aspWelcome to XHTML School/a.

Setting options for XHTML

This section has instructions for setting options that are optimal for XHTML coding.

For more information about these options and other language options that are not specific to XHTML, see "Setting options for markup languages" on page 83.

To set language options that are optimal for XHTML:

1 Open the **Options > Settings > Markup Languages** pane.

2 Select the following options:

- Lowercase all inserted tags.
- Include closing </p> when inserting paragraph tag
- Insert numeric values surrounded by quotes (Tag Inspector).
- Minimize empty tags (such as, <br/>).

3 In the Insert special characters as box, select Character entities.

(Numeric references must have the ampersand encoded; for example & instead of &).

4 Under Markup Languages, in the HTML/XHTML pane, select Compatibility mode for older browsers (such as, space before "/>").

This ensures that your code displays properly in older browsers; for example, by inserting a space before minimized empty tags (
 instead of <br/>).

5 If, when you select **Options > Set Document as XHTML**, you want to be able to choose between inserting the doctype declaration for the XHTML Strict, Frameset, or Transitional DTD, or not inserting or replacing a doctype declaration, select the option to Display DTD Selection Dialog When the XHTML Namespace is Specified. (The dialog box that appears has options to Always replace with this DTD or Don't ask me again, in case you change your mind).

If you clear this option, when you set a document as XHTML, no dialog box appears and no doctype declaration is inserted or replaced.

6 Click Apply.

Using coding tools that support XHTML

Tag editors, the Tag Inspector, Tag Insight, and other coding tools support the XHTML 1.0 specification. This means that, in an XHTML document, tag editors insert XHTML-compliant code, the Tag Inspector displays information in its attribute table that is appropriate for XHTML, and Tag Insight displays a list of completion options that are appropriate for XHTML.

The only thing you must do to take advantage of this support is to create an XHTML document, or set the current document to be XHTML. This enables XHTML support in the document.

To create an XHTML document, do either of the following:

- Create a document with one of the templates available in the **File > New > XHTML** dialog box (Strict, Transitional, or Frameset).
- Open or create a document containing a Strict, Transitional, or Frameset `doctype` declaration.

To set the current document to be XHTML:

1 Select **Options > Set Document as XHTML** for the current document.

This will have no effect if the current document is read-only.

2 In the **Options > Settings > Markup Languages > XHTML/HTML** pane, if the option to Display DTD Selection Dialog When the XHTML Namespace is Specified is selected, a dialog box appears. Select the `doctype` declaration for the DTD that you will use (there are three versions of the XHTML 1.0 specification).

3 In the dialog box, select one of the following options:

- **No replacement** Select this if you do not want to change or add a `doctype` declaration, for example if you are writing XHTML code to include in another page that already has its own `doctype` declaration.
- **Always replace with this DTD** Select this if you do not want the dialog box to display again and you will always use the same `doctype` declaration.
- **Don't ask me again** Select this if you do not want to display this dialog box, and you never want to replace or insert a `doctype` declaration.

4 Click OK.

If you must undo this, select **Options > Set Document as XHTML** again with the XHTML document as the current document in the Editor. In the Confirm dialog box, click Yes.

Setting color coding for XHTML

This section explains how to use the default color coding scheme for XHTML.

You can modify this default scheme as necessary. For more information, see "Using color coding schemes" on page 96.

To use the XHTML color coding scheme in your documents:

1 In the **Options > Settings > Editor > Color Coding** pane, select Extensible Hypertext Markup Language (XHTML) and click Set as Default.

2 Click Apply.

Using CodeSweepers to convert your code to XHTML

Following is one way to configure a CodeSweeper to convert your code to XHTML.

To use a preconfigured XHTML CodeSweeper:

1 Open the **Options > Settings > CodeSweeper > HTML Tidy CodeSweeper** pane.

2 Select Macromedia HTML Tidy Settings and click Edit Profile.

This opens the default settings for editing.

3 In the Formatting options box, select Convert document to XHTML.

4 In the Char encoding box, select UTF8.

5 (Optional) Set any other options.

For more information, see "Formatting code with CodeSweepers" on page 98.

6 Click Apply.

7 To use the CodeSweeper on the current document, select **Tools > CodeSweeper > Macromedia HTML Tidy Settings**.

Validating XHTML code

This section explains how to configure the validator to check against the XHTML 1.0 specification, and describes what the XHTML validator checks.

For more information about the validator, see "Validating code" on page 104.

Configuring the validator for XHTML

Following is one way to configure the validator to check XHTML code.

To configure the validator for XHTML:

1 In the **Options > Settings > Validation** pane, select XHTML 1.0 Strict.

2 Click Validator Settings.

3 In the Validator Configuration dialog box, on the Options tab, in the Report box, select every type of error except CFML Compiler Errors.

4 In the Other box, select the options to check for quotes in text and report special characters.

Checking for quotes ensures that each quotation mark is followed by another of the same type (' or ").

Reporting special characters catches errors like having & instead of &.

The Tags tab is for customizing the structure and requirements of a tag set; for example, to require an attribute for a tag.

The Versions tab is for extending the tag sets against which you can validate.

The Values tab is for validating regular expressions. All of these are unnecessary for a standard XHTML document, so this configuration is complete.

5 Click OK.

6 In the Settings dialog box, click Apply.

What the XHTML validator checks

When you configure the validator to check your code against the XHTML 1.0 specification, it checks the following rules:

- Tag element and attribute names must be in lowercase.
- Document must have an XHTML `doctype` declaration.
- The document must have tags for `html`, `head`, and `body`.
- The `title` attribute must be in the `head` element.
- Elements must nest symmetrically; for example:
  `<i><b></b></i>`, not `<i><b></i></b>`.
- Every tag must have an end tag (in pairs `<></>`, or shortcut form `</>`, `< />`).
- Empty elements must have a corresponding end tag or contain an ending slash in the start tag.
- Every attribute value must be quoted.
- An attribute value cannot be minimized. For example, you cannot have `<input checked>`; it must be `<input checked="checked">`.
- If an element has a `lang` attribute, it must also have an `xml:lang` attribute.

Note
Many of these rules also apply to HTML.

Using regular expressions

This section describes regular expressions and provides information and examples for using them in HomeSite+ for Dreamweaver MX.

About regular expressions

A regular expression is a pattern that defines a set of character strings. The **RegExp** parser in HomeSite+ for Dreamweaver MX evaluates the indicated text and returns each matching pattern.

As in arithmetic expressions, you can use various operators to combine smaller expressions and basic regular expressions can be concatenated into complex criteria. For more information, see "Anchoring a regular expression to a string" on page 95.

In HomeSite+ for Dreamweaver MX, you can use regular expressions for extended searches and validating code:

- **Extended search** Search a document for a pattern rather than a specific string of characters. For example, search for repeated words with `(" [A-Za-z] "){2,}`.

  In an extended search, all matches are added to the list of results. But in an extended search and replace, matches are immediately replaced with the replacement text. So consider not only what is matched but what is *not* matched; for example, there might be two or more strings that you must replace with the same text. Also, it is always a good idea to back up your files first!

  In a search and replace operation, the RegExp engine processes the entire document; it does not parse on a line-by-line basis. This affects the way that you should use characters such the asterisk (*), carat (^), and dollar sign ($).

- **Code validation** Define a special requirement for your code in the validator. For example, set the validator to search for (and flag as an error) any user input box that does not have an associated error message.

For more information, see "Using extended search commands" on page 160 and "Validating code" on page 104.

Writing regular expressions

The rules listed in this section are for creating regular expressions in HomeSite+ for Dreamweaver MX; the rules used by other RegExp parsers might differ.

To construct a regular expression in Expression Builder:

1 Select **Tools > Insert Expression**.

2 In the Expression Builder dialog box, in the tree pane, navigate to **Expression Elements > Constants > Regular Expressions**.

3 (Optional) For syntax and usage information for a regular expression, click the regular expression in the right pane and press F1.

4 Double-click a regular expression in the right pane.

The regular expression appears in the working area at the top of the dialog box.

5 (Optional) Modify the contents of the working area.

You can insert other expressions in the working area to construct a more complex expression. Expressions are inserted at the cursor position.

6 Click Insert.

The contents of the working area in Expression Builder are inserted into the active document, at the cursor position.

Using a special character

Because special characters are the operators in regular expressions, in order to represent a special character as an ordinary one, you must precede it with a backslash. To represent a backslash, for instance, use a two backslashes (\\).

Creating a single-character regular expression

You can use regular expressions in the **Search > Extended Find and Replace** command to match complex string patterns.

The following rules govern one-character RegExp that match a single character:

- Special characters are: + * ? . [ ] ^ $ ( ) { } | \ &
- Any character that is not a special character matches itself.
- Use the keyboard (Tab, Enter) to match whitespace characters.
- The asterisk (*) matches the specified characters throughout the entire document.
- The carat (^) matches the beginning of the document.
- The dollar sign ($) matches the end of the document.
- A backslash (\) followed by any special character matches the literal character itself; that is, the backslash escapes the special character.
- The pound sign (#) and hyphen (-) characters must be escaped in expressions (## --) just as though they were special characters.
- A period (.) matches any character, including a new line. To match any character except a new line, use `[^#chr(13)##chr(10)#]`, which excludes the ASCII carriage return and line feed codes.
- A set of characters enclosed in brackets ([]) is a one-character RegExp that matches any of the characters in that set. For example, `[akm]` matches an *a*, *k*, or *m*. Note that if you want to include a closing square bracket (]) in square brackets, it must be the first character. Otherwise, it does not work, even if you use \].

- Any regular expression can be followed by one of the following suffixes:
  - {m,n} forces a match of m through n (inclusive) occurrences of the preceding regular expression
  - {m,} forces a match of at least m occurrences of the preceding regular expression

  The syntax {,n} is not allowed.
- A range of characters can be indicated with a dash. For example, `[a-z]` matches any lowercase letter. However, if the first character of the set is the caret (^), the RegExp matches any character except those in the set. It does not match the empty string. For example, `[^akm]` matches any character except *a*, *k*, or *m*. The caret loses its special meaning if it is not the first character of the set.
- All regular expressions can be made case-insensitive by substituting individual characters with character sets, for example, `[Nn][Ii][Cc][Kk]`.

Using a character class

You can specify a character by using a POSIX character class. You enclose the character class name inside two square brackets, as in this example:

```
"Macromedia's Website","[[:space:]]","*","ALL")
```

This code replaces all the spaces with *, producing this string:
`Macromedia's*Website`

The following table shows the supported POSIX character classes:

Character class	Matches
alpha	Any letter, [A-Za-z]
upper	Any uppercase letter, [A-Z]
lower	Any lowercase letter, [a-z]
digit	Any digit, [0-9]
alnum	Any alphanumeric character, [A-Za-z0-9]
xdigit	Any hexadecimal digit, [0-9A-Fa-f]
space	A tab, new line, vertical tab, form feed, carriage return, or space
print	Any printable character
punct	Any punctuation character: ! ‘ # S % & ‘ ( ) * + , - . / : ; < = > ? @ [/] ^ _ { | } ~
graph	Any character defined as a printable character except those defined as part of the *space* character class
cntrl	Any character not part of the character classes: [:upper:], [:lower:], [:alpha:], [:digit:], [:punct:], [:graph:], [:print:], [:xdigit:]

Creating a multi-character regular expression

You can use the following rules to build multi-character regular expressions:

- Parentheses group parts of regular expressions together into grouped subexpressions that can be treated as a single unit. For example, (ha)+ matches one or more instances of "ha".
- A one-character regular expression or grouped subexpressions followed by an asterisk (*) matches zero or more occurrences of the regular expression. For example, [a-z]* matches zero or more lowercase characters.
- A one-character regular expression or grouped subexpressions followed by a plus (+) matches one or more occurrences of the regular expression. For example, [a-z]+ matches one or more lowercase characters.
- A one-character regular expression or grouped subexpressions followed by a question mark (?) matches zero or one occurrences of the regular expression. For example, xy?z matches either "xyz" or "xz".
- The concatenation of regular expressions creates a regular expression that matches the corresponding concatenation of strings. For example, [A-Z][a-z]* matches any word, regardless of case.
- The OR character (|) allows a choice between two regular expressions. For example, jell(y|ies) matches either "jelly" or "jellies".
- Braces ({}) are used to indicate a range of occurrences of a regular expression, in the form {m, n} where m is a positive integer equal to or greater than zero indicating the start of the range and n is equal to or greater than m, indicating the end of the range. For example, (ba){0,3} matches up to three pairs of the expression "ba".

Using a back reference

HomeSite+ for Dreamweaver MX supports back referencing, which allows you to match text in previously matched sets of parentheses. You can use a backslash followed by a digit *n* (\n) to refer to the n^{th} parenthesized subexpression.

One example of how you can use back references is searching for doubled words, for example, to find instances of "is is" or "the the" in text. The following example shows the syntax you use for back referencing in regular expressions:

```
("There is is coffee in the the kitchen",
        "([A-Za-z]+)[ ]+\1","*","ALL")
```

This code searches for words that are all letters ([A-Za-z]+) followed by one or more spaces [ ]+ followed by the first matched subexpression in parentheses. The parser detects the two occurrences of *is* as well as the two occurrences of *the* and replaces them with an asterisk, resulting in the following text:

```
There * coffee in * kitchen
```

Anchoring a regular expression to a string

You can anchor all or part of a regular expression to either the beginning or end of the string being searched:

- If a caret (^) is at the beginning of a subexpression, the matched string must be at the beginning of the string being searched.
- If a dollar sign ($) is at the end of a subexpression, the matched string must be at the end of the string being searched.

Regular expression examples

The following table shows some regular expressions and describes what they match:

Expression	Description
`[\?&]value=`	A URL parameter value in a URL
`[A-Z]:(\\[A-Z0-9_]+)+`	An uppercase DOS/Windows full path that is not the root of a drive, *and* that has only letters, numbers, and underscores in its text
`[A-Za-z][A-Za-z0-9_]*`	A ColdFusion variable with no qualifier
`([A-Za-z][A-Za-z0-9_]*)(\.[A-Za-z][A-Za-z0-9_]*)?`	A ColdFusion variable with no more than one qualifier; for example, Form.VarName, but not Form.Image.VarName
`(\+|-)?[1-9][0-9]*`	An integer that does not begin with a zero and has an optional sign
`(\+|-)?[1-9][0-9]*(\.[0-9]*)?`	A real number
`(\+|-)?[1-9]\.[0-9]*E(\+|-)?[0-9]+`	A real number in engineering notation
`a{2,4}`	Two to four occurrences of “a”: aa, aaa, aaaa
`(ba){3,}`	At least three “ba” pairs: bababa, babababa, ...
`(" [A-Za-z] "){2,}`	At least two occurrences of the same word

Using color coding schemes

Language color coding helps you to identify code blocks in large documents and distinguish between language types. The default color coding scheme is for HTML, but you can use any of the following color coding schemes for your documents:

- ActiveServer Pages (ASP), with schemes for generic ASP, ASP with JScript, and ASP with VBScript
- Cascading Style Sheets (CSS)
- Document Type Definition (DTD)
- Extensible Hypertext Markup Language (XHTML)
- HTML Full Tag (colors entire tags rather than the elements inside tags)
- Java
- JavaScript
- JavaServer Pages (JSP)
- Perl
- Personal Home Page (PHP)
- Structured Query Language (SQL)
- Text
- VisualBasic Script (VBScript)
- Visual Tools Markup Language (VTML)

To change the color coding scheme to use in your documents:

1 In the **Options > Settings > Editor > Color Coding** pane, select the color coding scheme to use and click Set as Default.

2 Click Apply.

Setting the supported file types for a scheme

The color schemes for the supported languages are grouped by file extensions in the Color Coding pane. You can change the file types for a scheme.

To modify file types for a scheme:

1 Select **Options > Settings > Editor > Color Coding**.

2 Select the scheme you want to change.

3 Click Edit Extensions to open the edit dialog box.

4 Edit the extensions list as needed.

Be sure to separate entries with a semi-colon.

5 Click OK to apply the new list.

You can click Reset to Defaults at any time to use the original values.

Setting the display of tags in the Editor

You can customize a color scheme by changing its default colors and font styles.

To change a color scheme:

1 Select **Options > Settings > Editor > Color Coding**.

2 Choose the scheme you want to change.

3 Click Edit Scheme to open the edit dialog box.

The preview pane shows the options that are set for the selected scheme.

4 Select an item from the Elements list.

5 To change either the foreground or background color for the element, first clear the Use default option below the element color box.

6 Right-click the color box and select a color from the Color dialog box.

7 To change a font style select one of the style boxes.

8 Repeat these steps as needed for other language elements.

9 Click OK to save the changes.

You can click Reset to Defaults at any time to use the original values.

Formatting code with CodeSweepers

CodeSweepers automate the task of formatting your code correctly. It is especially useful in the following situations:

- Enforcing a uniform coding style for developers by having them use the same Codesweeper options
- Cleaning up the code formatting of a project as you review its documents
- Applying formatting rules after using a visual editor

HomeSite+ for Dreamweaver MX includes several CodeSweepers, each configured for a specific type of development. You can edit these CodeSweepers, and you can create your own CodeSweepers.

About Macromedia CodeSweepers

CodeSweepers are available for HTML only, HTML/CFML, JSP, Web-XML, and WDDX. **My CodeSweeper** is useful for testing and editing, and **Macromedia Default HTML Tidy Settings** is a preconfigured HTML Tidy for basic code preservation.

For information on formatting documents so that they are XHTML-compliant, see "Using CodeSweepers to convert your code to XHTML" on page 89.

About HTML Tidy

HTML Tidy is an independent, open source code utility for verifying and formatting HTML code. It was developed by Dave Ragget under the auspices of the W3C. HomeSite+ for Dreamweaver MX provides HTML Tidy as an alternative to the Macromedia CodeSweepers. It has additional language support, and useful features such as line wrapping, converting tags, and working in XML.

If new features are added to HTML Tidy between HomeSite+ for Dreamweaver MX releases, you can update your version of HTML Tidy.

To upgrade HTML Tidy:

1 Expand the **Options > Settings > CodeSweepers** list and select a CodeSweeper under HTML Tidy CodeSweepers.

2 At the top of the pane for the CodeSweeper, click the HTML Tidy website link.

3 In the HTML Tidy website, download the latest version.

4 Close HomeSite+ for Dreamweaver MX.

5 Copy the tidy.exe files to the root directory of your HomeSite+ for Dreamweaver MX installation.

6 Restart HomeSite+ for Dreamweaver MX.

Updating the HTML Tidy version does not affect your configuration settings.

Editing an HTML Tidy CodeSweeper

The HTML Tidy CodeSweepers pane provides access to all the supported options, but you can also edit a profile directly. This is especially useful in these situations:

- When you do not want an option to appear in the CodeSweeper pane, delete the name:value pair for the option from the TDY file.
- When you download a new HTML Tidy version and want to enable a new option, add the option to the TDY file. The HTML Tidy website lists new features and the correct syntax for enabling them.

To edit an HTML Tidy CodeSweeper:

1 Open the TDY file from the \Extensions\Codesweepers directory.

2 Save a backup copy of the file.

3 Add, edit, and delete the name:value pairs as needed.

4 Save the file.

The HTML Tidy CodeSweeper pane displays the updated information.

Deleting an HTML Tidy CodeSweeper

The best way to delete an HTML Tidy CodeSweeper is to delete the CodeSweeper's TDY file from \Extensions\Codesweepers. You can do this in one of the Files tabs or in Windows Explorer.

Running a CodeSweeper

You can run the default CodeSweeper or specify a CodeSweeper to use.

To run a CodeSweeper, do one of the following:

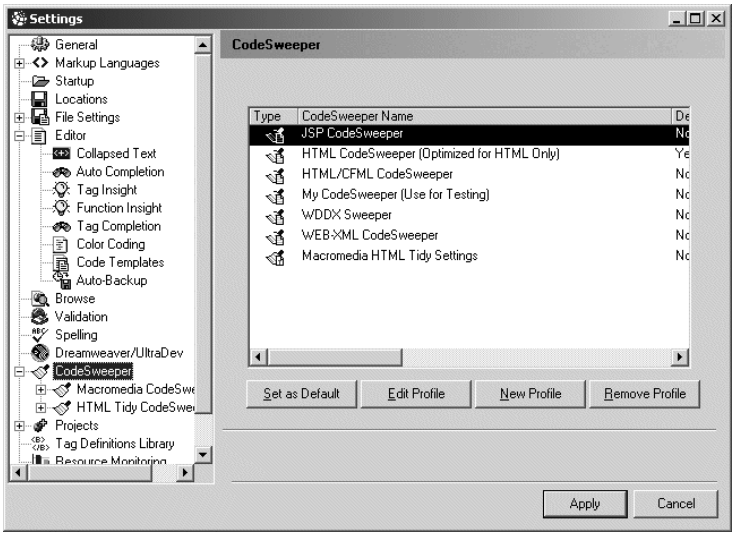
- Select **Tools > CodeSweeper** > Default CodeSweeper.
  The default Code Sweeper appears in the drop-down list with a red checkmark.
  To change the default CodeSweeper, see "Managing CodeSweepers" on page 100.
- Select **Tools > CodeSweeper**, and select a CodeSweeper from the drop-down list.

Managing CodeSweepers

You can specify a default CodeSweeper, and create, edit, and delete a CodeSweeper.

To set the default CodeSweeper:

1 In the **Options > Settings > CodeSweeper** pane, select the CodeSweeper and click Set as Default and then click Apply.



To create a CodeSweeper:

1 In the **Options > Settings > CodeSweeper** pane, click New Profile and enter a name for the CodeSweeper.

2 In the Select the CodeSweeper type box, select Macromedia CodeSweeper or HTML-Tidy CodeSweeper and click OK.

3 Select the node for the new HTML Tidy CodeSweeper under Macromedia CodeSweepers or HTML Tidy Codesweepers and complete the pane of information for the new CodeSweeper.

For a Macromedia CodeSweeper, see “Setting general options” on page 102 and “Setting tag-specific options” on page 103.

For an HTML Tidy CodeSweeper, see http://www.w3c.org/People/Raggett/tidy/#helphttp://www.w3c.org/People/Raggett/tidy/#help/a.

4 Click Apply.

To edit a Macromedia CodeSweeper:

1 In the **Options > Settings > CodeSweeper** pane, select the Macromedia CodeSweeper and click Edit Profile.

2 Update information on the General Settings and Tag-specific Settings tabs.

For more information, see "Setting general options" on page 102 and "Setting tag-specific options" on page 103.

3 Click Apply.

To edit an HTML Tidy CodeSweeper, see "Editing an HTML Tidy CodeSweeper" on page 99.

To delete a Macromedia CodeSweeper:

1 In the **Options > Settings > CodeSweeper** pane, select the CodeSweeper and click Remove Profile.

HomeSite+ for Dreamweaver MX immediately removes the CodeSweeper from the list without asking for a confirmation.

2 Click Apply.

To delete an HTML Tidy CodeSweeper, see "Deleting an HTML Tidy CodeSweeper" on page 99.

Setting Macromedia CodeSweeper options

The options described in this section apply only to Macromedia CodeSweepers.

For a description of options for HTML Tidy CodeSweeper, see http://www.w3c.org/People/Raggett/tidy/#helphttp://www.w3c.org/People/Raggett/tidy/#help/a.

Setting general options

The following table describes the options on the General Settings tab of a Macromedia CodeSweeper pane:

General option	Description
Format tag names	Formats tag/attribute names in one of these ways: • Preserve Case: Ignores the case of names. • All Upper Case: Forces every name to uppercase. • All Lower Case: Forces every name to lowercase.
Format attribute names	
Format event names	Formats event names in the same ways as described for tag and attribute names, with one additional option: • Mixed Case: Forces every event name to be mixed case; for example, `onclick` becomes `onClick`. This is recommended for case-sensitive JavaScript event handler names.
Attribute value quoting	Handles quotes around attributes in one of these ways: • Preserve Existing Quoting: Ignores quoting for attributes. • Quote As Needed: Inserts quotes around an attribute value when it is required. • Quote All: Inserts quotes around every attribute value.
Trim white space between tags	Deletes extra spaces that are produced by visual editors and some code generation tools. It is enabled by default. For best results, leave it enabled; you can disable it for individual tags.
Run in “silent mode”	Eliminates the display of CodeSweeper alerts. Use this option when you run a CodeSweeper unattended.
Write errors to a log file	Captures errors found by the CodeSweeper. If selected, you can specify a custom location for the log file in the text box.

You can also change the case of tags on demand for the current document, by selecting **Edit > Convert Tag Case**. This operation cannot be undone.

Macromedia CodeSweepers can process XML namespaces in tags—for example, `<xsl:thistag name="foo"/>`—because they allow the use of the colon character (:) in tag names.

Setting tag-specific options

You can set formatting rules for an existing or new tag, and you can remove a tag, so that the CodeSweeper ignores that tag in the documents it sweeps.

About the tag-specific settings

You can set the following options for each tag on the Tag-specific Settings tab of a Macromedia CodeSweeper:

Tag-specific option	Description
Add Tag	Adds a tag to be formatted by the selected CodeSweeper.
Update Tag	Updates rules for how the CodeSweeper formats the tag.
Remove Tag	Removes a tag so that the CodeSweeper ignores it.
Add a new line	Adds a new line in the code (not in the rendered page) before the start tag, after the start tag, before the end tag, and/or after the end tag.
Indent by	Indents the tag, the number of spaces or tabs that you specify.
Indent other sub tags from this tag	Indents any tags that are nested inside the selected tag.
Do not trim white space around this tag	Preserves white space around the selected tag. This option is only effective when you select the Trim white space between tags option on the General Settings tab.
Preserve tag formatting	Causes the CodeSweeper to ignore the tag, without you needing to remove it. If you encounter problems parsing server-based script code, try enabling this option.
Preserve tag contents	Causes the CodeSweeper to ignore everything in the tag, including nested tags.
Strip tag from document	Removes the tag from the document being sweeped. This is especially useful for deleting formatting tags when you must redesign a website to use CSS, and for deleting superfluous and unwanted tags that are inserted by visual editors and some code generation tools. If you only want to remove the end tag; for example, `</p>` or `</li>` tags, select (End tag only).

To set rules for a tag:

1 In the **Options > Settings > CodeSweeper** pane, select a Macromedia CodeSweeper and click Edit Profile.

2 Click the Tag-specific Settings tab and select a tag name from the list.

Note
If you select All Other Tags, the rules you specify on this tab will apply to every tag that is not in the list but is found in the document that you are sweeping.

3 If you do not see the tag you need in the list, add a tag, as described in the following procedure.

4 Modify the formatting rules for the tag and click Update Tag.

5 Click Apply.

To add a tag and set formatting rules:

1 In the **Options > Settings > CodeSweeper** pane, select a Macromedia CodeSweeper and click Edit Profile.

2 Click the Tag-specific Settings tab and click Add Tag.

3 In the Add Tag Specific Settings dialog box, enter the tag name and click OK.

4 Specify the formatting rules for the tag and click Update Tag.

For more information, see "About the tag-specific settings" on page 103.

5 Click Apply.

To remove a tag:

1 In the **Options > Settings > CodeSweeper** pane, select the tag to remove and click Remove Tag.

HomeSite+ for Dreamweaver MX immediately removes the tag from the list without asking for a confirmation.

2 Click Apply.

Validating code

You can use the HomeSite+ for Dreamweaver MX validator, or integrate the CSE HTML validator to work within HomeSite+ for Dreamweaver MX.

Using the default Validator

You can use the HomeSite+ for Dreamweaver MX validator to check and report syntax errors in HTML (including different browser extensions), XHTML, CFML, JSP, SMIL, and WML.

Configuring the validator

Configuration options are available to meet a wide range of validation requirements. This section contains a general procedure for configuring the validator, and procedures for setting specific options.

Note
If you validate both CFML and HTML (or XHTML) in a single document, the validator cannot assess the pound sign (#). This is because, in CFML, the single pound is an error and the double pound is correct; while in HTML/XHTML, the double pound sign is an error and the single pound is correct.

To configure the validator:

1 Open the **Options > Settings > Validation** pane.

2 Select the tag sets against which to validate.

Some tag sets are built on top of other tag sets. For example, if you select HTML 4.0, HTML 3.2 and HTML 2.0 are automatically selected as well, because the definition for HTML 4.0 is incomplete without them.

3 Click Validator Settings.

4 In the Validator Configuration dialog box, complete the Options tab as needed:

- In the Report section, specify the types of errors that you want the validator to report. To limit the number of errors that appear in the Results window after validating a document, specify a maximum number of errors to report.
- In the Other section, select options as needed; for example, to check the code for mismatched quotes and unconverted special characters.

  If you use ASP or PHP in your code, select the option to ignore that code. The validator does not fully support these languages and might incorrectly parse the code.

5 If you must validate something beyond the scope of the specifications selected in the **Options > Settings > Validation** pane, use the other tabs in this dialog box.

Following is a description of what you can do on each tab:

- **Tags** Require a tag, specify if a tag is case-sensitive, indicate if a tag has a closing tag and if it is optional or required, add or remove a tag, add or remove an allowed attribute for a tag, select a valid value for an attribute from a list of available values, require an attribute, allow a required attribute to be absent, and add or remove a **context** for a tag (a tag in which the tag can be nested).
- **Values** Add values, specific values or regular expressions, to the list of valid values that you can specify for an attribute. The check boxes on this tab allow you to filter the list of values.

  For information about regular expressions, see “Using regular expressions” on page 91.

- **Versions** Create a new family of languages, for example the XML language MathML, and add versions to language families. A version inherits everything from the parent language.

  You can add tags to a new family on the Tags tab, or you can back up the validator (VTV) files in the \Extensions and \Extensions\TagDefs directories, and then modify them in the Editor.

Caution
The changes you make on these tabs take effect immediately and cannot be undone, even if you click Cancel. Be especially careful when using the Remove toolbutton.

6 Click OK and then click Apply in the Settings dialog box.

To require an attribute for a tag:

1 Select **Options > Settings > Validation** and click Validator Settings.

2 In the Validator Configuration dialog box, click the Tags tab.

3 Expand a set of tags, for example HTML 4.0.

4 Expand a tag in the list.

5 Select the tag's Required folder and click Add.

6 In the Required Attribute dialog box, enter the name of an existing attribute and click OK.

  If the attribute is not in the Attribute folder, add it to the Attribute folder and then add it to the Required Attributes folder.

7 To remove a required attribute, select the attribute in the Required folder and click Remove.

8 If you are done setting validator options, click OK.

9 Back in the Settings dialog box, click Apply.

To validate against the HTML 4.0 specification:

1 In the **Options > Settings > Validation** pane, select HTML 4.0.

2 Click Validator Settings.

3 In the Validator Configuration dialog box, on the Options tab, in the Report box, select every type of error except CFML Compiler Errors.

4 In the Other box, select the options to check for quotes in text and report special characters.

  Checking for quotes ensures that each quotation mark is followed by another of the same type (' or "). Reporting special characters catches errors like having & instead of `&` in the HTML.

5 On the Tags tab, expand the HTML 4.0 node and select the noframes node.

6 In the Tag Options box, select Required in document and click Apply Options.

7 Expand the img node, select the Attributes folder and click Add.

8 In the New Attribute dialog box, enter alt and click OK.

9 Select the Required folder, and click Add.

10 In the Add Required Attribute dialog box, enter alt and click OK.

The Versions tab is for extending the tag sets against which you can validate, and the Values tab is for validating regular expressions. These are unnecessary for a standard HTML 4.0 document, so this configuration is complete.

11 Click OK.

12 Back in the Settings dialog box, click Apply.

Note
For an example of validating against the XHTML 1.0 specification, see "Configuring the validator for XHTML" on page 89.

Running the validator

You can run the validator for the current document or for a selected tag. However, validating a tag only checks the contents of the specific tag; for example, it does not check if a tag has an end tag or if the tag is in the wrong place in the document.

Note
If you validate both CFML and HTML in a single document, the validator cannot assess the pound sign (#). This is because, in CFML, the single pound sign is an error and the double pound sign is correct; while in HTML, the double pound sign is an error and the single pound sign is correct.

To validate the current document:

1 Select **Tools > Validate Current Document.**

The Validation Results pane displays a "No errors or warnings" message or lists the syntax errors that it found.

2 Double-click an error message to highlight it in the document.

You can then correct the code as needed.

To validate the contents of a single tag, do one of the following:

- Position the cursor inside the tag and select **Tools > Validate Current Tag**.
- Position the cursor inside the tag and press F6.
- Validate each tag automatically after you enter it. To do this, open the **Options > Settings > Validation** pane, select Tag validation - validates the current tag when the ">" key is typed, and click Apply.

The validator checks every tag as soon as you enter its closing bracket (>). It does not validate a tag whose closing bracket was entered automatically, for example by a tag editor or from a QuickBar toolbutton.

If something is wrong in the tag, an error message appears in red in the status bar.

Using the CSE HTML Validator

If you have installed this HTML validation tool, you can run it from HomeSite+ for Dreamweaver MX. The CSE Validator supports multiple languages and contains other useful features.

For more information, open the **Options > Settings >Validation** pane and click the CSE HTML Validator link to visit their website.

To use the CSE HTML Validator:

1 In the **Options > Settings > Validation** pane, select Use CSE HTML Validator when validating the entire document.

2 Click Configure CSE.

3 Complete the CSE HTML Validator Pro Configuration Editor dialog box.

For a description of each field on a tab, click Help.

4 Click CSE Options.

5 Complete the Validator Engine Options (Validator/Logging/Tools/Network) dialog box.

For a description of each field on a tab, click Help.

6 (Optional) To visit their website, click the CSE HTML Validator link.

7 When you are done, click Apply.

Working with tag definitions

Every tag in HomeSite+ for Dreamweaver MX has a definition file, written in VTML, that specifies the valid attributes for the tag and the content and formatting of the tag editor for the tag. Tag definitions are the source of information for tag editors, Tag Insight, Tag Tips, and Tag Inspector.

Following are the namespaces for the tag definitions that are installed with HomeSite+ for Dreamweaver MX: XHTML, HTML, CFML, VTML, JSP, Java, JRUN, WML, HDML, SMIL, IMFL, RTML, and Custom.

About namespace precedence

A version of a markup language (or **namespace**) can have a tag with the same name as a tag in another namespace. For example, both the JRun and Java namespaces contain a `servlet` tag. To avoid conflicts, the tag definitions for different namespaces are installed in different directories, in the \Extensions\TagDefs directory under the application root directory.

When you use a tag editor or the Tag Inspector, or Tag Insight is enabled, HomeSite+ for Dreamweaver MX needs to know which tag definition to use. The list in the **Options > Settings > Tag Definitions Library** pane represents the order in which HomeSite+ for Dreamweaver MX searches the directories when looking for a tag definition.

For example, if you enter `<region>` in a document, right-click it and choose Edit Tag, the SMIL tag editor appears, because no other directory contains a `<region>` tag. On the other hand, if you edit a `<servlet>` tag, either the Java or JRun version of the tag editor could appear. If JRun is higher on the namespace list than Java, then the program uses the JRun version of the servlet tag definition, without continuing its search to discover the Java version.

The following are two ways to manually override these defaults while working in the Editor:

- Enter the namespace and a colon (:) in the tag before the tag name; for example, enter `<java:servlet>`.
- Use Tag Chooser to insert the tag from whatever directory you want.
  For more information, see “Selecting a tag from Tag Chooser” on page 53.

Setting namespace precedence

This section explains how to set the order in which the program searches namespace directories for a tag definition.

To set the priority of namespaces:

1 In the **Options > Settings > Tag Definitions Library** pane, click Up and Down as necessary to set the list in the appropriate order.

  The namespace at the top of the list receives the highest priority.

2 If you do not want to search a namespace at all, clear the Enabled option for it.

3 If you want the current document's type to receive the highest priority regardless of the precedence set in this pane, select Current document's namespace receives highest priority.

For information about how a document's namespace is determined, see "How a language is detected" on page 83.

4 If you need the namespace to be treated as an XML application; for example, to disallow minimized boolean attributes such as `<dl compact>` (instead of `<dl compact="compact">`), then select the XML Based check box for the namespace.

5 Click Apply.

You can click Reset to Defaults at any time to reset the list to its default values.

Editing tag definitions

You can change a tag's definition from Tag Inspector, but the preferred method for changing a tag definition is to edit the VTM file directly, using VTML tag editors.

For more information about creating or modifying tag editors, see "Creating a tag definition file" on page 180.

To edit a tag definition using a VTML tag editor:

1 Open the VTM file from the appropriate language folder in \Extensions\TagDefs.

2 Right-click a tag, and select Edit Tag to change its attributes and values.

This method supports tag editors, Tag Inspector, Tag Insight, and Tag Tips (F2 in tag). It also ensures that these display properly on computers that are set for Large Fonts.

To edit a tag definition using Tag Inspector:

1 Click Edit Tag Definitions to open the Tag Definitions Library dialog box. The left pane lists all of the installed languages. You can edit the definitions list by using Add and Remove.

2 Expand a language entry to view individual tags. When you select a tag, the attributes and other components defined in its VTM file appear in the tabbed pane.

3 Make changes in the tag definition fields and click Done to save the changes.

This method has limitations, and it might cause the tag editor for the tag to not work properly. Be sure to back up the VTM file before making changes.

To create a tag definition, do one of the following:

- Modify an existing VTM file and save it as the new tag. This is the best way to create a tag definition file.
- Use the skeleton tag definition file, \Extensions\TagDefs\TagDefTemplate.vtm, as a starting point for creating your own tag.
- Create a tag definition from Tag Inspector. In this case, the new tag does not support a tag editor.

To add online Help for the tag, create an HTML file containing the Help text and save it with the tag name in the appropriate language folder in \Extensions\Docs.

Chapter 8

Accessing Data Sources

This chapter describes how to use the visual tools in HomeSite+ for Dreamweaver MX to accelerate development of data-driven ColdFusion applications.

Contents

- Introduction to database tools 114
- Working with data sources 114
- Using SQL Builder for database queries 116

Introduction to database tools

Visual database tools support application development that uses local, network, and remote data sources. When you configure a server, access to data sources becomes transparent.

The HomeSite+ for Dreamweaver MX database tools support:

- Developing a database remotely
- Browsing a data source schema and data
- Building SQL statements in a visual editor

Working with data sources

This section describes how to configure, connect to, and view ColdFusion data sources. It assumes that ColdFusion Server is installed and running properly.

Configuring a ColdFusion data source

A set of ODBC drivers and sample data is installed with ColdFusion Server. Native database drivers are installed with the Enterprise version; OLE-DB drivers are installed with the Professional and Enterprise versions.

To add a data source to ColdFusion Server:

1 In the ColdFusion Administrator, under Data Sources, select the ODBC or native drivers page.

2 Enter a name for the data source, select the appropriate driver, and click Add.

3 Click Verify.

HomeSite+ for Dreamweaver MX tests the connection and displays an error message if the connection fails.

Connecting to a data source

Data sources that are registered in the ColdFusion Administrator are automatically accessible from the Database tab in HomeSite+ for Dreamweaver MX. To access remote data sources, you can add remote ColdFusion Servers (RDS servers).

To add a remote server:

1 At the bottom of the Resources window, click the Database tab.

This is how the tab looks:

2 In the top pane, select Add RDS Server from the drop-down list.

3 Enter information in the Configure RDS Server dialog box and click OK.

For information about configuring a RDS server, see "Working with files on remote servers" on page 14.

Viewing a data source

As soon as a data source has been added in the ColdFusion Administrator, you can view it in HomeSite+ for Dreamweaver MX.

To open a data source:

1 In the Resources window, click the Database tab.

2 In the top pane, select a server from the drop-down list.

Accessing a large database on a remote servers might take a few moments.

3 Expand the data source tree:

- To expand the tree, click the plus sign (+) next to a data source name
- To view the database schema, click the plus sign (+) next to Tables
- To view the column definitions, click the plus sign (+) next to a table name

4 Double-click a table name.

The records in the table are displayed.

In the data browsing window you only can browse data. You cannot modify data or add new records here. Also, views are available only in databases supporting the creation of views or tables stored as queries.

To insert column or table names into a page:

- Drag a table or column from the Database tab to the Editor, to the desired position in your document.

  This is a useful shortcut when you are building database reports.

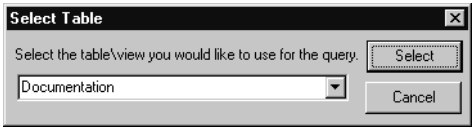
Using SQL Builder for database queries

SQL Builder is a powerful visual tool that you can use to build, test, and save SQL statements for use in application data queries. You can copy a SQL code block directly into your application or insert it into a `cfquery` tag. You can also use SQL Builder to test your queries.

To open SQL Builder, do one of the following:

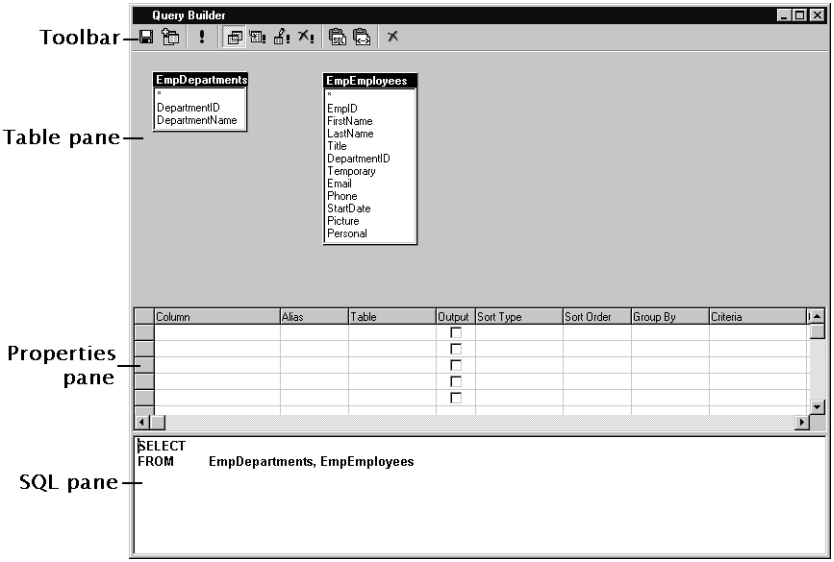
- In the Database tab, right-click a database name or a table and select New Query.
- Select **Tools > SQL Builder**. In SQL Builder, select a database from the drop-down list, and click New Query.
- Open the `cfquery` tag editor and click New Query.

In the Select Table dialog box (shown in the following graphic), select the table that you want to query against, and click Select.



The SQL Builder user interface

The following graphic shows the SQL Builder user interface:



The SQL Builder is divided into these sections:

- **Toolbar** Contains toolbuttons for SQL keywords and commands
- **Table pane** Provides a view of the tables in a query and lets you create joins between tables
- **Properties pane** Lets you set the properties of a query, such as the columns to select and the columns to update
- **SQL pane** Displays the SQL statement as it is being constructed.

  The SQL pane does not support reverse editing, so any changes that you make in this pane are not made in the Properties pane or the Table pane.

Writing an SQL statement

When SQL Builder opens, it displays a SELECT statement by default, since this is the most common type of query. If you have not yet selected a table for your query, you are prompted to select one.

SQL Builder supports these types of SQL statements:

- **Select** Retrieves rows or columns of data from a table
- **Insert** Adds data to a table
- **Update** Changes data in a table
- **Delete** Deletes data from a table

To change the type of SQL statement:

1 Open SQL Builder.

For instructions, see the opening SQL Builder procedure earlier in this section.

2 Select a statement type from the SQL Builder toolbar.

Building a SELECT statement

SQL SELECT statements let you specify data from which to build a recordset.

To create a SELECT statement:

1 Open SQL Builder.

For instructions, see the opening SQL Builder procedure earlier in this section.

2 Select a table to query.

3 Do one of the following:

- To add a table to the query, right-click in the Table pane and select Add Table, or click Add Table.
- To create a join between tables, drag a column name between tables.
- To delete a join, or to make it an inner or outer join, right-click the join and select the appropriate command.

4 Drag the columns to display onto the Column section of the Properties pane.

5 (Optional) Set additional query attributes in the Properties pane.

6 Save the query or insert it into a page.

To use a variable in a SQL WHERE clause:

1 Open SQL Builder.

For instructions, see the opening SQL Builder procedure earlier in this section.

2 In the Properties pane, click in the row for a column, in the Criteria column.

3 In the Select box, select a variable syntax.

4 Double-click on a variable placeholder name and enter a value.

To save a query:

1 Open SQL Builder.

For instructions, see the opening SQL Builder procedure earlier in this section.

2 On the Query toolbar, click Save Query.

3 In the dialog box that appears, enter a name for the query.

4 Click Save.

Saved queries are stored on the database server. They can be edited and used by anyone who has access to the server.

Note
HomeSite+ for Dreamweaver MX does not update the Properties and Table panes when you make changes in the SQL pane. If you edit a SQL statement in the SQL pane and save it, and then modify the Table pane or change any values in the Properties pane, a new SQL statement is generated which overwrites the changes that you made in the SQL pane.

Inserting SQL into a page

This section describes how to insert new and saved queries into a page in the Editor.

To insert a new query, do one of the following:

- If you opened SQL Builder from the CFQUERY tag editor, it prompts you to insert the query when you close the Editor.
- If you opened SQL Builder from a CFML page, it prompts you to insert the query when you close SQL Builder.
- To insert just the SQL statement from SQL Builder, click Copy SQL to Clipboard and close SQL Builder. HomeSite+ for Dreamweaver MX prompts you to save the query. Then you can paste the statement into your page.
- To insert the SQL statement into a `cfquery` tag, click Copy CFQUERY. Close SQL Builder and paste the query into your page. HomeSite+ for Dreamweaver MX prompts you to save the query.

To insert a saved query, do one of the following:

1 In the Resources window, click the Database tab.

2 Open a data source.

3 Open the Queries folder in the data source.

4 Drag and drop the query onto the page in the Editor.

or

1 On the CFML Basic QuickBar, click CFQUERY.

You can also select CFQUERY from Tag Chooser.

2 Enter a name for the query.

3 Click Insert Query.

4 In the select dialog box, select a server.

5 In the data source that you want to use, open the Queries folder.

6 Select a query and click Insert.

7 In the CFQUERY tag editor, click Apply.

The query appears in the Editor.

Testing and editing a query

You can refine SQL statements after constructing them.

Before inserting a query into an application, we recommend that you test it in SQL Builder. The following procedure explains how to run a query for testing.

Caution
Only test SELECT statements. When you click Run Query, the SQL statement is processed. If you run an INSERT, UPDATE, or DELETE statement, the changes that you coded in the SQL statement are made in the data source.

To test a query in SQL Builder:

1 Click Run Query.

2 HomeSite+ for Dreamweaver MX prompts you to enter values for the variables in the query.

If you save a query, you can edit it later. However, pages containing the query are not automatically updated. To make the changes take effect, you must reinsert the query into the pages.

To edit a query:

1 Open the query folder for the data source that you want to use.

2 Double-click a query.

3 Make changes in the Table pane and the Properties pane.

4 Save the updated query.

Note
HomeSite+ for Dreamweaver MX does not update the Properties and Table panes when you make changes in the SQL pane. If you write a query in the SQL pane and save it, the Properties and Table panes show no changes. But the SQL pane displays the SQL generated from changes made in the Properties and Table panes.

Chapter 9

Managing Projects

This chapter describes how to use a HomeSite+ for Dreamweaver MX project to manage the files in a website.

Contents

- Understanding projects 122
- Creating a project 124
- Working with a project 128
- Adding a project to source control 132

Understanding projects

This section describes the following topics HomeSite+ for Dreamweaver MX projects and how to structure them to fir your requirements.

What is a project?

A project is a collection of files that you use to develop and maintain your website. You can include any type of file in your project; for example, HTML files, Cascading Style Sheets, images, scripts, application code, and more. You can access these files from the Projects resource tab, regardless of where the physical files are located.

A project lets you create your own file system that contains only the files that you need for your website. This is especially useful if you do not have control of the physical disk drives in which the files are located; for example, on remote drives.

Why use a project?

These are some of the benefits of using projects:

- A project lets you access every file in your website from one central location, the Projects resource tab, even if the physical files are located in different local, network, and remote directories.
- If your website is contained in a project, you can copy it to a server, with confidence that no files are missing. This operation is known as **deployment**. For more information, see “Deploying Files” on page 143.
- You can perform maintenance tasks—such as search and replace and link verification—on an entire project, in one operation.

About project folders

Folders are the means of organizing the files in a project.

There are different types of folders:

- A **virtual folder** contains references to files from any number of directories on any number of disk drives.
- A **physical folder** is mapped to a single directory on a disk drive. You can filter the contents of a directory to exclude a type of file; for example, images or scripts.

  There are two types of physical folders: **manual-include** and **auto-include**.

The following table describes each folder type in detail:

Folder type	Icon	Description	Use to...
Virtual		Container for files that have no logical relationship to each other. You can put any collection of files in a virtual folder. You cannot put a virtual folder under source control. Also, to deploy a file that is in a virtual folder, you must manually select it.	Hold files that are located in different directories and disk drives, so you can access them from a single location.
Manual-include (physical)		Container for the files in a directory that you specify. Also, you can specify the file types of files that you want to add automatically. You cannot change the location of this folder.	Select the files in your project on a file-by-file basis, in a specific directory; for example, use this if you need only three files in a directory for your website.
Auto-include (physical)		Container for all files in a directory; except those with a file type that you exclude. This folder remains in sync with its corresponding directory on the disk drive. You can change the type of this folder to manual-include.	Organize pointers to files, or all files of a certain type, in a specific directory to be included in your project.

About the project file

The project (.apf) file controls the project folders and files that appear on the Projects resource tab for a project. It is the only element that HomeSite+ for Dreamweaver MX creates when you create a project. When you double-click a file on the Projects tab, the project file locates and opens the file.

The project file can do this because it contains the absolute path for every file in the project, similar to how a Windows desktop shortcut operates.

The project file is in WDDX format. For a virtual project folder, the project file lists each file with its absolute path. For a physical manual-include folder, the project file lists the directory for the files, the included file types, and each file with its relative path. For a physical auto-include folder, the project file lists the directory for the files, and any filtering by file type. For more information, see "About project folders" on page 122.

Creating a project

This section describes the following topics:

- Setting project options
- Creating a project
- Populating a project

Setting project options

Before creating a project, you can set options to apply to all new projects.

To set default options for new projects:

1 In the **Options > Settings > Projects** pane, set project options as needed.

The following table describes each project option:

Option	Result if completed
Maximum recent projects	Number of projects that appears on the Projects resource tab in the drop-down list is limited to the number that you specify here.
Default project folder type	By default, when you add a folder to a project, it is of the type **Manual include** or **Auto include**. For more information, see “About project folders” on page 122.
New auto-include folders include subfolders	When you add an auto-include folder to the project, it includes the files in its directory and also the files and folders in its subdirectories.
Include project resources check box and the resource table	When you add an auto-include or manual-include folder to a project, it includes files of the types that are selected in the resource table. For more information, see “Managing project resources” on page 130.
Options in the **Projects > Deployment** pane	See “Setting default deployment options” on page 144.

2 Click Apply.

Creating a project

This section describes how to create a project.

To create a project:

1 Select **Project > New Project**.

2 Complete the New Project dialog box, as follows:

- **Project Name** Enter a descriptive name that you want to display on the Projects resource tab, in the projects drop-down list.
- **Location of project file** Browse to the directory in which you want to store the project (APF) file. This is usually the project root folder.

  If you need to create this folder, create it in Windows Explorer.
- **Add all subfolders** (Optional) Select this option to include every folder below the project root.
- **Files types** (Optional) Select All files or, to limit the types of files that are included in your project, select a list of file types from the drop-down list.

  If you do not see the list of file types that you need, enter the file extensions for the file types to include, separated by a semicolon; for example `htm;html;gif;jpg;jpeg;png`.

3 Click OK.

HomeSite+ for Dreamweaver MX creates the project APF file in the directory that you specified.

Populating a project

Once you have created a project, you can add folders and files to it.

You can add any type of folder and any file to the project root. Within folders, you must add folders and files as described in the following table:

In this type of folder...	You can add this type of folder...	And these files...
Virtual	Virtual	Any
Manual-include (physical)	• Virtual • Physical, if it is a subdirectory of the manual-include directory	Files in manual-include directory or one of its subdirectories
Auto-include (physical)	None	None

For more information, see “About project folders” on page 122.

Adding folders to a project

You can add a folder to a project. Some restrictions apply, depending on the type of folder that you are adding, and where you are adding it. For more information, see "Populating a project" on page 125.

To add a virtual folder:

1 On the Projects resource tab, right-click the parent of the new folder (the project root or a virtual folder). Select Add Folder.

2 In the Add a Project Folder dialog box, select Virtual Folder.

3 In the Folder Name box, enter a name.

4 Click OK.

The virtual folder appears in the project folders pane. To populate the folder, see "Managing files in a project" on page 127.

To add a manual-include physical folder:

1 On the Projects resource tab, right-click the parent of the new folder (the project root or a folder). Select Add Folder.

2 In the Add a Project Folder dialog box, select Physical Folder.

3 In the Directory Path box, click the Browse icon and specify the path of the folder.

4 (Optional) In the Folder Name box, modify the name.

5 Click OK.

6 In the Populate New Folder dialog box, select any of the following options:

- **All files in the selected directory path** to add every file in the physical folder to the project
- **All subfolders under the selected directory path** to add every folder and subfolder in the physical folder to the project

Select both options to add every file, in the physical folder and in its subfolders, to the project. Note that a manual-include folder does not remain in sync with the physical folder after it is created; if you need this, add an auto-include folder.

7 Click OK.

The manual-include folder appears in the project folders pane, and it is populated according to the options that you set.

To add an auto-include physical folder:

1 On the Projects resource tab, right-click the parent of the new folder (the project root or a folder). Select Add Folder.

2 In the Add a Project Folder dialog box, select Physical Folder.

3 In the Directory Path box, click the Browse icon and specify the path of the folder.

4 (Optional) In the Folder Name box, modify the name.

5 Select Auto Include Files Using Filter. In the drop-down box, do one of the following:
   - Select All files.
   - Select a filter from the list.
   - Create a filter by entering the file extension of each file type to include, separated by semicolons; for example `htm;html;css;png;gif;jpg;jpeg`.

6 (Optional) To include the folders and subfolders in the physical folder, select Auto Include SubFolders.

7 Click OK.

The auto-include folder appears in the project folders pane, and it is populated according to the options that you set.

Managing folders in a project

You can edit the properties of a folder in a project, or remove it from the project.

To edit a folder:

1 On the Projects resource tab, right-click a folder and select Properties.

2 (Optional) In the Edit Folder Properties dialog box, change the information.

For more information, see "Setting project options" on page 124.

3 Click OK.

To remove a folder:

1 On the Projects resource tab, right-click a folder and select Remove Folder.

2 In the Warning dialog box, click Yes.

HomeSite+ for Dreamweaver MX removes the folder from the project folders pane, and removes the reference to the folder in the project's APF file. This does not delete the folder in your file system.

Managing files in a project

You can add a file to a virtual or manual-include folder, and remove it. For an auto-include folder, you can change the types of files that are added to it.

To add files to a manual-include or virtual folder:

1 On the Projects resource tab, right-click a folder and select Add Files to Folder.

2 In the Add Files to Folder dialog box, browse to the files and select them.

You can select multiple files by holding down the Shift or Ctrl key.

3 Click Add.

The files appear in the file pane.

Tip
You also can drag files from the bottom pane of the Projects tab to project folders.

To remove a file from a manual-include or virtual folder:

1 On the Projects resource tab, select a folder.

2 In the file pane, right-click a file and select Remove from this folder.

3 In the Confirm dialog box, click Yes.

HomeSite+ for Dreamweaver MX removes the file from the Projects resource tab, and removes the reference to the file in the project's APF file. This does not delete the file in your file system.

To change the types of files that are added to an auto-include folder:

1 On the Projects resource tab, right-click a folder and select Properties.

2 In the Edit Folder Properties dialog box, in the Auto Include Files Using Filter drop-down box, do one of the following:

- Select All files
- Select a list of file types
- Enter a list of file types, separated by semi-colons; for example, `htm;html;css;png;gif;jpg;jpeg`.

3 Click OK.

HomeSite+ for Dreamweaver MX adds and removes files to the folder according to the option that you set.

Working with a project

The Projects resource tab contains every user interface element that you need to work with a project. Within a project, you can perform all standard file management operations, set and modify resource filters, and perform other project-level tasks, such as link verification, search and replace, and deployment.

Using the Projects tab

In the Project resource tab, you can do the following:

- Select a project from the drop-down project list at the top of the tab
- Below the projects list, in the folders and files panes, navigate the project files
- Use the commands on the project toolbar and in the right-click menu

Managing project files

This section contains instructions for opening and closing a project and project files, editing a project's properties, and deleting a project.

For information on deploying a project and managing a project under source control, see "Performing other project-level tasks" on page 131.

To open a project:

- At the top of the Projects resource tab, in the drop-down list, select a project.

The project structure appears on the Projects tab. you can expand the project node to display the project's folders and resources.

To open multiple project documents:

- On the Projects resource tab, do one of the following:
  - Right-click the project root and select Open All Documents in Project.
  - Right click a folder and select Open All Documents in Folder.

Every text-based file in the project or folder displays in the Editor, with a tab for each file. For files that are not text-based, you might be asked to clear an option.

To edit properties within a project:

1 On the Projects resource tab, select a project from the drop-down list.

2 Right-click a project level and select Properties.

The following table shows the properties for each project level:

Project level	Properties
Project root (blue globe)	Path options for the project structure Can also be opened from **Projects > Properties**
Project root folder (top folder in project tree)	Project folder type, options for auto include folders, and deployment options
Resource folder (under Resources)	Resource name and filter

3 (Optional) In the Edit Project Properties dialog box appears, edit properties.

4 Click OK.

To close the project:

- Select **Project > Close Project**.

To delete a project:

1 On the Projects resource tab, right-click the project and select Delete Project.

2 In the confirmation message box, click Yes.

HomeSite+ for Dreamweaver MX removes the project from the Projects tab and deletes the APF file. This does not delete or affect the actual files that were included in the project.

Managing project resources

The Resources tree provides a view of the files in your project based on file type. This is useful for isolating application files, media files, and other content.

By default, the Resources tree includes these resources: HTML, CFML, and Images. You can add and edit the default resources for a project.

Note
To prevent slow execution of project link verification and extended search and replace operations, HomeSite+ for Dreamweaver MX installs a list of file extensions that it excludes from these operations in the Windows Registry, in the LinkVerifyExcludeExts key. These excluded files are large binary files such as EXE, PDF, ZIP, and media file types.

To view resources:

1 In the top pane of the Projects tab, expand the Resources node.

2 Click a resource type.

The files of the selected resource type appear in the lower pane.

To set the default resources for a project:

1 In the **Options** > **Settings** > **Projects** pane, select Include project resources.

2 In the resource table, select every resource type to include in new projects.

3 (Optional) Add a resource, as follows:

   a Click Add.

   b In the Add Project Resource dialog box, in the Resource Name box, enter a name for the resource.

   c In the Resource Filter box, enter one or more file extensions associated with the resource, separated by semicolons; for example, `gif;jpg;jpeg;png`.

   d Click OK.

   The resource is added to the resource table.

4 (Optional) Edit the resource name or resource filter (file extensions), as follows:

   a Click Edit.

   b In the Edit Project Resource dialog box, change the resource name or filter.

   c Click OK.

5 (Optional) Remove the resource from the list of default project resources, as follows: Selecting the resource in the resource table and click Delete.

The resource is removed immediately without asking you for a confirmation.

6 Click Apply.

To add a resource for a project:

1 On the Projects resource tab, in the project folder pane, right-click Resources and select Add Resource.

2 In the Add Resource Folder dialog box, in the Resource Name box, enter a name for the resource.

3 In the Resource Filter box, enter one or more file extensions to associate with the resource, separating file extensions with semicolons; for example, `gif;jpg;jpeg;png`.

4 Click OK.

To edit a resource in a project:

1 On the Projects resource tab, in the project folder pane, right-click the resource and select Properties.

2 In the Edit Resource Folder dialog box, edit the resource name or resource filter and click OK.

To remove a resource from a project:

- On the Projects resource tab, in the project folder pane, right-click the resource and select Remove Resource.

Performing other project-level tasks

The following table lists other project-level tasks that you can perform:

Task	See
Verifying links in a project	“Verifying links” on page 169
Using Find and Replace for a project	“Searching documents” on page 158
Working in a source control system	“Adding a project to source control” on page 132
Deploying a project	“Setting project-level deployment” on page 146

Adding a project to source control

With HomeSite+ for Dreamweaver MX projects, you can manage project files in a source control system.

You can establish a relationship between a source control application and your project, so that you can manage check-out, check-in, and other source control operations without opening the source control application.

Why use source control?

A source control system controls file management during application development.

Using source control for website content is essential when you need to coordinate the work of a development team in a complex project.

Source control enables team members to do the following:

- Share files on a LAN without overwriting the work of others
- Track versions of files, as files are modified
- Control the deployment of an application to customers, and also to employees for development and testing purposes

Supported source control systems

HomeSite+ for Dreamweaver MX uses the Microsoft Source Code Control (SCC) API to connect to standard source control applications. The SCC API works with client-based and server-based systems.

HomeSite+ for Dreamweaver MX automatically generates a list of source control applications that it detects on your computer. When you first select the Choose Source Control Provider command for a project, you can select a source control application from the list.

The user interface and command structure for source control applications varies from vendor to vendor, so check your software documentation.

Setting up a project in source control

There are three basic steps to establish a relationship between your HomeSite+ for Dreamweaver MX project and your source control application. The exact procedure differs, depending on your source control application.

1 Make sure that your source control application is installed on your computer and that it complies with the Source Code Control (SCC) API.

Otherwise, HomeSite+ for Dreamweaver MX cannot detect it.

2 Choose the source control application to use for your project.

To do this, right-click on the project root and select **Source Control > Choose Source Control Provider**. Select a provider and click OK.

3 Map your project to the directory that your source control application uses for operations such as checking files in and out.

To do this, right-click on the project root and select **Source Control > Map Project to Source Control**.

For more information, see Knowledge Base article http://www.allaire.com/Handlers/index.cfm?ID=14802&Method=Full14802/a.

Integrating a project with Microsoft Visual SourceSafe

The following procedure explains how to create a Microsoft Visual SourceSafe® (VSS) project, and add HomeSite+ for Dreamweaver MX project files to VSS source control. You can adapt this procedure to work in your source control system.

Visual SourceSafe uses the term **project** to refer to a distinct set of files stored in its database. This procedure uses the term **VSS project** to distinguish it from a HomeSite+ for Dreamweaver MX project.

To set up a project in Visual SourceSafe (VSS):

1 In VSS, create a project.

For more information, see the VSS documentation.

2 Select **File > Set Working Folder**, and set the working folder to be the root folder for your HomeSite+ for Dreamweaver MX project.

3 In HomeSite+ for Dreamweaver MX, open the project that you want to add to source control.

4 In the Projects tab, right-click the project root and select **Source Control > Choose Source Control Provider**.

The Choose Source Control Provider dialog box appears, listing every source control application detected on your computer.

5 Select Microsoft Visual SourceSafe and click OK.

6 Right-click the project root and select **Source Control > Map Project To Source Control**.

You might be required to login to VSS.

7 In VSS, select **File > Add Files**, and add the files and folders in your HomeSite+ for Dreamweaver MX project to the VSS project folder.

8 (Optional) In VSS, select **Tools > Options**, and set file handling rules, views, and other options.

The relationship between the source control application and your HomeSite+ for Dreamweaver MX project is established.

To re-establish this connection in future HomeSite+ for Dreamweaver MX sessions, open the project that you mapped to the source control application.

Using source control in HomeSite+ for Dreamweaver MX

You can control your source from within HomeSite+ for Dreamweaver MX or, if you must work directly in the source control application, you can open it from HomeSite+ for Dreamweaver MX.

To control your source from within HomeSite+ for Dreamweaver MX:

- Right-click project files and folders on the Projects tab, and select from the commands in the context-sensitive menu.

To open the source control application:

- In the Source Control QuickBar, click Source Control Application.

Displaying the Source Control toolbar

The Source Control toolbar contains toolbuttons for working with a version source control system, for example to check in, check out, and add to source control. The toolbar commands are applied to the current document in the Document window.

You can also access the toolbar commands by right-clicking in the Projects pane.

Note
You cannot add a file to source control unless it is located in a physical project folder (auto-include or manual-include) for a project that has been mapped to a source control application.

To display the source control toolbar:

1 Right-click in the QuickBar and select Source Control.

2 Drag the floating Source Control toolbar onto the QuickBar.

If the options on the toolbar are disabled, open a project that is mapped to the source control application. This establishes a connection for the duration of your session.

Sharing project files in Visual SourceSafe

For information on setting up multiple user access to Visual SourceSafe files, see Knowledge Base article http://www.allaire.com/Handlers/index.cfm?ID=16754&Method=Full16754/a.

Chapter 10

Debugging Application Code

When debugging a basic page, you can use the debugging information that is reported from the application server. However, for complex development tasks, you need a robust and interactive debugging program.

The interactive debugger runs against dynamic pages on the ColdFusion Server. You can set breakpoints and step through the code to isolate and correct problems.

A tabbed debug window provides breakpoints, variables, recordsets, tag and page hierarchies, and page output. From these panes, you can set watches; manage breakpoints; and evaluate variables and expressions.

Contents

- Overview of the Interactive Debugger 136
- Setting up a debugging session 137
- Using the Debugger 140

Overview of the Interactive Debugger

You can run the debugger against your application pages to find errors in your code, when Remote Development Services (RDS) is enabled and a server mapping is defined. Debugging is not supported on Windows 98.

The debugger lets you perform these tasks:

- Set breakpoints and watches
- Evaluate variables and expressions
- Step through lines of code
- Investigate the code stack
- Monitor recordsets
- Observe variables in all scopes

To run debugging processes, use the Debug menu or the Debug toolbar.

Setting up a debugging session

These are the tasks that you perform to set up a debugging session:

- Select or configure a RDS server against which to run a debugging session.
- Specify server, file, and browser mappings so that HomeSite+ for Dreamweaver MX can locate the ColdFusion development server and the files to debug.

The following procedure describes these tasks. For more information, see "Working with files on remote servers" on page 14.

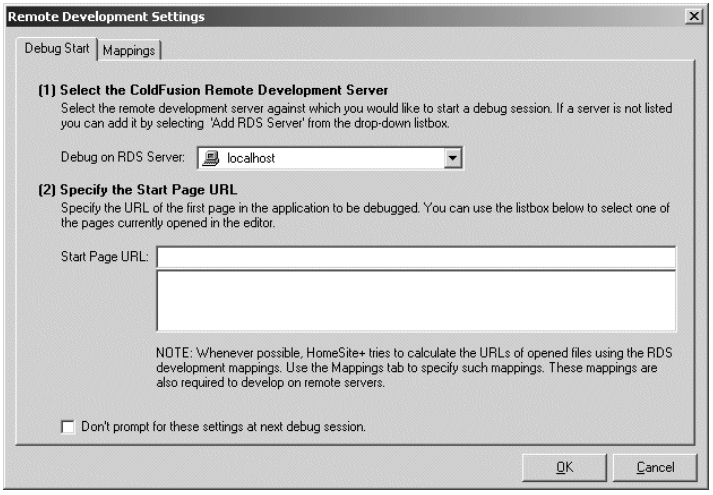
To set up a debugging session:

1 In the **Options > Settings > File Settings > FTP & RDS** pane, select the Enable Explorer shell integration option, if it is not already selected.

2 Open an application page.

3 Select **Debug > Debug Settings**.

The Remote Development Settings dialog box appears:

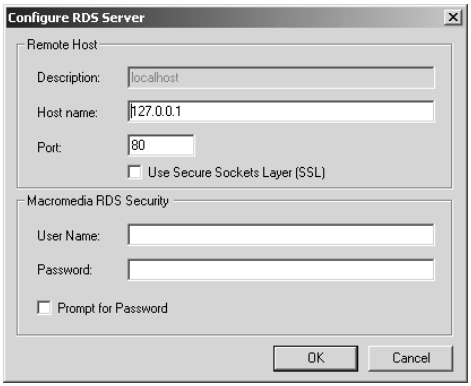


4 If ColdFusion Server is installed on the same computer as HomeSite+ for Dreamweaver MX, select the server in the Debug on RDS Server box, on the Debug Start tab.

5 If ColdFusion Server is *not* installed on the same computer as HomeSite+ for Dreamweaver MX, perform the following steps:

a On the Debug Start tab, in the Debug on RDS Server box, select Add RDS Server.

The Configure RDS Server dialog box appears:



b In the Remote Host box, enter the RDS server description, host name, and port.

(Optional) Select the Use Secure Sockets Layer (SSL) option.

c In the Macromedia RDS Security box: if RDS Security is implemented on your system, enter a username and password to access the server. If you do not have this information, ask your network administrator.

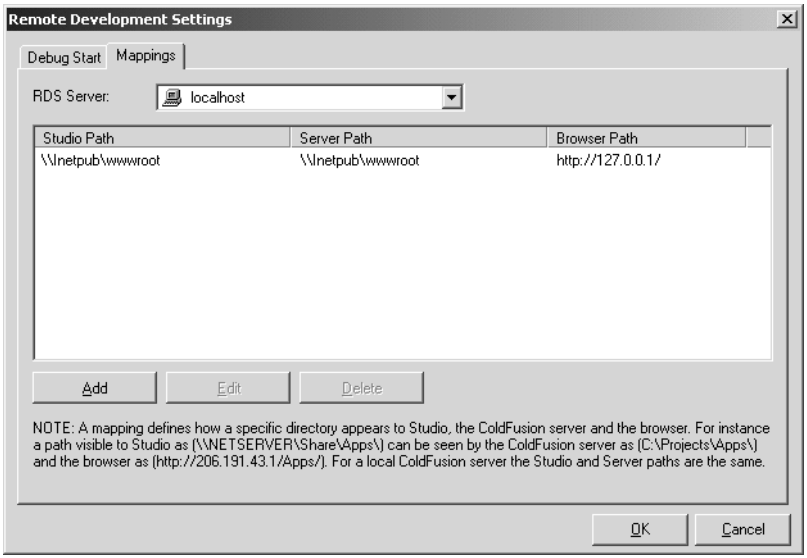
(Optional) Select the Prompt for Password option.

d Click OK.

The Remote Development Settings dialog box displays the description and start page URL of the RDS server.

6 (Optional) Select the option called Don't prompt for these settings at next debug session.

7 Click the Mappings tab.



8 Complete the fields as follows:

- **RDS Server** Select a RDS server against which to run the debugging session.
- **HomeSite+ Path** Enter or browse to the path that HomeSite+ for Dreamweaver MX uses to locate the page you are debugging.
- **ColdFusion Server**

  If you debug against a local server, enter the same path as for HomeSite+ for Dreamweaver MX.

  If you debug against a remote server, enter its absolute path. The path must be the same as the alias or virtual mapping that your web server uses.
- **Browser Path** Enter the browser path, or URL, of the application.

Click Add.

9 For each debugging mapping, repeat the previous step.

10 Click OK.

The Remote Development Settings dialog box closes

You can begin debugging in the Editor.

Using the Debugger

Running the interactive debugger helps you find problems in code by tracing how ColdFusion evaluates a page, step by step. You place one or more breakpoints in a page to pause its execution, then step through the code, checking the values of variables and expressions as you go, to ensure that t he code executes as you expect.

When you have configured RDS and file mappings, you can use the commands on the Debug menu and the Debug toolbar to run the debugger.

You can undock the Debug toolbar by double-clicking on the undock bars on the left-hand side of the toolbar, or you can drop the toolbar into the QuickBar.

The Debug Start dialog box displays each time you click Start on the Debugger toolbar. To disable the display of this box, select the option on the check Debug Start pane, the next time you click Run.

Debug breakpoint lines are shaded red. To turn breakpoints on and off, select **Debug > Toggle Breakpoints**. To delete all breakpoints in the current document, select **Debug > Clear All Breakpoints**.

You cannot debug encrypted CFML templates.

About the Debugger toolbar

This toolbar displays by default below the Resources window, but you can drag it to any other location in the workspace. The toolbuttons let you quickly start, manage, and complete a debugging session.

Running the Debugger

To run the interactive debugger:

1 Open an application page.

The Debug toolbar displays at the bottom of the application window.

2 In the application page, set one or more breakpoints in your code, by clicking in the gutter on the left side of the Editor.

3 In the Debug toolbar, click the Start/Continue tool.

The Remote Development Settings dialog box displays the default local server.

4 Enter the Start URL, which is the page URL. Click OK.

The list box displays the pages that are open in HomeSite+ for Dreamweaver MX.

Enter a fully qualified file path relative to your local ColdFusion Server; for example, http://127.0.0.1/SomeServerPath/index.cfm.

HomeSite+ for Dreamweaver MX calculates URLs of open file using the RDS development mappings (which you set in the Mappings tab.) The mappings are also required to develop code on remote servers.

Based on the RDS Security configuration for your site, a login prompt for accessing protected resources displays.

5 Enter a user name and password, if necessary.

The debugger starts. It forwards your URL to the Browse view. When the ColdFusion Server encounters a breakpoint, a blue bar displays at the breakpoint in Edit view. ColdFusion Server pauses until you click Start/Continue. The server processes the code to the next breakpoint.

When you click Start/Continue after the last breakpoint, ColdFusion executes the page and outputs it to the browser.

6 To stop the debugging session, click End.

The debugger remains active until you select **Debug > End**.

About the Debug window

To open the Debug window, select **View > Debug Window**. These are the panes in the Debug window:

Pane	Purpose
Variables	Displays all scopes of local variables
Watches	Lets you set watches and evaluate expressions and variables
Recordsets	Displays the list of recordsets initialized in the current application page. Tracks CFQUERY-based database recordsets and dynamic recordsets generated programmatically
Output	Shows the output of a page as it is being generated
Breakpoints	Shows the breakpoints that you have set in all files. Lets you view, disable, edit conditions, and remove breakpoints
Tag Stack	Shows a hierarchy of tag and page attributes and values

Each pane has an associated toolbutton on the Debug toolbar.

You can undock panes individually, so that, for example, you can display Breakpoints while you're displaying Watches.

Debugging across multiple pages

The debugger is active after a page is loaded, until you click End on the Debug toolbar or select **Debug > End**. This lets you debug applications across multiple

HTML and CFML pages. For example, you can test the submittal of an HTML form and its subsequent processing by a ColdFusion application page.

Stepping through code

To execute code in debug mode, you use these tools on the Debug toolbar:

- **Step Into** Proceeds to the next unit of execution in your code. Use this command to step through code line-by-line.

  If the next step is inside an included file or CFX, the debugger steps into the file.
- **Step Over** Same as Step Into, except that it executes included code but does not trace through included code step-by-step. Does not step in included files (CFINCLUDE, CFMODULE, or CFLOCATION) or custom tags.
- **Step Out** Steps back to the location in the original page where you entered included code.
- **Run to Cursor** Executes to the cursor position; no breakpoint required. The cursor location must be below the current position. If there are breakpoints between the current position and the cursor, Run to Cursor stops at them.

Evaluating an expression and setting a watch

To evaluate an arbitrary expression, when the debugger is suspended at a breakpoint, you use the evaluator box at the top of the Watches pane in the Debug window. Use the evaluator when you want to know how an expression evaluates as you step through code.

A watch lets you evaluate the same expression or variable each time you stop execution. When you set a watch, the debugger evaluates the watched expression. If the watched expression's value changes, a hand icon displays.

You can use the evaluator to change values of variables, create new variables, or to insert ColdFusion functions in your expressions.

To set a watch:

1 Select **Debug > Watches** or click Watches on the Debug toolbar. The Watches pane displays.

2 Cut and paste an expression or variable into the list box at the top of the pane.

3 To find the value of the expression at the next breakpoint or line where the Debugger stops, select Evaluate.

4 To add an expression in the evaluator list box to the list of watched expressions, select Watch.

  The Watch area shows the values of watched expressions and any error messages in resolving these parameters.

5 To continue debugging, click Start/Continue.

6 Click End. Debugging stops.

Chapter 11

Deploying Files

Deployment is the process of copying all of the files in your project to one or more host servers. You can specify a deployment path for the entire project or for individual folders and files.

This chapter describes how to set default deployment options and how to deploy files to a server.

Contents

- Setting default deployment options 144
- Deploying a single file or folder 145
- Performing a custom deployment 146
- Saving deployment results..... 153

Setting default deployment options

Optionally, you can specify options to apply for every deployment by default.

To set other project deployment options:

1 In the **Options > Settings > Projects > Deployment** pane, specify logging options:

- If you do not want to create a log file tracking a project's deployment, select Disable Logging.
- To enable logging, clear Disable Logging and, in the Log File box, enter an absolute file path to a LOG file.

2 Set the default deployment options.

The following table describes each option:

Option	Result when selected
Create Missing Folders/ Directories	Makes the directory structure on the server match the directory structure on the client. If this is not selected and you add a new directory to the client, then when you try to deploy you will receive an error.
Upload Only Newer	Compares the files on the client to the files that are already on the deployment server, and copies only the updated files from the client to the server.
Encrypt CFML	Prevents end-users of your web application from seeing and taking your CFML code.
Force to Lower Case	Changes all filenames to be lowercase, but does *not* lowercase the link references to your files. So selecting this option can cause your links to break on a UNIX server. For example, the following link would break: `<a href="ABCs.htm">ABC's</a>`.
Show Deployment Warning Dialog	Displays a warning before deploying your files, asking you to confirm that you are ready to deploy. This dialog box also has an option to not show the warning again.

3 Click Apply.

Deploying a single file or folder

This section explains how to deploy a single folder or file to one or more servers.

To deploy to a single location:

1 On the Projects resource tab, open a project.

For more information, see "Managing project files" on page 129.

2 Do one of the following:

- Right-click a file and select Deploy File.
- Right-click a folder and select Deploy Files in Folder.

A dialog box appears with the message "Are you deploying to multiple servers?"

3 Click No.

The Specify Deployment Location dialog box appears, with Absolute Deployment Path selected and the cursor positioned in its Deployment Path box.

4 Browse to the directory where you must deploy the file or folder, or enter the absolute path in the Deployment Path box. Click OK.

The Deployment Warning dialog box appears, listing the file or files selected for deployment. You can select Do Not Show this Dialog Again if you do not want a confirmation to appear for future deployments.

5 Click Deploy.

To deploy to multiple servers:

1 On the Projects resource tab, open a project.

For more information, see "Managing project files" on page 129.

2 Do one of the following:

- Right-click a file and select Deploy File.
- Right-click a folder and select Deploy Files in Folder.

A dialog box appears with the message "Are you deploying to multiple servers?"

3 Click Yes.

The Dynamic Deployment Location dialog box appears with the cursor positioned in the Deployment Path box.

4 Browse to a server under Macromedia FTP & RDS, or enter the absolute path (minus drive letter or server name) in the Deployment Path box. Click OK.

5 In the Select Deployment Servers dialog box, select servers and click OK. If you do not see a server that you need, see "Adding a deployment server" on page 148.

The Deployment Warning dialog box appears, listing the file or files selected for deployment. You can select Do Not Show this Dialog Again if you do not want a confirmation to appear for future deployments.

6 Click Deploy.

Performing a custom deployment

This section describes the advanced method of deployment that offers maximum flexibility and functionality. This method involves building an absolute path for the product to deploy the files. This absolute path is comprised of three parts:

Server + **Project Path** + **Folder Level**

To build this deployment path, and to deploy, you must do the following steps:

1 Select the folders and files to deploy.

2 Add deployment servers.

3 Run the Deployment Wizard, with or without scripts.

The rest of this section describes each of these steps.

Selecting folders and files to deploy

You can deploy a whole project or select specific files to deploy.

Setting project-level deployment

To set the deployment for a project, enter an absolute path for the location.

Note
The Relative to the Parent Folder Deployment Location option is not available at the project level because the project folder functions as the root, and it has no parents. The Do Not Deploy option is not available at the project level, because then no files would be deployed.

To set deployment for a project:

1 On the Projects resource tab, open a project.

For more information, see “Managing project files” on page 129.

2 Right-click the project root and select Properties.

The Edit Project Properties dialog box appears.

3 In the Deployment Path box, browse to the location to which you must deploy, or enter the absolute path.

For more information, see “About deployment options” on page 148.

4 Click OK.

Setting folder-level deployment

The folder level of deployment allows more flexibility, but requires more steps.

To set deployment for a folder:

1 On the Projects resource tab, open a project.

For more information, see "Managing project files" on page 129.

2 Right-click a folder and select Properties.

The Edit Folder Properties dialog box appears.

3 Click the Deployment tab and select a deployment option.

For more information, see "About deployment options" on page 148.

4 If you selected Dynamic Deployment Location, in the Deployment Path box, browse to the deployment location.

If you must enter the absolute path, follow these rules:

- Include the drive letter and use backslashes for pathnames to local and network drives; for example, C:\Directory\Subdirectory or \\Directory\Subdirectory.
- Use forward slashes for pathnames to remote servers.
- Use drive letters for RDS servers; for example, C:/Directory/Subdirectory.
- Do not use drive letters for FTP servers; for example, use Directory/Subdirectory.

Note
Typically, you do not include the server name for the host in the pathname. When you select a deployment server in the Deployment Wizard, HomeSite+ for Dreamweaver MX appends the server name to the path. If you do enter a server name as part of the path, and do not select the server in the Deployment Wizard, an error appears in the Results pane.

5 Click OK.

Your deployment settings are applied to the folder.

The project deployment information is written to the project APF file. If you deploy the APF file along with the project, you maintain the same deployment settings. For more information, see "About the project file" on page 123.

About deployment options

The following table describes the deployment options on the Deployment tab of the Edit Folder Properties dialog box:

Option	When selected, you can...
Relative to the Parent Folder Deployment Location	Deploy a folder to a location that is relative to its parent folder or project. For example, if you deploy a folder called Parent to ServerDirectory/Parent, then its child folder, called Child, is deployed to ServerDirectory/Parent/Child. The product calculates child paths for you automatically.
Dynamic Deployment Location	Specify the path to which HomeSite+ for Dreamweaver MX deploys the file or folder, on one or more servers.
Absolute Deployment Path	Specify the path *and server* to which HomeSite+ for Dreamweaver MX deploys the file or folder. You cannot use this option to deploy to more than one server.
Do Not Deploy	Tell HomeSite+ for Dreamweaver MX to ignore the folder and its contents during deployment.

Adding a deployment server

Adding a deployment server is similar to configuring a remote server in the Files panel. The primary difference is that deployment server information is added to the project file, while remote server information is stored in the Windows Registry.

So, even if you already added a remote server using Macromedia FTP & RDS, you must enter the same server information to configure the server for deployment. For a full description of configuration options for FTP and RDS servers, see "Working with files on remote servers" on page 14.

To add a deployment server:

1 On the Projects resource tab, open a project.

For more information, see "Managing project files" on page 129.

2 Right-click the Deployment Servers node and select Add FTP Server or Add RDS Server.

3 Complete the server configuration dialog box (Configure FTP Server or Configure RDS Server).

For instructions, see "Working with files on remote servers" on page 14.

4 Click OK to save the server information.

The server is listed under Deployment Servers on the Projects tab. You select servers from this list when you deploy your project.

Note
When you deploy to multiple servers, the project files are copied to identical locations on each server. Make sure the servers contain the same directory structure.

To view the deployment servers for a project:

1 On the Projects resource tab, open a project.

2 In the top pane of the tab, expand the Deployment Servers node.

To edit properties for a deployment server:

1 Expand the Deployment Servers node in the top pane of the Projects tab.

2 Right-click the server whose properties you want to change and select Properties from the popup menu.

3 Make changes in the Configure FTP Server or Configure RDS Server dialog box.

4 When you are satisfied, click OK to save your changes.

To remove a deployment server for a project:

1 Expand the Deployment Servers node in the top pane of the Projects tab.

2 Right-click the server to remove and select Remove Server from the popup menu.

3 Click Yes to confirm that you want to remove the server.

Running the Deployment Wizard

You can use the Deployment Wizard to accomplish the following tasks:

- Deploy a project directly to one or to multiple host servers
- Create a deployment script

The advantage of scripts is that you can schedule them and have them perform additional tasks, such as file compression and custom logging.

When you deploy to multiple servers, the project files are copied to identical locations on each server, for example C:/Web/MyApps. So unless you are deploying to the root directory, make sure that the path to the intended deployment folder is the same on each server.

The Deployment Wizard automatically selects your deployment servers in the server list. Any remote servers that have been mapped to in the Files panel are also included in this list. You can deploy your project to any of these servers; however, the APF file for the project only includes information for the deployment servers. For more information, see “About the project file” on page 123.

Deploying directly

When you deploy a project directly, you are copying the files to the host server(s) at the time of deployment. You can deploy to the same computer that you are currently using (localhost), or you can deploy to one or more remote servers.

To deploy a project:

1 On the Projects resource tab, open a project.

For more information, see "Managing project files" on page 129.

2 Select **Projects > Deployment Wizard**.

3 In the first pane of the Deployment Wizard, click Direct Deployment and click Next. The second pane displays.

4 Specify options for deploying to local host or to one or more remote servers:

- To deploy to local host, select Local/Network Deployment, set file handling options, and click Next.
- To deploy to one or more remote servers, click Remote RDS/FTP Deployment and click Next.

  In the third pane, highlight entries in the Select Remote Deployment Server list if you must deploy to defined servers other than those already associated with the project. Click Next.

5 Click Finish.

The Results window Deployment tab opens and shows the progress of the file transfer. When deployment is complete, the Results window also displays the status of each file and the amount of time required for the deployment.

Using a deployment script

You can create a deployment script to run at any time, and you can modify a script with custom code after it has been generated.

About deployment scripts

Deployment scripts are based on the Visual Tools Object Model (VTOM), and are generated in JScript or VBScript. For more information, see "Scripting the Visual Tools Object Model" on page 201.

You can generate three types of deployment scripts:

- **Project-wide upload script** Copies the files in a project to the host server(s), just as if you were deploying the project directly. During a project-wide upload, the project is opened, uploaded, then closed.

  You can only add custom code before and after deployment.

- **File-by-file deployment script** Uploads each of the project files individually. During a file-by-file upload, the project is opened, each file is uploaded individually, and the project is closed. A file must be in a project to be uploaded.

You can edit the script to deploy only a specified set of files, and you can add custom code to run during deployment.

- **Project element iterator script** Performs a file-by-file upload with a set of nested loops, or iterations. The script iterates through each server, then each folder, then each file. This script runs independent of specific servers or folders.

  Programmers can insert code in between the loops to add custom functionality during deployment; you can change anything for a project except its name.

To create a deployment script:

1 On the Projects resource tab, open a project.

  For more information, see "Managing project files" on page 129.

2 Click Deploy Project or select **Project > Deployment Wizard**.

  The Deployment Wizard window appears.

3 Select Scriptable Deployment and click Next.

4 In the Deployment Task Name box, enter a name for the script.

5 In the Script Language box, select VBScript or JScript.

6 In the Save Script to File box, modify the filename that HomeSite+ for Dreamweaver MX entered based on the project name, or click Browse to a enter the filename using the Save As dialog box.

7 (Optional) To see the script in the Editor, select the Open script in editor after generation option.

8 Click Next.

9 Select the type of script that you want to generate, and click Next.

  For more information, see "About deployment scripts" on page 150.

10 Select destination, file, and logging options:

  - To deploy to a local or network path, click Local/Network Deployment and click Next.
  - To deploy to one or more remote servers, click Remote RDS/FTP Deployment, click Next, select the server or servers to deploy to, and click Next.

11 Click Finish.

HomeSite+ for Dreamweaver MX generates the script.

Managing deployment scripts

To edit a deployment script:

1 On the Projects resource tab, in the top pane, expand the Deployment Scripts node.

2 Right-click a script and select Open Script.

3 Edit the script in the Editor and save your work.

To run a deployment script:

1 On the Projects resource tab, in the top pane, expand the Deployment Scripts node.

2 Right-click a script and select Run Script.

The Results pane opens and displays the status of your deployment.

To view a deployment script's properties:

1 On the Projects resource tab, in the top pane, expand the Deployment Scripts node.

2 Right-click a script and select Properties.

A dialog box listing the script properties appears.

3 (Optional) Enter a new description for the script and click OK.

To remove a deployment script:

1 On the Projects resource tab, in the top pane, expand the Deployment Scripts node.

2 Right-click a script and select Remove Script.

3 Click Yes to confirm.

The script disappears from the Deployment Scripts list.

Saving deployment results

Regardless of your deployment method, you can review and save the results. The results can be saved in a log file and/or in HTML format.

To review the results of the last deployment:

1 Select **View > Results > Deployment**.

2 To select display options, right-click in the Deployment pane and select the appropriate command.

To save the results in a log file:

1 Open the **Options > Settings > Projects > Deployment** pane.

2 In the Logging box, clear Disable Logging.

3 (Optional) In the Log File box, specify a new location for the log file.

By default, results are written to the deployment.log file in the root directory.

4 Click Apply.

With each succeeding deployment, the information from the Results window, including a date/time stamp for each deployed file, is appended to the log file. This file can grow considerably over time, so you should reduce its size as needed.

To save the results in HTML format:

- Follow the instructions in “Saving results” on page 157.

Chapter 12

Testing and Maintaining Web Pages

As your website develops, you must update it and test its accuracy, completeness, and efficiency. HomeSite+ for Dreamweaver MX provides a full set of tools to accomplish these necessary tasks.

Contents

- Working in the Results window 156
- Searching documents 158
- Checking spelling 165
- Verifying links 169
- Using Site View to check page links 171
- Testing page download times 172

Working in the Results window

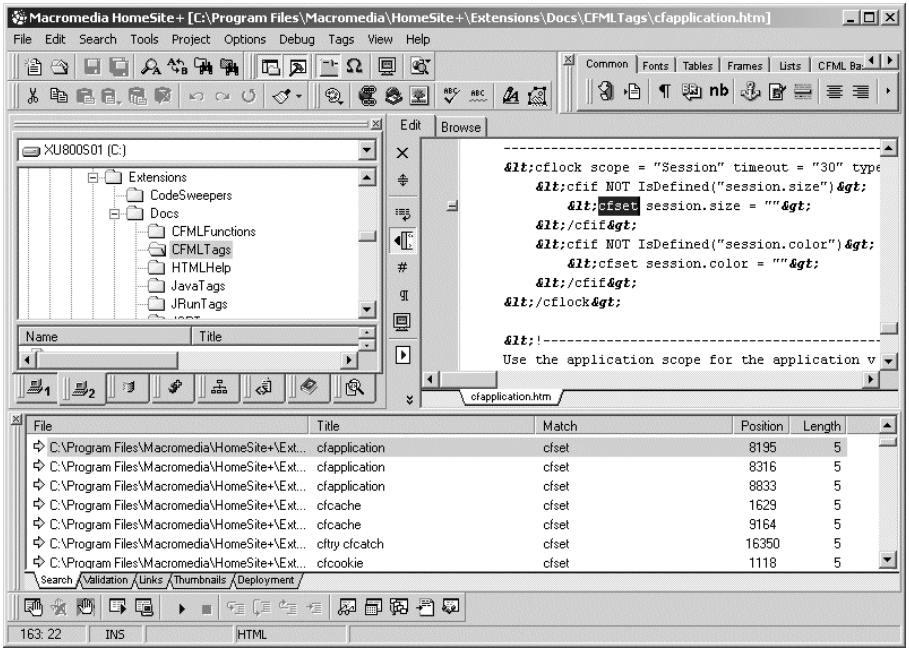
The Results window is a section of the user interface that you can display if you want, but otherwise it only appears when you need it—when you must see the results of an operation such as an extended search or link verification.

Opening the Results window

The Results window displays output from these operations:

- Extended Find or Extended Replace
- Validate code
- Verify links
- Display image thumbnails
- Deploy a project

The Results window displays the results of an operation on the appropriate tab; for example, search results appear on the Search tab.



You can manually open the Results window, or display specific tabs on the Results window. You can also open documents from search results; for details, see "Working with the results of extended search operations" on page 162.

To manually open the Results window:

1 Select **View > Results**.

2 To close the window, right-click in the Results window and select Close Results.

You can also press Shift+Ctrl+L to alternate between opening and closing the Results window.

To open specific tabs on the Results window:

- Press the Shift+Ctrl+F1 for the Search tab, Shift+Ctrl+F2 for the Validation tab, Shift+Ctrl+F3 for the Links tab, Shift+Ctrl+F4 for the Thumbnails tab, and Shift+Ctrl+F5 for the Deployment tab.

  You can also use these keyboard shortcuts to switch between tabs.

Saving results

When you run an operation that displays the Results window, HomeSite+ for Dreamweaver MX preserves the results of the operation on the appropriate tab, even when you move from one tab to another or close the Results window. However, results are discarded when you repeat the same operation or close HomeSite+ for Dreamweaver MX.

For example, you could run an extended search, verify the links in your project, close the Results window, and then run another extended search. In this case you would lose the results from the first search, but you could click between the Search and Links tabs to see the results of the second search and the link verification.

If you want to keep the results from an extended search, link verification, or code validation, save them before you close HomeSite+ for Dreamweaver MX.

To save the results as an HTML document:

1 Right-click in the Search, Validation, or Links tab and select Open in Browser.

The document displays in the default browser.

2 Use the commands available in your browser to save the source as an HTML document.

Searching documents

HomeSite+ for Dreamweaver MX provides basic and extended search capabilities. These enable you to find and replace alphanumeric strings—including regular expressions—across folders and projects, filter the files to search by file type, select how tags are processed by the search engine, automatically replace special and extended characters with their HTML equivalents, make a double-spaced document single-spaced, select documents to edit or browse from the search results, and more. This section describes each search capability.

Selecting search text

By selecting text in a document and invoking a search, HomeSite+ for Dreamweaver MX automatically inserts the selected text into the Find what text box.

In a basic search or replace, if the selected text exceeds 100 characters, no text is inserted into the Find what box, and HomeSite+ for Dreamweaver MX searches the selected text instead of the entire document. If you *want* to only replace text within a selection, you can select the Selection option in the Replace dialog box.

Optionally, you can configure HomeSite+ for Dreamweaver MX so that, when you do not select any search text, it selects the word nearest to the cursor position and inserts this word into the Find what box.

To enable nearest word search selection:

- Select **Options > Settings > Editor > Select nearest word for searches**.
  This enables the following behaviors in both basic and extended searches:
  - If no word is highlighted, the word closest to the current cursor position is inserted in the Find what text box.
  - If the cursor is on a blank line, the search default text will be the nearest word preceding the position of the cursor.

Saving search text

You can reuse search strings for basic and extended searches, but the method for saving the search text differs.

To save search text:

- In a basic search, search text is automatically saved.
- In an extended search, you can selectively save search text. Click the arrow next to the Find what box and select Save find text from the pop-up menu.

To use a saved search string in a basic search:

- Open the drop-down list for the Find what text box and select one of the last ten search strings.

To use a saved search string in an extended search:

1 In the Extended Find or Extended Replace dialog box, click the arrow next to the Find what box and select Open find text from the popup menu.

2 In the Open find text dialog box, select the search string to use and click Open.

Using basic search commands

You can perform a basic search or replace operation on the current document, even if it is an untitled, unsaved document.

To search the current document:

1 Select **Search > Find**.

2 In the Find dialog box, in the Find what box, enter the search text.

The last 10 items are saved in the Find what drop-down list. You can select from this list. You can also select text in the Editor to appear in the Find what box. For details, see "Selecting search text" on page 158.

3 (Optional) Select the Match whole words, Match case, and Direction options.

To set more advanced options, perform an extended search on the current document. For instructions, see "Performing an extended search" on page 160.

4 Click Find Next to sequentially highlight each match in the document.

5 If the search dialog box closes, you can press F3 to resume the search from the current cursor position in the document.

To replace text in the current document:

1 Select **Search > Replace**.

2 In the Replace dialog box, in the Find what and Replace with boxes, enter the text to find and the text to replace.

The last 10 items are saved in the Find what and Replace what drop-down lists. You can select from these lists. You can also select text in the Editor to appear in the Find what box. For details, see "Selecting search text" on page 158.

3 (Optional) Select the Match whole words, Match case, and Direction options.

To set more advanced options, perform an extended replace on the current document. For instructions, see "Performing an extended search" on page 160.

4 Do either of the following:

- Click Replace to replace the first highlighted match and to highlight the next match in the document.
- Click Replace All to replace all matches without reviewing them first.

Using extended search commands

You can use the Extended Find and Extended Replace commands to perform more complex search operations across multiple documents.

Extended search and extended replace operations include untitled, unsaved documents. These are listed in the Results window by their tab label in the Editor (Untitled1, Untitled2, and so on).

The rest of this section provides instructions for performing extended search and extended search and replace operations.

Performing an extended search

This section contains instructions for performing extended searches in the current document, all open documents, in a folder, or in a project.

To perform an extended search:

1 Select **Search > Extended Find** to display the Extended Find dialog box.

2 Enter the appropriate text in the Find what box.

To re-use a previously saved search string, see "Saving search text" on page 158.

3 In the Find where box, select one of the following options:

- **Current document** searches the current document only, using more advanced options than are available in a basic search.
- **All open documents** searches all open documents, even those that are not yet saved.
- **In folder** searches the documents in a specific folder. You can select a folder from the drop-down box, enter a full path, or browse to the folder.

  To search documents in the folder's subdirectories, select Include subfolders.

  If you want to limit your search to files of certain types, select a filter from the File Types drop-down box or enter your own; for example, ***.html;*.htm;*.txt**.
- **In project** searches the documents in a project. You can select a project from the drop-down box, enter the absolute path of a project, or browse to a project.

  If you want to limit your search to files of certain types, select a filter from the File Types drop-down box or enter your own; for example, ***.html;*.htm;*.txt**.

4 (Optional) Select any of the following options:

- **Match case** highlight a match only if it has identical uppercase and lowercase as the selection.
- **Regular expressions** enables parsing of regular expression entries.

  For information on the specific syntax to use for regular expressions in HomeSite+ for Dreamweaver MX, see "Using regular expressions" on page 91.

- **Skip tags while searching** searches only the page content, not the tags themselves—especially useful when searching for words like *width* or *value*.

  This option is not available when Regular expressions is selected.
- **Exclude Binary Files** skips binary files such as EXE, PDF, ZIP, and media file types. Searching these types of files would significantly slow down the search. HomeSite+ for Dreamweaver MX installs a list of excluded file extensions in the Windows Registry LinkVerifyExcludeExts key.
- **Display Line Info** displays the contents of the line in which the match was found. This slows down the performance in large searches.

5 Click Find.

6 To cancel the search operation, press the Esc key.

The Results window displays a list of documents matching the search string.

Performing an extended replace

This section contains instructions for performing an extended search and replace operation in the current document, all open documents, in a folder, or in a project.

Caution
An extended replace operation cannot be undone. For best results, select the Make backups option. Also, an extended replace operation skips all read-only files.

To perform an extended replace:

1 Select **Search > Extended Replace** to display the Extended Replace dialog box.

2 Enter the appropriate text in the Find what and Replace with boxes.

To re-use a previously saved search string in the Find what box, see "Saving search text" on page 158.

3 In the Find where box, select one of the following options:

- **Current document** Replaces text in the current document only, using more advanced options than are available in a basic search and replace operation.
- **All open documents** Replaces text in all open documents, even those that are not yet saved.
- **In folder** Replaces text in the documents of a specific folder. You can select a folder from the drop-down box, enter a full path, or browse to the folder.

  To replace text in documents that are in the folder's subdirectories, select Include subfolders.

  If you want to limit your search and replace operation to files of certain types, select a filter from the File Types drop-down box or enter your own; for example, *.html;*.htm;*.txt.

- **In project** Replaces text in the documents of a project. You can select a project from the drop-down box, enter the absolute path of a project, or browse to a project.

  If you want to limit your search and replace operation to files of certain types, select a filter from the File Types drop-down box or enter your own; for example, *.html;*.htm;*.txt.

4 Select any of the following options as needed:

- **Match case** Only highlights a match if it has identical uppercase and lowercase as the selection.
- **Regular expressions** Enables parsing of regular expression entries.

  For information on the specific syntax to use for regular expressions in HomeSite+ for Dreamweaver MX, see "Using regular expressions" on page 91.
- **Skip tags while searching** Searches only the page content, not the tags themselves—especially useful when searching for words like *width* or *value*.

  This option is not available when Regular expressions is selected.
- **Exclude Binary Files** Skips over binary files such as EXE, PDF, ZIP, and media file types. Searching these types of files would significantly slow down the search. HomeSite+ for Dreamweaver MX installs a list of excluded file extensions in the Windows Registry LinkVerifyExcludeExts key.
- **Display Line Info** Displays the contents of the line in which the match was found. This slows down the performance in large searches.
- **Make backups** Select this for folder and project replace operations, because an extended replace cannot be undone. Backs up every file in the folder or project before performing the extended replace operation.

  If you select this option, you must specify one of these backup locations:

  Backup directory Backs up in the \AutoBackup folder in the program root; for example, C:\Program Files\Macromedia\HomeSite+\AutoBackup.

  Original directory Backs up in the current directory. You can differentiate these files from your regular files by their name, since backed up files use the following naming convention: *filename + an incremented 3 digit number + the file extension*; for example, myfile000.htm.

  For information about how HomeSite+ for Dreamweaver MX determines the current directory with two Files tabs, see "About the Files tabs" on page 42.

5 Click Replace.

6 To cancel the search and replace operation, press the Esc key.

Working with the results of extended search operations

The Results window displays a list of documents in which text was replaced.

After an extended search, the Results window displays a list of documents in which the search string was found and/or replaced. You can work with these search results in many ways; for example you can edit or browse the documents in which matches were found, or view the search results list in a browser.

To edit a document for a match:

1 Double-click a match in the results list.

This opens the document in the Editor if it is not already open, and highlights the match in the document.

2 Edit the file as necessary, and then save and close it.

To edit documents for more than one match:

1 Highlight one or more matches in the search results list.

2 Right-click the selection and select Open in Editor.

This opens every document in which the selected matches were found, and highlights the selected matches in the documents.

3 Edit the files as necessary, and then save and close them.

To open the search results in a browser:

- Right-click in the search results pane and select Open in Browser.

This opens the search results list in the default external browser. You can print the list from the browser.

To clear the search results:

- Right-click in the search results pane and select Clear.

Replacing extended and special characters

You can replace special and extended characters in the current document with their HTML equivalents; for example, you can replace "&" with &.

To replace extended and special characters:

1 Open the document in which to make the replacements, if it is not already open.

2 Select **Search > Replace Extended Characters**.

Replacing double-spaced lines with single-spaced lines

Because of the way that different operating systems treat carriage returns, text files saved on UNIX or Macintosh computers might be double-spaced when you open them in HomeSite+ for Dreamweaver MX.

To make a double-spaced document single-spaced:

1 Open the double-spaced document, if it is not already open.

2 Select **Search > Replace Double Spacing with Single Spacing**.

The double-spaced lines are collapsed to be single-spaced lines.

Searching with regular expressions

You can use regular expressions (or **RegExp**) to match patterns in character strings during Extended Find and Extended Replace operations.

To search with regular expressions:

1 Select **Search > Extended Search** or **Search > Extended Replace** as necessary.

2 In the Extended Find or Extended Replace dialog box, select Regular expressions.

Enabling Regular expressions disables the Skip tags while searching option.

3 If replacing text in a project or folder, select Make Backups and specify a backup location. If replacing in the current document or in all open documents, consider making backup copies before proceeding.

4 In the Find what box, enter the regular expression for the pattern you want to find.

For more information, see “Using regular expressions” on page 91.

5 Complete the rest of the search dialog box as you would in any extended search or replace.

For details, see “Using extended search commands” on page 160.

6 Click Find or Replace, as applicable.

Checking spelling

You can check the spelling of your document content and, optionally, code syntax. This section provides instructions for configuring and using the spelling checker.

Configuring the spelling checker

This section describes how to configure your spelling checker to best meet your needs.

To set spelling checker options:

1 Open the **Options > Settings > Spelling** pane to view Spell Check options.

2 In the Main Dictionaries box, select every dictionary that you want to use. By default, dictionaries for American English and HTML are installed.

- To spell check the content of HTML tags against an HTML dictionary, select HTML. This does *not* check the validity of your HTML code. To validate your HTML code, see "Validating code" on page 104.
- To add a dictionary to the Main Dictionaries box, copy a dictionary (CLX) file to the Spelling folder under the application root directory.

  The dictionary is added to the Main Dictionaries box in the **Options > Settings > Spelling** pane.

3 In the User Dictionary box, accept the default entry or specify an alternate location and text file to supplement the main dictionaries. For more information, see "About the user dictionary" on page 165.

4 In the Options box, select every option that you need.

  For a description of each option, see "About spelling checker options" on page 167.

5 Set the maximum number of suggestions—up to 20—that the spelling checker should generate for a misspelling. This affects the number of suggestions that appears in the Spell Check dialog box or when you right-click a marked word.

  This option is only available if you select Mark Spelling Errors in the Options box.

6 Click Apply.

About the user dictionary

You cannot add or delete words from the main dictionaries, but you can add or delete words from a custom user dictionary that supplements the main dictionaries. By default, this user dictionary is contained in a file called userdct.txt, which is located in the UserData folder underneath the application root directory.

To use a different user dictionary:

1 In the **Options > Settings > Spelling** pane, in the User Dictionary box, enter or browse to an alternative location and file.

If you specify a file that does not exist, it is created when you first add a word to the user dictionary.

2 Click Apply.

To use the same user dictionary as with Microsoft Office:

1 Locate the Custom.dic file on your computer.

This file is installed with Microsoft Word or another Microsoft Office application.

2 In the **Options > Settings > Spelling** pane, in the User Dictionary box, enter or browse to the absolute path of the Custom.dic file.

3 Click Apply.

To modify the user dictionary, do either of the following:

- Modify the file indirectly while checking spelling, by adding or removing a word in the Spelling dialog box.
- Edit the file directly in HomeSite+ for Dreamweaver MX manually adding and deleting words from the file. The user dictionary is a text file with one word on each line.

About spelling checker options

This following table describes the options that you can select to customize the spelling checker:

Option	Result when selected	Example
Case sensitive checking	Treats different capitalizations of the same word as different words	*State* is different from *state*
Ignore domain names	Skips words which appear to be part of a domain name	http://www.nobodyhere.com/justme/index.html
Ignore HTML markup	Skips text that is enclosed in tag brackets	*width* in <hr width=”75%”>
Ignore words in all capitals	Skips words that are in all capital letters	*HTML* and *CFML*
Ignore words with digits	Skips words containing digits	*Win2K* and *Y2K*
Ignore words with initial capitals	Skips words starting with a capital letter	*Jackie* and *Chan*
Ignore words with mixed case letters	Skips words that contain unusual mixed capitalization	*USoA and THen*
Ignore words with non-alphabetic characters	Skips words that contain characters other than alphabetic letters	*HomeSite+*
Perform OEM conversion	Translates Windows ANSI characters to the currently installed OEM character set	Correctly identifies Chinese characters on a Windows computer that uses a Chinese keyboard
Report duplicate consecutive words	Flags duplicate words that appear next to each other, and allow you to delete the repeated word from the Spelling dialog box	*the the*
Strip possessives	Ignores the possessive suffix of a word	*user's* is checked as *user*
Suggest split words	Suggests two correct words in place of one incorrect word	Suggests *web page* for *webpage*
Suggest words that “sound” similar	Suggests replacements for a misspelling based on phonetic patterns	Suggests *selling* for *celing*
Suggest words that are spelled similarly	Suggests replacements for a misspelling based on typographical similarity	Suggests *cling* for *celing*
Treat contracted (apostrophe-separated) words as separate smaller words	Checks each word that is separated by an apostrophe as a separate word	For *quell'anno*, checks *quell* and *anno* separately
Treat hyphenated words as separate smaller words	Checks each word in a hyphenation as a separate word	For *double-click*, checks *double* and *click* separately

Using the spelling checker

You can mark spelling errors in the current document, run the spelling checker against the current document, or run the spelling checker against all open documents.

To check spelling, do any of the following:

- To mark misspellings in the current document, select **Tools > Mark Spelling Errors**.

  This marks misspellings in the document as it is when you select this command. This does *not* mark misspellings as you type.
- To correct marked misspellings, select **Tools > Mark Spelling Errors** and, in the document, right-click each misspelling and select from the Spelling suggested word list.
- To check spelling in the current document, select **Tools > Spell Check**
- To check spelling in all open documents, select **Tools > Spell Check All**

Verifying links

HomeSite+ for Dreamweaver MX simplifies the job of maintaining sites by testing each link in a selection and reporting any problems it finds. You can verify links to local HTML files, graphics and other media files and other websites. By default, HomeSite+ for Dreamweaver MX checks all links in a selection, but you can skip specific links. Tags in comments are skipped.

If you have installed Linkbot 5.0 or later, you can use it within HomeSite+ for Dreamweaver MX. Select **Tools > Verify Links with Linkbot** to check links in specific files, or select
Tools > Verify Project with Linkbot.

Note
You cannot verify links to secure pages (HTTPS), FTP links, or mailto links. You also cannot perform link verification on large binary files such as EXE, PDF, ZIP, and media file types. This would significantly slow down the operation. Therefore, HomeSite+ for Dreamweaver MX installs a list of excluded file extensions in the Windows Registry LinkVerifyExcludeExts key.

To set options for link verification, do any of the following:

- To change the URL or local directory that HomeSite+ for Dreamweaver MX should use to process the relative link, select a link in the list and click Set Full URL.

  Entries that you make in this dialog box are stored in the drop-down list for future use. Setting the root URL does not modify the source document; it just tells HomeSite+ for Dreamweaver MX how to test relative links.
- To enter a time value after which the link verifier moves to the next link in the list, click Set Timeout.
- If the link is routed through a proxy server, click Set Proxy and enter a server name and port number.

To verify links in the current document:

1 Select **Tools > Verify Links** to open the Links tab on the Results window.

The link list shows the following information for each link in the document:

Field	Description
Source	Current document name, preceded by an icon identifying the link as a document or media file
Link	Link as referenced in the document, including links set with the name attribute in the Anchor tag
Full URL	Path or IP address of the link
Status	Status of the link, initially set to Untested

HomeSite+ for Dreamweaver MX checks each link in order. The Status column displays OK for successful links and, for failed links, a File not found message or the server-generated code.

2 To end the link validation before it completes, click Stop.

To verify links in a project:

1 In the Resources window, on the Projects tab, right-click a project name and select Verify Links.

2 In the Verify Links in Project dialog box, select a root URL option for project files.

3 (Optional) Set a timeout value for the processing of each link.

4 Click OK to run the verification routine.

5 To end the link validation before it completes, click Stop.

To verify a single link:

- In the Results window, on the Links tab, right-click a link in the list and select Verify this Link.

To produce a report of failed links:

- In the Results window, on the Links tab, click Print.

  The report displays in your default browser. You can publish the report, e-mail it, or print it from the browser.

To edit a link:

- In the Results window, on the Links tab, right-click a link in the list and select Edit this Link.

To browse to a link:

- In the Results window, on the Links tab, right-click a link in the list and select Browse this Link.

Using Site View to check page links

Site View provides a graphic representation of each link, beginning in the current document. You can view the links in a tree or chart format. Each link is identified by a file type icon (internal, web, image, mailto, and so on), and the filename.

Site View displays links for the following tags:

- Anchor (a)
- Image (img)
- applet with a code or codebase attribute
- frame with a src attribute

To display the contents of the title tag:

- Right-click in the Site View pane and select **Options > Show <Title>**.
  The link text in the Site View display is replaced with the contents of the `<title>` tag.

To edit a link in the current document:

1 Click the link in Site View to highlight it in the document.

2 Edit the link text as necessary.

3 To update Site View, right-click in the Site View pane and select Refresh.

To view a page linked from the current document:

1 Click the Browse tab in the Editor.

2 Click the link in the current document that goes to the page that you want to view.

To display links beyond the current document:

1 In the Site View pane, double-click a link in the current document.

  The URL is processed and the links in that page are added to the Site View display.

2 Repeat this as necessary for subsequent pages that contain links.

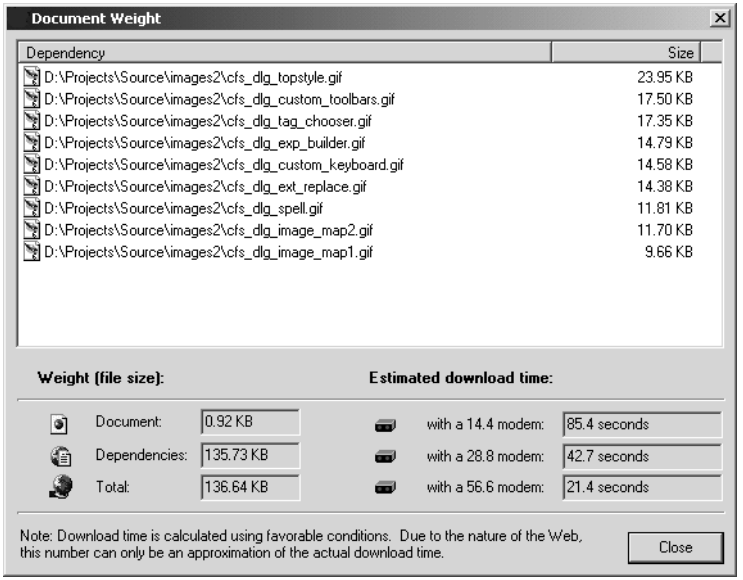
You can drag the vertical window border to enlarge the viewing area as the link list expands. If the view gets too dense, right-click a link in the tree and select Set as Root. This makes the selected link the top item in the tree.

Testing page download times

A document's **weight**, that is, the time it takes for the page to download, is a good metric for testing a site's design and effectiveness. Public sites and intranets with remote users must achieve a balance between attractive content and tolerable page-loading times. HomeSite+ for Dreamweaver MX can help you find that balance by supplying values for document download times across a range of modem speeds.

Only image files (GIF, JPG, PNG) are tested and included in the Dependencies list.

The following graphic shows a sample Document Weight list:



To test the current document's download time:

1 Select **Tools > Document Weight**.

The file's dependencies, such as image and media files, are listed along with file size and download times for a range of modem speeds. Tags in code comments are not calculated.

2 If the page's download time exceeds the site's requirements, edit the page to decrease the number or size of dependencies, then retest.

HomeSite+ for Dreamweaver MX uses the Root URL setting in your FTP configuration to determine the relative path to files.

To set the root URL for an FTP server:

1 In the Resources window, click a Files tab.

2 In the Drive List, select Macromedia FTP & RDS.

3 Right-click a server name and select Properties.

4 In the Configure FTP Server dialog box, complete the Root URL field and click OK.

Chapter 13

Customizing the Development Environment

This chapter introduces the Visual Tools Markup Language (VTML) and the Wizard Markup Language (WIZML), tag-based languages that are used internally by HomeSite+ for Dreamweaver MX.

You can use VTML to generate tag-specific dialog boxes and other user interface elements for entering and editing HTML, XHTML, CFML, JSP, and all other supported language elements.

WIZML enables you to design and build custom wizards to gather user input and to drive application output.

See the VTML Help Reference for the full syntax and descriptions of both languages.

Contents

- About Visual Tools Markup Language (VTML) 176
- Using Tag Chooser 177
- About dialog definition files 179
- Creating a tag definition file 180
- Building a tag editor 182
- Adding tag Help 186
- Container and Control examples 187
- Building a custom wizard 192
- Creating a wizard definition page 193
- Creating a wizard output template 196
- Wizard definition page library 198

About Visual Tools Markup Language (VTML)

You can change the content and layout of tag editors by editing VTML files directly in HomeSite+ for Dreamweaver MX.

You can also create your own editors. Many custom tag developers use VTML to build tag editors to distribute with their tags libraries. You can find a selection of custom dialogs and tag editors in the VTML section of the Visual Tools Developer Exchange at http://devex.macromedia.com/developer/gallery/VisualTools.cfm.

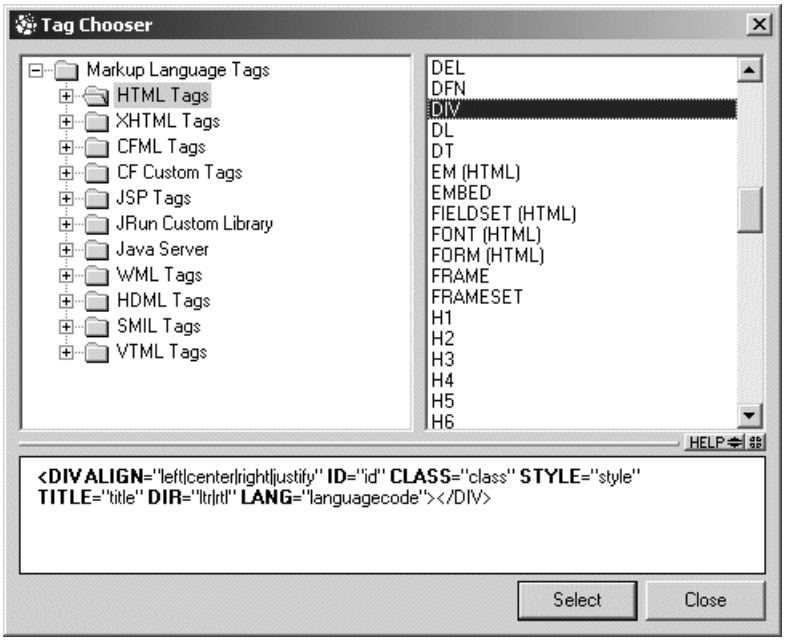
VTML controls and customizes the following user interface elements:

- Tag Inspector
- Tag Tips (F2 in a tag)
- Tag Insight
- Tag Editing Dialogs
- Tag Outline Profiles
- Wizards
- Tag Chooser elements
- Expression Builder

Using Tag Chooser

Tag Chooser contains all supported tag sets and is the primary user interface for tag selection. The following screen shot shows Tag Chooser with an HTML tag highlighted and the embedded Help for that tag displayed.

Press Select to open the tag editor for the highlighted tag. Tags that do not have attributes, and therefore do not need a tag editor, are indicated in the list by displaying both start and end tags, and are inserted directly into the document.



Tag Chooser content and behavior is determined by a set of VTML files which can be identified by opening the Extensions\MarkUpTag.vtm files.

Expression Builder displays a hierarchical view of all supported expression elements, and fully supports ColdFusion 5. Elements can be selected to create complex expressions for insertion in applications and for other programming needs.

You can adapt these tools to meet new requirements imposed by the evolution of HTML, CFML, JSP, other tag-based languages, and supported web technologies.

The content and the behavior of these tools is controlled by two templates: MarkupTags.VTM and ExpressionElements.VTM. Both are located in the \Extensions\ subdirectory of your visual tools installation directory.

Two VTML tags, `cat` and `e`, let you customize the content of tag dialog boxes:

Exploring the new VTML structure

If you modified existing language elements or added new elements, such as custom tags, in previous versions, you will notice a significant change in the Extensions folder and in the Extensions\MarkUpTags file. In previous releases, the MarkUpTags file was a repository of VTML coded information for all the supported languages. It was a large complex document. This release takes a more modular approach to VTML language support by using the new `vtinclude` tag to group languages in discrete files that are then included in the MarkUpTags file.

To modify existing elements, you first identify the language file in MarkUpTags.vtm, then open that file in the Extensions\Includes folder to make your changes.

About dialog definition files

The structure of the included files that are referenced in MarkUpTag.vtm and in ExpressionElements.vtm is basic. The files contain a set of category and element tags. Category tags can contain any number of elements or other nested category tags, for example:

```
<cat … main category>
  <cat … sub-category No.1>
    <e … >
    <e … >
    <e … >
  </cat>
  <cat … sub-category No.2>
    <e … >
    <e … >
  </cat>
</cat>
```

Category tag

The `cat` tag defines a category in the Category tree. The Category tree populates the left pane of Tag Chooser and Expression Builder.

For syntax and usage information, see ../VTML_Reference/cat.htmlcat/a in the VTML Reference.

Element tag

The e tag defines elements within a category. These elements populate the right pane in Tag Chooser and the Expression Builder.

For syntax and usage information, see ../VTML_Reference/e.htmle/a in the VTML Reference.

The following section explains how to create and update category and element tags.

Creating a tag definition file

User interfaces, such as Tag Insight and Tag Inspector require, tag-specific information to operate properly. For instance, Tag Inspector needs to know the attributes of the tag being entered, the type of each of those attributes, and, in some instances, the enumerated values of an attribute. Individual tag definition files store this information.

These tag definitions are stored in \Extensions\TagDefs\. For instance, all the information about the `applet` tag is stored in \Extensions\TagDefs\HTML\Applet.vtm. The definition files are organized in language directories to prevent possible name conflicts between various markup languages.

Whenever you make changes to a VTM file or create a new one, save the file, then press Ctrl+Alt+Shift+C to apply the changes.

The following features use tag definition files:

- Tag Inspector
- Tag Tips
- Tag Insight
- Tag editors

Tag definition file structure

You can customize existing tag definitions and create new tag definition files. Every tag editor file contains the following markup structure:

```
<tag>
  <attributes>
  … Defines tag attribute properties and behavior
  </attributes>

  <attribcategories>
  … Defines logical grouping for tag attributes
  </attribcategories>

  <editorlayout>
  … Defines the layout of a tag editor
  </editorlayout>

  <taglayout>
  … Defines the tag generation template
  </taglayout>

  <tagdescription>
  … HTML-based documentation for the tag

  </tagdescription>

</tag>
```

You can create the definition file in one of the following ways:

- Write it manually.
- Create it from an existing tag definition file.
- Use the Tag Definitions Library dialog box to add a tag and edit the generated file.

The following sections describe how to define attribute categories

Defining attributes

The `attributes` block defines attributes inside the main `tag` block.

See ../VTML_Reference/attributes.html`attributes`/a in the VTML Reference for syntax and usage information.

The `attributes` block can only contain attrib tags. The following example demonstrates the definition of four tag attributes:

```
<attributes>
  <attrib name="value">
  <attrib name="title">
  <attrib name="alt">
  <attrib name="align">
</attributes>
```

In most cases, features such as Tag Insight require more than just the names of the attributes. You can use the `attrib` tag to define the following attributes:

- Attribute value types
- Enumerated values for the `align` attribute

The optional `caption` attribute specifies the form in which the option appears in the drop-down lists, while the `value` attribute specifies the underlying value used by the attribute.

See ../VTML_Reference/attrib.html`attrib`/a in the VTML Reference for syntax and usage information.

Defining attribute categories

Use the `attribcategories` section to organize the attributes when viewed in Tag Inspector. The `attribcategories` block can only contain `attribgroup` tags. The following example shows four attribute category definitions.

```
<attribcategories>
  <attribgroup name="appearance"
    elements="background,bgproperties,leftmargin,,topmargin"/>
  <attribgroup name="colors"
    elements="bgcolor,vlink,alink,link,text"/>
  <attribgroup name="misc"
    elements="gizmo"/>
</attribcategories>
```

See ../VTML_Reference/attribgroup.html`attribgroup`/a in the VTML Reference for syntax and usage information.

Building a tag editor

Defining a tag editor is requires the following three tasks:

1 Layout the dialog box controls.

2 Specify how to populate the controls with tag attributes.

3 Define the tag generation template.

Open the \Extensions\TagDefs\mytag.vtm file to see how these tasks are coded.

Defining controls

The `control` and `container` tags represent graphical controls. The tags are nearly identical in definition, the difference being that only `container` tags have the capability to contain `control` tags.

A Panel control is a good example of a control that can be a container containing a control, such as a label or a text box:

```
<tag>
  <editorlayout height=50 width=200>
    <container  name="Panel1" type="Panel" width=150 height=50>
      <control name="lblCode" type="Label" caption="Code" down=20
        right=20 width=70/>
      <control name="txtCode" type="TextBox" anchor="lblCode"
      <corner="NE" width="30"/>
    </container>
  </editorlayout>
</tag>
```

You can name the above template mytag.vtm and test it by attempting to edit an empty `mytag` tag.

The example displays a single Panel container containing `Label` and `TextBox` controls. Only the following controls can be containers:

- **TabDialog** A tab dialog control capable of containing `TabPage` container controls.
- **TabPage** Only used inside a `TabDialog` container control.
- **Panel** A general purpose panel container control which can contain any regular or container controls.

The control represented by a `control` or a `container` tag is defined by the `type` attribute; the `width` and `height` attributes determine the size.

The `anchor` and `corner` attributes determine the point relative to which the control is positioned. The anchor can be specified as the name of any control that was already laid down. The corner specifies the corner of the anchor control to be used for positioning. The possible values are `NE`, `NW`, `SE`, and `SW`. The `down` and `right` attributes then specify the pixel offset from the `anchor` control.

See ../VTML_Reference/control.html`control`/a and ../VTML_Reference/container.html`container`/a in the VTML Reference for syntax and usage information.

Populating dialog boxes with tag data

Once the layout of controls is completed, you must define the way in which the tag editor controls are populated when you are editing an existing tag. This is done in the `attributes` block of the main tag editor template.

The `attributes` block can contain `attrib` and `event` tags. The `attrib` tag defines the way in which tag attribute values are inserted into the dialog controls, for example:

```
<attributes>
  <attrib name="value" control="txtName"/>
  <attrib name="title" control="txtTitle"/>
  <attrib name="title" control="txtTitle2"/>
  <attrib name="alt" control="txtAltText"/>
  <attrib name="align" control="dropAlign"/>
</attributes>
```

The `name` attribute of the `attrib` tag specifies the name of the attribute, while `control` specifies which control the value of that attribute should be assigned to. You can have multiple `attrib` tags with the same name. This is common for more complex tag editor dialog boxes where a single attribute value might have to be filled into multiple controls.

Special attrib tag variable names

The following special variables can be used:

- **$$TAGBODY** This special tag attribute name is used when a control needs to be populated by the body of a tag. An example of such a tag editor is the editor for the HTML tag `textarea`. The body of the `textarea` tag is filled into the `txtTextAreaContent` control using the following `attrib` code:

  ```
  <attrib name="$$TAGBODY" control="txtTextAreaContent"/>
  ```

- **$$TAGSTRING** and **$$WHOLETAGSTRING** These attributes can be used to populate controls with the start tag string or the whole tag including its start tag, body and the end tag. This allows ActiveX controls to be passed to the entire tag for custom processing and specialized tag editor behavior.
- **$$EmbeddedCodeString** This attribute represents just the text within a tag. It excludes the tag name, delimiters, and white space outside the text block.

Generating a tag

The final stage in the process of building a tag editor is defining how a tag gets generated from the data entered in the editor controls. The tag generation logic is stored in the `taglayout` block. This block contains a short template used to generate the final tag string.

A good way to get started is to have a look at the taglayout sections of existing HTML tag editors located in the \Extensions\TagDefs\HTML directory.

See ../VTML_Reference/taglayout.html`taglaout`/a in the VTML Reference for syntax and usage information.

Variables passed to the layout template

The value of each control of the tag editor is passed to the template using a variable with the same name, for example, a ColorPicker control named `colorBGColor` will pass its value in a `colorBGColor` variable. The `taglayout` template can then use this data to generate the tag string.

```
<taglayout>
  <mytag color="$${colorBGColor}">
</taglayout>
```

This example shows a basic layout template for a hypothetical tag with a single attribute, `color`. Notice that variables are embedded using the $${} delimiters. If the user selects White in the `colorBGColor` ColorPicker control, the template generates this tag:

```
<mytag color="White">
```

Special variables

In addition to the control variables, a few other parameters get sent to the `taglayout` template:

- **OPTIONLowerCaseTags** Returns true or false. Specifies whether the tag should be generated using lowercase.
- **EDITORTagIndentString** Maintains indentation for tag attributes and body.
- **OPTIONLinearLayout** Returns true or false. Specifies whether the tag should be generated with its attributes in a single line or not.
- **TAGDATAUnknownAttributes** Returns a string containing all attributes that were contained in the edited tag string but are not recognized by the editor.

Using OPTIONLowerCaseTags

You can use this parameter to create a layout template, which generates a tag in lowercase or uppercase, based on user preferences. Here is a version of the mytag layout template responding to case preferences:

```
<taglayout>
  <WIZIF OPTIONLowerCaseTags EQ 'true'>
    <mytag color="$${colorBGColor}">
  <WIZELSE>
  <mytag COLOR="$${colorBGColor}">
</WIZIF>
</taglayout>
```

Maintaining tag indentation

The variable `EditorTagIndentString` contains an indentation string for the currently selected tag. If the start tag is indented using tabs and characters, the string is represented as the corresponding combination of tabs and spaces. This variable can be used to correctly indent tag attributes, as well as tag body contents for tags which are already indented.

Using OPTIONLinearLayout

Here is a version of the mytag layout template that responds to user preferences for single line or indented layout:

```
<taglayout>
  <WIZIF OPTIONLinearLayout EQ 'true'>
    <WIZSET Spacer = ' '>
  <WIZELSE>
    <WIZSET Spacer =  Chr(13) & Chr(10) & ' '>
  </WIZIF>
  <mytag color="$${clrBGColor}"$${Spacer}face="$${fontFace}
    "$${Spacer}size="$${txtSize}">
</taglayout>
```

The template would generate a tag based on the following user layout preference:

LINEAR:

```
<mytag color="White" face="Arial" SIZE="10">
```

NONLINEAR:

```
<mytag color="White"
  face="Arial"
  size="10">
```

Using TAGDATAUnknownAttributes

The `TagDataUnknownAttributes` tag contains the list of attributes that are contained in the original tag string but are not supported by the editor. For example, you can write an editor for the HTML tag `input`. that provides editing capabilities for all basic attributes, however, the editor will not cover JavaScript event attributes such as, `onCLick`). To prevent loss of what are in effect unknown attributes during the editing process, the editor engine creates the TagDataUnknownAttributes variable containing a list of unknown attributes together with their original values. You can use this variable to "stamp" all the unsupported attributes at the end of the tag you are generating.

```
<taglayout>
  <mytag color="$${colorBGColor}">
    <wizif TagDataUnknownAttributes NEQ''> $$TagDataUnknownAttributes
    </wizif>
</taglayout>
```

If you edit a tag `<mytag color="Blue" onClick="CallThis">`, the above template preserves the `onClick` attribute even though it is not supported in the editor.

Adding tag Help

You can associate an HTML-based Help document with a tag by embedding the Help text inside the `tagdescription` block.

Here is an example of a `tagdescription` block and the result in a tag editor:

```
<tagdescription height=100>
<B>cfapplication</B>
<P>Defines scoping for a ColdFusion application and
enables or disables storing client variables in the system
registry. By default, client variables are disabled.
Cfapplication is typically used in the application.cfm
file to set defaults for a specific ColdFusion application.
</tagdescription>
```

As the Help content grows, it might become cumbersome to specify the entire body of the Help inside the `tagdescription` block. In addition, large bodies of Help embedded in the editor file will cause the editor dialog box to open more slowly as more markup has to be parsed to compose the editor. Under these circumstances, it advisable to provide large Help contents in a separate HTML file. Such files can then be referenced using a relative path from the tag editor template.

For example:

```
<tagdescription helpfile="Docs/TagHelpFile.htm"/>
```

Container and Control examples

This section contains example code for VTML containers and controls.

TabDialog

```
<container name="MainTabDialog" type="TabDialog" width=maximum
     height=maximum>
  <container  name="TabPage1" type="TabPage" caption="TEXTAREA Tag">
... embedded controls
  </container>
  <container  name="TabPage2" type="TabPage" caption="Content">
... embedded controls
  </container>
</container>
```

See ../VTML_Reference/container.html#tabdialog`tabdialog`/a in the VTML Reference for syntax and usage information.

TabPage

```
<container name="MainTabDialog" type="TabDialog" width=maximum
     height=maximum>
  <container  name="TabPage1" type="TabPage" caption="TEXTAREA Tag">
    ... embedded controls
  </container>
  <container  name="TabPage2" type="TabPage" caption="Content">
    ... embedded controls
  </container>
</container>
```

See ../VTML_Reference/container.html#tabpage`tabpage`/a in the VTML Reference for syntax and usage information.

Panel

```
<editorlayout height=225>
  <container  name="MainTabDialog" type="TabDialog" width=maximum
    height=maximum>
    <container  name="TabPage1" type="TabPage" caption="mytag Tag">
    <container  name="Panel1" type="Panel" down=5 right=10
      width="maximum" height=125>
      <control name="lblSource" type="Label" caption="Source:"
        down=17 right=10 width=50/>
      <control name="txtSource" type="TextBox" anchor="lblSource"
        corner="NE" width="maximum"/>
      <control name="lblAlign" type="Label" caption="Align:"
        anchor="lblSource" corner="SW" down=11 width=50/>
      <control name="dropAlign"type="DropDown" anchor="lblAlign"
        corner="NE" width=100>
        <item value="TOP" caption="TOP" />
```

```
        <item value="MIDDLE" caption="MIDDLE" selected/>
        <item value="BOTTOM" caption="BOTTOM" />
      </control>
    </container>
    <container  name="Panel2" type="Panel" caption=" Panel 2 "
      anchor="Panel1" corner="SW" down=5 width="maximum"
        height="maximum">
    </container>
    </container>
    <container  name="Advanced" type="TabPage" caption="Advanced">
    </container>
  </container>
</editorlayout>
```

See ../VTML_Reference/container.html#panel`panel`/a in the VTML Reference for syntax and usage information.

Label

```
<control name="lblSource" type="Label" caption="Source:" down=17
        right=10 width=50/>
<control name="lblSource" type="Label" caption="Source:" down=17
        right=10 width=50/>
<control name="txtSource" type="TextBox" value="Some Value"
  anchor="lblSource" corner="NE" width="maximum"/>
```

See ../VTML_Reference/control.html#label`label`/a in the VTML Reference for syntax and usage information.

DropDown

```
<control name="lblAlign"
  type="Label" caption="Align:"
  anchor="lblSource" corner="SW" down=11 width=50/>
<control name="dropAlign"
  type="DropDown" anchor="lblAlign" corner="NE" width=100>
  <item value="TOP"    caption="TOP" />
  <item value="MIDDLE" caption="MIDDLE" selected/>
  <item value="BOTTOM" caption="BOTTOM" />
</control>
```

See ../VTML_Reference/control.html#dropdown`dropdown`/a in the VTML Reference for syntax and usage information.

ListBox

```
<control name="lblSource"
  type="Label" caption="Source:"
  down=17 right=10 width=50/>
```

See ../VTML_Reference/control.html#listbox`listbox`/a in the VTML Reference for syntax and usage information.

FontPicker

```
<control name="lblFace" type="Label" caption="Font:" down=17 right=10
  width=50/>
<control name="fontFace" type="FontPicker" anchor="lblFace" corner="NE"
  width="maximum"/>
```

See ../VTML_Reference/control.html#fontpickerfontpicker/a in the VTML Reference for syntax and usage information.

ColorPicker

```
<control name="lblColor" type="Label" caption="Color:" anchor="lblFace"
  corner="SW" down=11  width=50/>
<control name="colorColor" type="ColorPicker" anchor="lblColor"
  corner="NE" width="maximum"/>
<control name="lblColor" type="Label" caption="Color:" anchor="lblFace"
  corner="SW" down=11  width=50/>
<control name="colorColor" type="ColorPicker" anchor="lblColor"
  corner="NE" width="maximum"/>
```

See ../VTML_Reference/control.html#colorpickercolorpicker/a in the VTML Reference for syntax and usage information.

Checkbox

```
<control name="checkNoShading" type="CheckBox" caption=" No Shading"
  anchor="numWidth" corner="NE" down=4 right=20 width="maximum"/>
```

See ../VTML_Reference/control.html#checkboxcheckbox/a in the VTML Reference for syntax and usage information.

RadioGroup

```
<control name="radioNameConflict"
     type="RadioGroup" caption="Radio One"
     anchor="lblAccept" corner="SW" down=35
     height=maximum width="maximum">

  <item value="error" caption="Error - The file
  will not be saved and ColdFusion will return
  an error." selected="true"/>

  <item value="SKIP" caption="Skip - Neither
  saves the file nor throws an error."/>
  <item value="overwrite"  caption="Overwrite -
  Replaces the existing file if name conflict occurs." />

  <item value="makeunique" caption="Makeunique - Automatically
    generates a unique filename for the upload." />
</control>
```

See ../VTML_Reference/control.html#radiogroup`radiogroup`/a in the VTML Reference for syntax and usage information.

TextArea

```
<control name="txtContent" type="TextArea"
  down=5 right=5
  width=maximum height=maximum
  MAXWIDTHPADDING=5 MAXHEIGHTPADDING=5/>
```

See ../VTML_Reference/control.html#textarea`textarea`/a in the VTML Reference for syntax and usage information.

SQLTextArea

```
<container  name="TabPage1" type="TabPage" caption="CFQUERY Tag">
  <container  name="Panel1" type="Panel" down=5 right=10
    width="maximum" height=80>
    <control name="lblQueryName"
      type="Label" caption="Query Name:"
      down=17 right=10 width=80/>
    <control name="lblDataSource"
      type="Label" caption="Data Source:"
      anchor="lblQueryName" corner="SW"
    down=10 right=0 width=80/>
    <control name="txtQueryName" type="TextBox" anchor="lblQueryName"
      corner="NE"  width=130/>
    <control name="txtDataSource" type="TextBox"
      anchor="lblDataSource" corner="NE" width=130/>
    <control name="lblMaxRows" type="Label" caption="Max Rows:"
      anchor="txtQueryName"  corner="NE" down=0 right=10 width=70/>
    <control name="lblTimeout" type="Label" caption="Timeout:"
      anchor="txtDataSource" corner="NE" down=0 right=10 width=70/>
    <control name="numMaxRows" type="TextBox" anchor="lblMaxRows"
      corner="NE" width=30/>
    <control name="numTimeout" type="TextBox" anchor="lblTimeout"
      corner="NE" width=30/>
    <control name="checkDebug" type="CheckBox" caption="Print debug
      info" anchor="numTimeout" corner="NE" right=10 down=4
      width=maximum/>
  </container>
    <control name="lblSQLStatement" type="Label" caption="SQL
      Statement:" anchor="Panel1" corner="SW"down=10 right=0
      width=110/>
    <control name="txtSQLStatement" type="SQLTextArea"
      anchor="lblSQLStatement" corner="SW" down="8" width=maximum
      height=maximum DSNAMECONTROL="txtDataSource"
      QUERYNAMECONTROL="txtQueryName"/>
</container>
```

See ../VTML_Reference/control.html#sqltextarea`sqltextarea`/a in the VTML Reference for syntax and usage information.

FileBrowser

```
<control name="lblSource" type="Label"
  caption="Source:" down=17 right=10
  width=60/>
<control name="txtSource" type="FileBrowser" anchor="lblSource"
  corner="NE" width="maximum" RELATIVE
        FILTER="*.htm;*.html;*.cfml;*.cfm;*.asp" />
```

See ../VTML_Reference/control.html#filebrowser`filebrowser`/a in the VTML Reference for syntax and usage information.

Image

```
<control name="imgApplet" type="Image" filepath="Images/Applet.bmp"
  down=10 right=10 autosize="Yes"/>
```

See ../VTML_Reference/control.html#image`image`/a in the VTML Reference for syntax and usage information.

StyleTextBox

See ../VTML_Reference/control.html#styletextbox`styletextbox`/a in the VTML Reference for syntax and usage information.

ActiveX

```
<control name="activexGizmoPicker" type="ActiveX"
  PROGID="company.Gizmo"/>
```

See ../VTML_Reference/control.html#activex`activeX`/a in the VTML Reference for syntax and usage information.

Building a custom wizard

This section describes how to collects information from a user with a wizard. Wizards are an integral part of many software products because they enable users to perform complex tasks in an orderly, comprehensible user interface. Also, a well-designed wizard controls its input and ensures a high probability of user success.

HomeSite+ for Dreamweaver MX includes a number of wizards, and you can create your own with the Wizard Markup Language (WIZML). If you have worked with VTML to create or edit tag dialog boxes, you are familiar with building user interface containers and controls and with defining page layout. You can use these skills to add wizards to your applications.

To create a wizard:

1 Write a wizard definition (VTM) file to specify the pages, parameters, output, and logical flow.

2 Implement one or more output template (WML) files for the wizard.

3 Create wizard graphic (BMP) files.

Each of these steps is described in detail in the following sections.

The recommended way to organize wizards and supporting files is to save the vtm and wml files in the \Wizards\Custom folder and to save the image files in the \Wizards\Images folder of your HomeSite+ for Dreamweaver MX directory.

Creating a wizard definition page

The first step in building a custom wizard is to write a VTML file to define the user interface and output parameters. This section describes the VTML tags used in this part of the process, and presents an example definition file.

For a complete description of the WIZML tag set, see ../VTML_Reference/wizml.htmlThe WIZML Language/a in the VTML Reference.

For syntax and usage information of VTML tags used to provide the user interface in wizard development, see ../VTML_Reference/wizards.htmlWizards/a in the VTML Reference.

Dynamic expressions in tags

Any tag attribute can combine static, constant text with embedded dynamic expressions that reference parameters or input controls. To embed an expression within a text string, the following syntax is utilized:

```
$${ expression }
```

So, for example, to set the REQUIRED attribute of a parameter based on whether another value was set, you would use the following syntax:

```
<PARAM name="RowsPerPage" value="10"
  REQUIRED="$${ ParameterExists('Customize') }">
```

Or, to customize the OUTPUTFILE attribute of the TEMPLATE tag using a name attribute entered by the user, you would use the following syntax:

```
OUTPUTFILE="$${Name}Admin.cfm">
```

The expression syntax supported within the wizard configuration file is the same as the one supported in wizard output templates (see the reference section for more details).

Bound controls

One of the most powerful capabilities of wizard pages is bound controls. Bound controls allow you to place controls onto the wizard page and have their values automatically bound to wizard parameters. To do this, simply add an INPUT sub-tag to the PAGE tag for each control you wish to bind, making sure that the name attribute of the INPUT tag matches the Name property of the control. All controls specified in the layout can be bound.

Wizard definition page example

This sample wizard creates an HTML template.

```
<WIZARD name="DefaultTemplate" caption="Default HTML Template">
<!--- wizard parameters --->
<PARAM name="sDocType" value="HTML 4.0" REQUIRED="true">
<PARAM name="sTitle" value="">
```

```
<PARAM name="bMetaDescr" value="false">
<PARAM name="sMetaDescr" value="">
<PARAM name="bMetaKeywords" value="false">
<PARAM name="sMetaKeywords" value="">

<!--- WIZARD PAGE --->

<!--- attributes page --->
<PAGE name="DocAttribs" type="DYNAMIC"
  caption="HTML Document Attributes"
  IMAGE="..\\images\\main.bmp">

  <PAGELAYOUT>
    <control name="lblDocType" type="label"
      down="10" right="10"
      width="90"
      caption="Document Type:"
    />

    <control name="ddDocType" type="DropDown"
      EDITABLE="no"
      anchor="lblDocType" corner="NE" width="maximum" down="-5">
           <item caption="HTML 2.0" value="HTML 2.0"/>
           <item caption="HTML 3.2" value="HTML 3.2"/>
           <item caption="HTML 4.0" value="HTML 4.0"/>
    </control>

    <control name="lblTitle" type="label"
      anchor="lblDocType" corner="SW" down="20"
      width="90"
      caption="Title:"
    />


    <control name="tbTitle" type="TextBox"
      anchor="lblTitle" corner="NE" width="maximum" down="-5"
    />

  <container name="pnlMetaDescription" type="Panel"
       caption="Meta Description"
       anchor="lblTitle" corner="SW" down="20"
       width="maximum" MAXWIDTHPADDING="10" height="80"
         LFHEIGHT="90">

       <control name="chkMetaDescr" type="CheckBox"
         caption="Add meta description:"
         down="20" right="15"
         width="maximum"/>

       <control name="tbMetaDescr" type="TextBox"
         anchor="chkMetaDescr" corner="SW" down="8"
         width="maximum"/>

    </container>
```

```
  </PAGELAYOUT>

  <INPUT name="ddDocType" PARAM="sDocType">
  <INPUT name="tbTitle" PARAM="sTitle" REQUIRED="yes" VALIDATIONMSG=
    "Please enter a document title" or some equivalent message>
  <INPUT name="chkMetaDescr" PARAM="bMetaDescr">
  <INPUT name="tbMetaDescr" PARAM="sMetaDescr">
</PAGE>

<!--- attributes page --->
<PAGE name="MetaKeywords" type="DYNAMIC"
  caption="Meta Keywords"
  IMAGE="..\\images\\main.bmp">

  <PAGELAYOUT>

    <control name="chkMetaKeywords" type="CheckBox"
      caption="Add meta keywords:"
      down="15" right="10"
      width="maximum"/>

    <control name="taMetaKeywords" type="TextArea"
      anchor="chkMetaKeywords" corner="SW" down="10"
      height="maximum" width="maximum"/>

  </PAGELAYOUT>

  <INPUT name="chkMetaKeywords" PARAM="bMetaKeywords">
  <INPUT name="taMetaKeywords" PARAM="sMetaKeywords">

</PAGE>


<!--- OUTPUT TEMPLATE --->

<TEMPLATE
  name="Custom.wml"
  OUTPUTFILE="MyFile.cfm"
  DESCRIPTION="New HTML file">

</WIZARD>
```

Creating a wizard output template

The Wizard Markup Language (WIZML) enables the customization of files produced by the wizards. WIZML is used inside the templates to dynamically create files based on the data provided by the wizard. For example, if a wizard generates a tag called <GIZMO> with a single attribute FILEPATH, the template could look like this:

```
<GIZMO FILEPATH="$${txtFilePath}">
```

This example creates a file with a single <GIZMO> tag. Notice the syntax $${variablename} that is used to populate the value of FilePath with the actual value entered in the wizard.

WIZML output templates use a high-level markup syntax that works very much like CFML. Supported tags include WIZSET, WIZELSE/WIZELSEIF, WIZLOOP, and WIZINCLUDE. In addition, a simple expression syntax that is a subset of the ColdFusion expression syntax and function library is supported within output templates.

Parameters

Output templates are driven by the values of parameters, much like ColdFusion templates are driven by the values of Form and URL parameters. Parameters can be output directly or can be used to customize the type of output generated. The values of these wizard parameters can originate from several locations:

- From a value set by a PARAM tag provided by the wizard
- From an embedded tag editor control
- Through execution of the WIZSET tag within the output template

To output the value of a parameter within a template, use a double dollar sign escape sequence. For example, to output the value of a variable named Color you would use the syntax `$${Color}`. While this is the recommended syntax, you can use a simpler form for a parameter value within the attribute of a WIZ tag. For example, `<WIZIF Color= "black">` is valid.

Expressions and functions

In addition to outputting and manipulating basic parameter values, an expression syntax is also provided. To output the value of an expression, you add a set of curly braces to the $$ and include the expression within the braces; for example:

```
$${ 'This is the ' & Color }
$$( 'The result of 7 divided by 22 is ' & 7/22 }
$$( Left( 'FooBar', 3 ) }
```

Strings are delimited using the single quote character. Arithmetic and concatenation operators are supported (+,-,*,/,&). The comparison operators LT, LTE, GT EQ, and NEQ are supported, and logical comparisons using AND, NOT, and OR are supported.

For syntax and usage information, see ../VTML_Reference/functions.htmlExpressions and Functions/a in the VTML Reference.

WIZ Tags

The behavior of wizard output templates is controlled by the use of WIZ tags in the template. These tags are like ColdFusion tags except that they are prefixed with the characters WIZ instead of ColdFusion.

The following tags are supported tags:

- **WIZSET** Sets a wizard parameter.
- **WIZINCLUDE** Includes another wizard output template.
- **WIZLOOP** Iterates over a set of outputs.
- **WIZBREAK/WIZCONTINUE** Assists in loop flow control.
- **WIZIF/WIZELSEIF/WIZELSE** Sets conditional flow control.

Special considerations

Strings used within an output template tag attribute use the C-language convention (\) for escaping special characters. For example, the following attribute uses a newline character to split the value into two lines:

```
caption="This is line one\nThis is line two"
```

Other special characters of note include carriage-return (\r), tab (\t), and slash (\\). For example, the following attribute references a template in a directory two levels above the current directory:

```
TEMPLATE="..\\..\\Header.wml"
```

Wizard definition page library

A set of seven page definition files is available in HomeSite+ for Dreamweaver MX. The library can be used to quickly build data access capabilities into wizards.

Examples

Following are examples of each of the seven wizard definition pages available in the library. For complete syntax and usage information, see the ../VTML_Reference/wizards.htmlWizards Definition Page Library/a section of the VTML Reference.

SelectNameAndLocation

```
<PAGE name="SelectWizardNameAndLocation" type="SelectNameAndLocation"
        caption="Data Drill-Down Application"
        image="..\\images\\Main.bmp">

  <INPUT name="editApplicationName" param="ApplicationName"
    required="yes"
    validationMsg="You cannot leave the Application Name field
    blank">

  <INPUT name="editLocation" param="Location"required="yes"
    validationMsg="You cannot leave the Location field blank">

</PAGE>
```

SelectDataSource

```
<PAGE name="DataSource" type="SelectDataSource" caption="Data Source"
  image="..\\images\\SelectData.bmp">

  <PARAM name="ListBoxLabel" value="Select data source:">
  <PARAM name="ListBoxDescription"
    value="Choose the data source from which you would like to display
    data.\n\nIf your database is not registered as ODBC data source,
    open the ODBC administrator in Control Panel and add system data
    source for this database.">
  <PARAM name="ResetParams" value="Joins">
  <PARAM name="RemoveParams" value="Tables,SearchFields,ResultFields,
    DetailFields,UniqueIdentifier">

  <INPUT name="cbDataSources" param="DataSource" required="yes"
    validationMsg="You did not select the data source. Please select
    one before proceeding.">

</PAGE>
```

SelectTables

```
<PAGE name="Tables" type="SelectTables" caption="Tables"
  image="..\\images\\SelectTable.bmp">

  <PARAM name="DataSource" value="$${DataSource}">
  <PARAM name="ListBoxLabel" value="Select database tables:">
  <PARAM name="ListBoxDescription"
    value="Please specify the tables which will be involved in this
    application. This should include any tables against which you
    would like to search or tables containing data that will be
    displayed on either the Result or Detail pages.\n\nPress Ctrl or
    Shift together with the mouse click in order to select more than
    one table. Do not select unrelated tables.">
  <PARAM name="MultiSelect" value="yes">
  <PARAM name="ResetParams" value="Joins">
  <PARAM name="RemoveParams" value="SearchFields,ResultFields,
    DetailFields,UniqueIdentifier">

  <INPUT name="lstTables" param="Tables" required="yes"
    validationMsg="You did not select any tables. Please select at
    least one before proceeding.">

</PAGE>
```

SelectTable

```
<PAGE name="Table" type="Table" caption="Table"
  image="..\\images\\SelectTable.bmp">

  <PARAM name="DataSource" value="$${DataSource}">
  <PARAM name="ListBoxLabel" value="Select database table:">
  <PARAM name="ListBoxDescription"
    value="Records from this table will be displayed in the record
    viewer.">
  <PARAM name="RemoveParams" value="Table,ViewFields,EditFields,
    UniqueIdentifier">

  <INPUT name="cbTables" param="Table" required="yes"
    validationMsg="You didn't select the table. Please select one
    before proceeding.">

</PAGE>
```

SelectTableJoins

```
<PAGE name="TableJoins" type="SelectTableJoins" caption="Table Joins"
        image="..\\images\\SelectJoins.bmp">

  <PARAM name="DataSource" value="$${DataSource}">
  <PARAM name="Tables" value="$${Tables}">
  <PARAM name="ListContent" value="$${Joins}">
```

```
  <INPUT name="lstJoins" param="Joins">

</PAGE>
```

SelectFields

```
<PAGE name="SearchFields" type="SelectFields" caption="Fields for
        Search
  page" image="..\\images\\SearchCriteria.bmp">

  <PARAM name="DataSource" value="$${DataSource}">
  <PARAM name="Tables" value="$${Tables}">
  <PARAM name="ListBoxLabel" value="Select the search fields:">
  <PARAM name="ListBoxDescription"
    value="Choose all fields that should be included as search
    criteria on the Search page. Press Ctrl or Shift together with the
    mouse click in order to select more than one field.">
  <PARAM name="MultiSelect" value="yes">

  <INPUT name="lstFields" param="SearchFields" required="yes"
    validationMsg="You did not select any fields. Please select at
    east one before proceeding.">

</PAGE>
```

SelectField

```
<PAGE name="IDField" type="SelectField" caption="Unique Identifier"
  image="..\\images\\UniqueIDDetail.bmp">

  <PARAM name="DataSource" value="$${DataSource}">
  <PARAM name="Tables" value="$${Tables}">
  <PARAM name="ListBoxLabel" value="Select the unique identifier for
    the Detail page:">
  <PARAM name="ListBoxDescription" value="In order to 'drill-down' to
    the detail page the wizard needs to know the unique identifier for
    the detail page. This is the field that determines which record
    should be displayed in detailed form.\n\nFor example, if you are
    building an application to search an employee database you might
    use a field called 'Employee_ID' as the unique identifier.">
  <PARAM name="MultiSelect" value="no">

  <INPUT name="cbFields" param="UniqueIdentifier" required="yes"
    validationMsg="You did not select the unique identifier. Please
    select one before proceeding.">

</PAGE>
```

Chapter 14

Scripting the Visual Tools Object Model

Using the Macromedia Visual Tools Object Model (VTOM), developers can extend their work in two broad categories:

- Power users can script recurring tasks.
- Application developers can call HomeSite+ for Dreamweaver MX functionality for use in their applications.

This chapter describes how to write and execute scripts in HomeSite+ for Dreamweaver MX. It also provides the syntax and examples of the objects in HomeSite+ for Dreamweaver MX.

Contents

- Writing and executing scripts 202
- Application object 205
- ActiveDocument object 236
- DocumentCache object 247
- Project object 250
- ProjectManager object 252
- DeploymentManager object 258
- HTTPProvider object 266
- ZIPProvider object 276
- ActiveScripting examples 282
- Third-party add-ins 286
- Table of CommandID values 288
- Table of SettingID values 292

Writing and executing scripts

HomeSite+ for Dreamweaver MX exposes an Object Model, enabling developers to manipulate program functionality from external applications. In addition, power users can create scripts to automate tasks using JScript or VBScript and run the scripts from HomeSite+ for Dreamweaver MX.

This internal scripting feature requires the Microsoft ActiveScripting engine version 3.1 or later, also known as Windows Script. The engine is not installed with HomeSite+ for Dreamweaver MX. If you installed Microsoft Internet Explorer 4.0 or later on your computer, then you have the correct ActiveScripting engine. Otherwise, you must download it from http://www.microsoft.com/msdownload/vbscript/scripting.asphttp://www.microsoft.com/msdownload/vbscript/scripting.asp/a.

Note
If you have trouble running an external Windows (*.ws) script, check if your anti-virus software is blocking its execution. Anti-virus software should not affect JScript or VBScript scripts that are run from within HomeSite+ for Dreamweaver MX.

The VTOM hierarchy

The object hierarchy is basic; the Application object is the parent of all the other objects and none of the other objects is a parent.

The VTOM hierarchy can be illustrated as follows:

Application
- ActiveDocument
- DocumentCache
- Project
- ProjectManager
- DeploymentManager
- HTTPProvider
- ZIPProvider

The simplicity of the object model makes syntax notation very straightforward. To call a property or method, begin at the object root, append the child object name (if any), the property or method name, separated by a period, and then the parameters.

For example,

```
Application.ActiveDocument.SaveAs(CurrentFolder + '\\' + sFile);
```

Writing a script

For a script to be executable from within HomeSite+ for Dreamweaver MX, it must contain a Main routine. In JScript, create a function Main; in VBScript, create a Sub Main routine. Without this routine, the script fails.

The program determines which language engine to use based on the extension of the script file. If the file extension is BAS, VB, or VBS, the VBScript engine executes the script. Otherwise, the program uses the JScript engine.

To automate HomeSite+ for Dreamweaver MX tasks, you must familiarize yourself with the Visual Tools Object Model (VTOM). The main object is the Application object. The Application object contains two important child objects, the ActiveDocument object and the DocumentCache object. You can use these objects to write scripts for common tasks.

In addition, the Application object contains a number of toolbar-related functions, which enable you to create toolbars dynamically.

For best performance, you should create an Application object variable and use it throughout the script rather than continually referencing the Application object directly.

For example, the following code creates an Application object variable in JScript:

```
function Main() {
      var app = Application; //create application object variable
      app.WindowState = 2; //maximize the window
```

The following code creates an Application object variable in VBScript:

```
Sub Main
  Dim app
  set app = Application 'create application object variable
  app.WindowState = 2 'maximize the window
End Sub
```

The Application object is only available from scripts that are executed within HomeSite+ for Dreamweaver MX.

To access the Application object from an external program, use the language's equivalent of `CreateObject("AllaireClientApp.TAllaireClientApp")`.

Note
When writing and editing VTOM scripts and when porting code from VBScript, remember that JScript is case-sensitive.

Executing a script

If the scripting engine encounters an error while executing your custom script, the script file opens in the Editor with the error line highlighted. In addition, information about the error displays in the status bar, which helps you debug the problem.

In the Customization dialog box, the Keyboard Shortcuts tab includes a shortcut (Shift + Ctrl + Q) for the Execute current document as ActiveScript command. You can use this command to pass the current document to the ActiveScripting engine, which provides you with a tool for debugging your scripts. As with toolbutton-based scripts, the scripting language is determined by the file extension of the current document.

Creating a custom toolbutton or toolbar

After you write a script, you can create a custom toolbutton to execute the script from within the program. If you intend to write multiple scripts, you can create a toolbar for them. You can rearrange the buttons and add separators in the toolbar.

To create a custom toolbutton:

1 On the Toolbars tab, select your custom script toolbar from the Toolbars list.

The toolbar displays at the top of the tab.

2 Click Add Custom Button.

3 In the Custom Toolbutton dialog box, select the Execute an ActiveScript file option.

4 Enter a path and filename in the Script File box.

5 Make selections in the Button Image, Button Caption, and Button Hint boxes.

You can optionally create script hot keys on the Script Shortcuts tab and then click the Show keyboard shortcuts in toolbutton hints box on the Toolbars tab.

6 Click OK.

To create a toolbar for custom scripts:

1 Select **Options > Customize**.

2 Click the Add Toolbar button on the Toolbars tab.

3 Enter a name for the toolbar and click OK.

4 On the Toolbars tab, select the check box on the Toolbars list to display the new toolbar in the workspace.

Application object

You can script the Application object's child objects to perform common tasks and to create toolbars dynamically.

Properties

ActiveDocument

Syntax `ActiveDocument: object (read-only)`

Description The current document. For details, see "ActiveDocument object" on page 236.

ApplicationType

Syntax `ApplicationType: integer (read-only)`

Description The current application type.

Sample ApplicationType script

```
//*********************************//
// Tests Application.ApplicationType property
// 0 = HomeSite
// 1 = CF Studio
// 2 = JRun Studio
//*********************************//
function Main(){
  var iAppType;
  var sMessage;

  with (Application) {
    if (IsColdFusionStudio)
      MessageBox('IsColdFusionStudio returns True',
        'Application Type', 0)
    else
      MessageBox('IsColdFusionStudio returns False',
        'Application Type', 0);
  iAppType = ApplicationType;
    switch(iAppType) {
            case 0: {
                sMessage = 'HomeSite';
                break;
            }
            case 1: {
                sMessage = 'ColdFusion Studio';
                break;
            }
            case 2: {
                sMessage = 'JRun Studio';
                break;
```

```
            }
            default: {
                sMessage = 'an unknown application type';

        }MessageBox('You are enjoying ' + sMessage + '.',
        'Application Type', 0);
}
```

AppPath

Syntax `AppPath: OleString (read-only)`

Description Path to application executable.

Example

```
function Main() {
var sPath;

with (Application) {
     sPath = AppPath; // Store the path in a variable
}
}
```

CurrentFolder

Syntax `CurrentFolder: OleString`

Description Path currently displayed in the local file list.

Example

```
function Main(){
  with (Application){
    CurrentFolder = 'C:\\InetPub\\wwwroot';
  }
}
```

CurrentView

Syntax `CurrentView: ITCurrentViewType`

Description Sets/gets the current view.

The following values are allowed:

```
1 - vwEditSource
2 - vwPreview (browse)
3 - vwHelp
```

The value 3 applies only if the Help tab is already visible in the application.

Example

```
function Main() {
with (Application){
     CurrentView = 2; // Set the view to Browse
}
}
```

Sample CurrentView script

```
//*********************************************************//
//  This script switches the views inside the application
//*********************************************************//
function Main () {
var sMessage;

with (Application) {

   CurrentView = 1;
   sMessage = "You are now in Edit View of your " +
        VersionText;
   MessageBox (sMessage, VersionText, 0);

   CurrentView = 2;
   sMessage = "You are now in Browse View of your " +
        VersionText;
   MessageBox (sMessage, VersionText, 0);

   CurrentView = 3;
   sMessage = "You are now in Help View of your " +
        VersionText;
   MessageBox (sMessage, VersionText, 0);
  }
}
```

DocumentCache

Syntax `DocumentCache: array of objects (read-only)`

Description For details, see “DocumentCache object” on page 247.

Example

```
function Main(){
  with (Application){
    // Save the first document
    if (DocumentCache(0).Modified){
      DocumentIndex = 0;
      ActiveDocument.Save();
    }
  }
}
```

DocumentCount

Syntax `DocumentCount: integer (read-only)`

Description Number of open documents.

Example

```
function Main() {
Var sMessage;

with (Application){
     sMessage = "There are ";
     sMessage = sMessage + sDocumentCount + " open.\n";
        // Get the number of open documents.
     sMessage = sMessage + sDocumentIndex + ".\n";
        // Get the index of the current document.
}
}
```

DocumentIndex

Syntax `DocumentIndex: Integer`

Description Tab index of current document.

Example

```
function Main() {
Var sMessage;

with (Application){
     sMessage = "There are ";
     sMessage = sMessage + sDocumentCount + " open.\n";
        // Get the number of open documents.
     sMessage = sMessage + sDocumentIndex + ".\n";
        // Get the index of the current document.
}
}
```

ExeName

Syntax `ExeName: OleString (read-only)`

Description filename of application executable, including path.

Example

```
function Main() {
Var sExeName;

with (Application){
    sExeName = ExeName;  // Store the path in a variable
}
}
```

Height

Syntax `Height: integer`

Description Height in pixels of main window.

Example
```
function Main() {
Var sMessage;

with (Application){
     sMessage = "Top-left corner of your Window has the
        following coordinates: \n";
     sMessage = sMessage + "Top: " + Top + "\n";
        // Get top
     sMessage = sMessage + "Left: " + Left + "\n\n";
        // Get left
     sMessage = sMessage + "And the following measurements: \n"
     sMessage = sMessage + "Width: " + Width + "\n";
        // Get Width
     sMessage = sMessage + "Height: " + Height + "\n";
        // Get Height
}
}
```

HInstance

Syntax `HInstance: integer (read-only)`

Description Instance handle of the application.

hWnd

Syntax `hWnd: integer (read-only)`

Description Handle to the main window.

IsColdFusionStudio

Syntax `IsColdFusionStudio: WordBool (read-only)`

Description Boolean. Returns `True` if the application is ColdFusion Studio or JRun Studio, `False` if HomeSite.

Example
```
function Main(){
  with (Application){
    if (IsColdFusionStudio){
      // Show CF Advanced toolbar
      ShowToolBar('CFML Advanced');
    }
  }
}
```

Left

Syntax `Left: integer`

Description Left (x-coordinate) of main window.

Example
```
function Main() {
Var sMessage;

with (Application){
     sMessage = "Top-left corner of your Window has the following
        coordinates: \n";
     sMessage = sMessage + "Top: " + Top + "\n";
        // Get top
     sMessage = sMessage + "Left: " + Left + "\n\n";
        // Get left
     sMessage = sMessage + "And the following measurements: \n"
     sMessage = sMessage + "Width: " + Width + "\n";
        // Get Width
     sMessage = sMessage + "Height: " + Height + "\n";
        // Get Height
}
}
```

ResourceTabShowing

Syntax `ResourceTabShowing: WordBool`

Description Boolean. Specifies whether the resource tab displays.

Example
```
function Main(){
  with (Application){
    // Toggle resource tab on/off
    ResourceTabShowing = !ResourceTabShowing;
  }
}
```

ResultsShowing

Syntax `ResultsShowing: WordBool`

Description Boolean. Specifies whether the results tab displays.

Example
```
function Main() {
with (Application){
      If (ResultShowing)
   ResultShowing = false;
        //If the result bar is showing - hide it
      Else
   ResultShowing = true;
        //Hide it otherwise
}
}
```

Top

Syntax `Top: integer`

Description Top (y-coordinate) of main window.

Example
```
function Main() {
Var sMessage;

with (Application){
     sMessage = "Top-left corner of your Window has the following
        coordinates: \n";
     sMessage = sMessage + "Top: " + Top + "\n";
        // Get top
     sMessage = sMessage + "Left: " + Left + "\n\n";
        // Get left
     sMessage = sMessage + "And the following measurements: \n"
     sMessage = sMessage + "Width: " + Width + "\n";
        // Get Width
     sMessage = sMessage + "Height: " + Height + "\n";
        // Get Height
}
}
```

VersionText

Syntax `VersionText: OleString (read-only)`

Description Application name and version.

Example
```
function Main() {
Var sMessage;

with (Application){
     If (IsColdFusionStudio){
         sMessage = "You are running ColdFusionStudio - " +
        VersionText;
         MessageBox(sMessage,"Sample",0);
    }
     Else{
         sMessage = "You are running HomeSite - " +
        VersionText;
         MessageBox("You are Running HomeSite", "Sample", 0);
    }
}
}
```

Width

Syntax `Width: integer`

Description Width in pixels of the main application window. Use this property with the Height property to return the size of the main window as well as to resize the window.

WindowState

Syntax `WindowState: integer`

Description Set and get window state.

The following values are allowed:

```
0 - Normal
1 - Minimized
2 - Maximized
```

Example

```
function Main(){
  var iNormal = 0;
  var iMinimized = 1;
  var iMaximized= 2;

  with (Application){
    // Toggle through all window states
    WindowState = iMaximized;
    Wait(1000);
    WindowState = iMinimized;
    Wait(1000);
    WindowState = iNormal;
  }
}
```

Methods

BringToFront

Syntax `BringToFront();`

Description Brings the main window to the front of other applications.

Example

```
function Main() {
with (Application){
     BringToFront();
}
}
```

BrowseText

Syntax `BrowseText(sText, BaseHREF: OleVariant);`

Description Displays the passed text in the internal browser. Use the `BaseHREF` parameter to interpret relative paths. For local files, `BaseHREF` is the folder that contains the file.

Example
```
function Main() {
Var sMessage;

sMessage = "You are viewing this text in the browse mode of your ";
sMessage = sMessage + VersionText;

with (Application){
     BrowseText (sMessage);
}
}
```

CloseAll

Syntax `CloseAll(wbPromptToSave: WordBool): WordBool;`

Description Closes all open documents. If `wbPromptToSave` is `True`, the user is prompted to save any changes. Returns `True` if successful, that is, the user didn't cancel if `wbPromptToSave` is `True`.

Example
```
function Main() {
with (Application){
     CloseAll();
}
}
```

ExecCommand

Syntax `ExecCommand(nCmdID: integer, nOptions: integer);`

Description Boolean. Execute a specific command based on its CommandID. For available commands, see "Table of CommandID values" on page 288. Use `nOptions` with cursor movement commands to determine whether text is selected during cursor movement (`nOptions = 1`) or unselected (`nOptions = 0`). For all other commands, pass `nOptions = 0`.

Example
```
function Main() {
with (Application){
     ExecCommand(3); // Executes an Open File command.
}
}
```

ExtractFileName

Syntax
```
ExtractFileName(const wsFile: WideString): WideString;
```

Description Returns only the file portion of the passed filename.

Example
```
function Main() {
Var sFullFilePath;
Var sFileName;

sFullFilePath = "C:/Temp/MyScript.js"

with (Application){
     sFileName = ExtractFileName(sFullFilePath);
        // Returns 'MySript.js'
}
}
```

ExtractFilePath

Syntax
```
ExtractFilePath(const wsFile: WideString): WideString;
```

Description Returns the path of the passed file (includes trailing '\').

Example
```
function Main() {
Var sFilePath;
Var sFullFilePath;

sFullFilePath = "C:/Temp/MyScript.js"

with (Application){
     sFilePath = ExtractFilePath(sFullFilePath);
        // Returns 'C:/Temp/'
}
}
```

GetApplicationSetting

Syntax
```
GetApplicationSetting(nSettingID: Integer);
```

Description Retrieves a specific application setting based on a SettingID.

Example
```
function Main(){
  var SET_CLOSE_PARA_TAGS = 80;

  with (Application){
    if (GetApplicationSetting(SET_CLOSE_PARA_TAGS) == 0){
      ActiveDocument.InsertText('<p>', false);
    }
    else {
```

```
      ActiveDocument.InsertText('<p></p>', false);
    }
  }
}
```

GetImageHeight

Syntax `GetImageHeight(const wsImageName: WideString): Integer;`

Description Returns the height in pixels of the passed image. Returns 0 on error.

Example

```
/*
  Tests GetImageHeight, GetImageWidth
  Looks for first GIF image in current directory
*/

function Main(){
  with (Application) {
    aFileObj= new ActiveXObject("Scripting.FileSystemObject");
    aFolder= aFileObj.GetFolder(CurrentFolder);
    aFiles= new Enumerator(aFolder.files);
    sExtToTest= 'gif';
    sFile= '';

    for (; !aFiles.atEnd(); aFiles.moveNext()){
      if (aFileObj.GetExtensionName(aFiles.item())
        == sExtToTest){
        sFile = aFiles.item();
        break;
      }
    }

    if (!sFile == ''){
      sMsg = sFile + '\n\n' + 'Height = ' +
        GetImageHeight(sFile) + '\n';
      sMsg = sMsg + 'Width = ' + GetImageWidth(sFile);
      MessageBox(sMsg, 'GetImageHeight/Width Test', 0);
    }
    else{
      MessageBox('No images found in current directory',
        'GetImageHeight/Width Test', 0);
    }
  }
}
```

GetImageSize

Syntax
```
GetImageSize(const wsImageFile: WideString; var nHeight,
nWidth: Integer): WordBool;
```

Description Boolean. Retrieves the size of the passed image. Returns `False` on error.

Example
```
function Main() {
Var sFullImagePath;
Var Height;
Var Width;

sFullImagePath = "C:/Temp/photos/MyPic.jpg";

with (Application){
     sFilePath = GetImageSize(sFullImagePath,Height,Width);
        //Store the image parameters in Height and Width

}
}
```

GetImageWidth

Syntax
```
GetImageWidth(const wsImageName: WideString):Integer;
```

Description Returns the width in pixels of the passed image. Returns `0` on error.

Example See the `GetImageHeight` example.

GetMemoryStatus

Syntax
```
GetMemoryStatus(iMemType);
```

Description Returns an integer value. On Windows 98, the values for 0, 1, and 2 are real numbers. On Windows NT, since there is no corresponding API call to get resource levels, these types always return the value 80%.

The following values are allowed:
```
0 - Available System resources    (%)
1 - Available GDI resources    (%)
2 - Available User resources    (%)
3 - General memory used    (%)
4 - Total physical memory    (bytes)
5 - Available physical memory    (bytes)
6 - Total swap file storage space    (bytes)
7 - Available swap file storage space    (bytes)
8 - Total virtual space    (bytes)
9 - Available virtual space    (bytes)
```

GetRelativePath

Syntax `GetRelativePath(const wsBaseURL, wsFolderURL: WideString): WideString;`

Description Returns the relative path of a folder given a base URL. For example,

```
GetRelativePath ("http://www.macromedia.com/", "http://
        www.macromedia.com/software/")
```

returns "products/".

Example

```
function Main() {
Var sFullPath1;
Var sRelativePath;
Var sFullPath2;

sFullPath1 = "http://www.macromedia.com/";
sFullPath2 = "http://www.macromedia.com/software";

with (Application){
     sRelativePath = GetRelativePath (sFullPath1,sFullPath12);
}
}
```

GetTabIndexForFile

Syntax `GetTabIndexForFile(const wsFile: WideString): Integer;`

Description Returns the index in the document tab of the passed file. Returns `-1` if the file is not open.

Example

```
function Main(){
  var  iIndex;

  with (Application){

    // Switch to index.htm if it is open
    iIndex = GetTabIndexForFile('D:\\Test\\index.html');
    if (iIndex > 0){
      DocumentIndex = iIndex;
    }
  }
}
```

GetURL

Syntax `GetURL(const wsURL: WideString): WideString;`

Description Retrieves a URL and returns its contents.

Example

```
function Main() {
Var sURL;
Var sContents;

sURL = "http://www.macromedia.com/";

with (Application){
      sContents = GetURL(sURL);
        // Storing HTML code of the URL in sContents
}
}
```

GetURLResponse

Syntax `GetURLResponse(const wsURL: WideString): WideString;`

Description Retrieves a URL and returns its contents.

Example

```
/*
  Tests GetURLResponse, GetURLStatusCode
*/

function Main(){
  with (Application) {
    sURL_1= 'http://www.macromedia.com';
    sURL_2= 'http://www.this_should_not_exist.com';

    sResponse = GetURLResponse(sURL_1);
    iStatus= GetURLStatusCode(sURL_1);

    sMsg = 'URL:  ' + sURL_1 + '\n\n';
    sMsg = sMsg + 'Response: ' + sResponse + '\n';
    sMsg = sMsg + 'Status:   ' + iStatus;

    MessageBox(sMsg, 'GetURLResponse/StatusCode Test', 0);

    sResponse = GetURLResponse(sURL_2);
    iStatus= GetURLStatusCode(sURL_2);

    sMsg = 'URL:  ' + sURL_2 + '\n\n';
    sMsg = sMsg + 'Response: ' + sResponse + '\n';
    sMsg = sMsg + 'Status:   ' + iStatus;

    MessageBox(sMsg, 'GetURLResponse/StatusCode Test', 0);

  }
}
```

GetURLStatus

Syntax
```
GetURLStatus(const wsURL: WideString;
var vResponse: OleVariant): Integer;
```

Description Returns the HTTP status code for the passed URL. The text of the server response is returned in the second parameter.

GetURLStatusCode

Syntax
```
GetURLStatusCode(const wsURL: WideString): Integer;
```

Description Returns the status code for the passed URL.

Example See the `GetURLResponse` example.

HideProgress

Syntax
```
HideProgress();
```

Description Hides the progress bar.

Example
```
function Main() {
with (Application){
      HideProgress();   // Hides the progress bar
}
}
```

HTMLConvertTagCase

Syntax
```
HTMLConvertTagCase(const wsHTML: WideString;
const wbUpperCase: WordBool): WideString;
```

Description Boolean. Converts the case of the passed HTML string. Retains the contents of `script`, `style` or `comment` tags, and retains the case of attribute values.

Example
```
function Main() {
Var sSource;

with (Application) {

if (HTTPProvider.State == 0){

  HTTPProvider.URL = InputBox(VersionText, "Please Enter the URL.",
        "http://www.yahoo.com");
  HTTPProvider.Get();    // Perform HTTP Get Request
  sSource = '';
  NewDocument (false);   //Open a new document inside the currently
        utilized application
  sSource = GetURL(sURL);
        sSource = HTMLConvertTagCase (sSource, true);
        // Converting the tags to uppercase.
```

```
    }
}
}
```

HTMLGetAttribute

Syntax `HTMLGetAttribute(const wsInTag, wsAttr: WideString): WideString;`

Description Returns the value for a particular attribute of a tag. For example,

`HTMLGetAttribute("<TABLE WIDTH=100>", "WIDTH");`

returns 100.

Example

```
function Main(){
  var sWidth;

  with (Application){
    // Get width attribute of a table tag
    sWidth = HTMLGetAttribute("<table width=100>", "width");
  }
}
```

HTMLGetTitle

Syntax `HTMLGetTitle(const wsFile: WideString): WideString;`

Description Returns the contents of an HTML file's `title` tag. This only operates on local files.

Example

```
function Main() {
Var sFile;
Var sTitle;
sFile = "C:\\Temp\index.html";

with (Application) {

     sTitle = HTMLGetTitle (sFile);
        // Store the contents inside <INDEX> in sTitle

}
}
```

InputBox

Syntax `InputBox(const wsCaption, wsPrompt, wsDefault: WideString): WideString;`

Description Displays a dialog box for obtaining user input.

Example

```
function Main() {
Var sInput;
Var sOutput;
```

```
with (Application) {
     sInput = InputBox(VersionText, "What is your name?", "Alex");
     sOutput  = "I know you, your name is " + sInput + ".";
     MessageBox (sOutput, VersionText, 0);
}
}
```

InstallParserScript

Syntax `InstallParserScript(const wsScriptFile, wsFileExtAssoc: WideString): WordBool;`

Description Boolean. Returns `False` on error. Installs a parser (color-coding) script and associates it with the passed list of semicolon-separated file extensions. If an existing parser is assigned to any of these extensions, they are removed from the existing parser and assigned to the new one. The parser script is copied from the passed location to the application \Parsers subdirectory.

Example

```
function Main(){
  with (Application){
    InstallParserScript('D:\\Download\\XHTML_2.scc', 'xhtml');
  }
}
```

IsFileModified

Syntax `IsFileModified(sFile: OleVariant): WordBool;`

Description Boolean. Returns `True` if the passed file is open in the Document tab and was modified.

Example

```
function Main(){
  with (Application){
    // Save current file if it is modified
    if (IsFileModified(ActiveDocument.FileName)){
      ActiveDocument.Save();
    }
  }
}
```

IsFileOpen

Syntax `IsFileOpen(sFile: OleVariant): WordBool;`

Description Boolean. Returns `True` if the passed file is open in the Document tab.

Example

```
function Main(){
  sFile = 'D:\\Test\\index.html';

  with (Application){
    if (!IsFileOpen(sFile)){
      OpenFile(sFile);
```

```
    }
  }
}
```

LogMemoryStatus

Syntax `LogMemoryStatus(const wsLogFile, wsDescrip: WideString);`

Description Writes the current memory status to a log file, `wsLogFile: logfilename.` Creates the file if it does not exist, otherwise appends status to the file. The description text for the entry is entered in `wsDescrip: text`.

Example

```
function Main(){
  with (Application){
    LogMemoryStatus('D:\\Test\\MemLog.txt', 'Application Start');
  }
}
```

MessageBox

Syntax `MessageBox(const wsText, wsCaption: WideString, nType: Integer): Integer;`

Description Displays a message dialog box for obtaining a user response. The `nType` parameter determines the type of dialog box displayed, and uses a combination of the following sets of values:

```
MB_ICONINFORMATION = 64
MB_ICONWARNING = 48
MB_ICONQUESTION= 32
MB_ICONSTOP= 16
MB_ABORTRETRYIGNORE= 2
MB_OK = 0 (Default)
MB_OKCANCEL = 1
MB_RETRYCANCEL = 5
MB_YESNO = 4
MB_YESNOCANCEL = 3
```

The function's result contains the ID of the button that the user clicked.

The following ID values are allowed:

```
IDOK = 1
IDCANCEL = 2
IDABORT = 3
IDRETRY = 4
IDIGNORE = 5
IDYES = 6
IDNO = 7
IDCLOSE = 8
```

Example

```
function Main() {
Var sInput;
Var sOutput;

with (Application) {
     sInput = InputBox(VersionText, "What is your name?", "Alex");
     sOutput  = "I know you, your name is " + sInput + ".";
     MessageBox (sOutput, VersionText, 0);
}
}
```

NewDocument

Syntax

```
NewDocument(wbUseDefaultTemplate: WordBool);
```

Description Boolean. Creates a new document, optionally from the default template.

Example

```
function Main(){
  with (Application){
    NewDocument(true);
  }
}
```

NextDoc

Syntax

```
NextDoc();
```

Description Moves to the next document in the Document tab. If the last document is showing, wraps to the first.

Example

```
function Main() {
Var sMessage;

sMessage = "Hello world!";

with (Application) {
     NewDocument(false);
     NextDoc();
     ActiveDocument.InsertText(sMessage);   // Moving to the
        newly-created document
}
}
```

OpenFile

Syntax

```
OpenFile(const wsFile: WideString): WordBool;
```

Description Boolean. Opens the passed file. Returns `True` if the file opens or is already open. Passing an empty string to `OpenFile` displays the Open File dialog box, which enables the user to select the files to open.

When using this method in JScript, you must escape backslashes inside a string. For example, in `Application.OpenFile("C:\\Documents\\MyFile.htm");` each backslash is preceded by an additional backlash.

Example

```
function Main() {
Var sFile;
Var bResult;

sFile = "C:\\Temp\myDoc.txt"

with (Application) {
      If(OpenFile(sFile))
  MessageBox("File opened successfuly.", VersionText);
      Else
  MessageBox("File does not exist or already open.", VersionText);
}
}
```

PreviousDoc

Syntax `PreviousDoc();`

Description Moves to the previous document in the Document tab. If the first document is showing, wraps to the last.

Example

```
function Main(){
  with (Application){
    // Create a new blank document
    NewDocument(false);
    // Move back to previous file
    PreviousDoc();
  }
}
```

Quit

Syntax `Quit();`

Description Attempts to exit from or quit the program. Prompts the user to save any unsaved documents prior to quitting.

Example

```
function Main(){
  var MB_YESNO = 4;
  var IDYES = 6;

  with (Application){
    if (MessageBox('Exit HomeSite?', 'Confirmation Message',
        MB_YESNO) == 6){
      Quit();
    }
  }
}
```

RunCodeSweeper

Syntax `RunCodeSweeper();`

Description Runs the CodeSweeper on the current document using the active CodeSweeper. To change the active CodeSweeper, use `SetActiveCodeSweeper`.

Example
```
function Main() {
with (Application) {
     RunCodeSweeper ();
}
}
```

SaveAll

Syntax `SaveAll(): WordBool;`

Description Boolean. Saves all open documents. Returns `True` if successful.

Example
```
function Main() {
with (Application) {
     SaveAll ();
}
}
```

SaveResultsToFile

Syntax `SaveResultsToFile(const wsFile): WideString;`

Description Saves the contents of the active Results window to the named file.

Example
```
function Main() {
with (Application) {
     SaveResultsToFile();
}
}
```

SendToBack

Syntax `SendToBack();`

Description Sends the main window behind the other applications.

Example
```
function Main() {
with (Application) {
     SendToBack ();
}
}
```

SetActiveCodeSweeper

Syntax `SetActiveCodeSweeper(const wsFileName: WideString): WordBool;`

Description Changes the active CodeSweeper format file.

Example
```
function Main(){
  var sCS_File = 'C:\\Program Files\\Macromedia\\ColdFusion Studio
        5\\Extensions\\CodeSweepers\\WebXML.vtm';

  with (Application){
    SetActiveCodeSweeper(sCS_File);
  }
}
```

SetActiveResults

Syntax `SetActiveResults(resType: TCurrentResultsType);`

Description Boolean. Sets the active page in the Results tab.

The following values are allowed:

```
resSearch
resValidator
resLinks
resThumbnails
```

Example
```
function Main() {
with (Application) {
     SetActiveResults ();
}
}
```

SetApplicationSetting

Syntax `SetApplicationSetting(nSettingID: Integer ovSettingVal: OleVariant);`

Description Sets a specific application setting based on its SettingID.

Example
```
function Main(){
  var SET_EDITOR_FONTNAME= 300;

  with (Application){
    SetApplicationSetting(SET_EDITOR_FONTNAME, 'Lucida Console');
  }
}
```

SetFileTabFolder

Syntax `SetFileTabFolder(iTab: integer; sFolder: string);`

Description Sets the active folder for each of the Files tabs.

Example
```
function Main(){
  var sFolder1    = 'C:\\Program Files\\Bradbury\\TopStyle2';
  var sFolder2    = 'C:\\InetPub\\wwwroot';

  with (Application) {
        SetFileTabFolder(1, sFolder1);
        SetFileTabFolder(2, sFolder2);
  }
}
```

SetProgress

Syntax
```
SetProgress(nProgress: Integer);
```

Description Sets the position of the progress bar in the status area.

The values 1-100 are allowed:

Example
```
function Main() {
with (Application) {
     SetProgress (15);
}
}
```

SetStatusText

Syntax
```
SetStatusText(sMessage: OleString);
```

Description Sets the text that displays in the status area.

Example
```
function Main() {
with (Application) {
      SetStatusText("Progress Indicator: ");
}
}
```

ShellToApp

Syntax
```
ShellToApp(const wsAppFileName: WideString): WordBool;
```

Description Executes an external application. Returns `True` if application launched successfully. You can include command lines in the filename parameter, so the following syntax is valid:

```
Application.ShellToApp("notepad.exe " +
        Application.ActiveDocument.Filename)
```

ShellToAppAndWait

Syntax `ShellToAppAndWait(const wsAppFileName: WideString);`

Description Boolean. Same as `ShellToApp` but waits for the external program to close before returning. The application is locked until `ShellToAppAndWait` returns, so use this method with caution.

Example
```
function Main(){
  with (Application){
    // Edit current document in notepad and then reload it
    ShellToAppAndWait('notepad.exe ' + ActiveDocument.Filename);
    ActiveDocument.Reload(false);
  }
}
```

ShowProgress

Syntax `ShowProgress();`

Description Shows the progress bar.

Example
```
function Main() {
with (Application) {
      ShowProgress();
}
}
```

ShowThumbnails

Syntax `ShowThumbnails(sFolder: OleString);`

Description Shows thumbnails for all images in the passed folder.

Example
```
function Main(){
  with (Application){
    ShowThumbnails(CurrentFolder);
  }
}
```

StatusError

Syntax `StatusError(const wsMsg: WideString);`

Description Displays an error message in the status bar. Message appears on a red background and displays for at least 5 seconds.

Example
```
function Main(){
  with (Application){
    if (ActiveDocument.Modified){
      StatusError('Current document is modified');
    }
  }
}
```

StatusWarning

Syntax `StatusWarning(const wsMsg: WideString);`

Description Displays a warning message in the status bar. Message appears on a blue background and displays for at least 5 seconds.

Example
```
function Main(){
  with (Application){
    if (ActiveDocument.ReadOnly){
      StatusWarning('Current document is read-only');
    }
  }
}
```

TagCase

Syntax `TagCase(const wsTag: WideString): WideString;`

Description Changes the case of the passed string based on the "Lowercase all inserted tags" setting in the Options > Settings > Markup Languages panel. Does not modify the case of attribute values.

Example
```
function Main(){
  with (Application){
    ActiveDocument.InsertText(TagCase('<a href="http://
        www.macromedia.com"'), false);
  }
}
```

ToolbarDir

Syntax `ToolbarDir: WideString (read-only);`

Description Returns the path where toolbar files are located.

Example
```
function Main(){
  with (Application){
    CurrentFolder = ToolbarDir;
  }
}
```

Wait

Syntax `Wait(nMilliseconds: Integer);`

Description Pauses for given number of milliseconds. Use Wait to enable scripts to execute loops, yet still allow access to the UI. Without the call to Wait in the loop, the application appears locked and the user cannot change views.

Example The following JScript sample waits for the user to return to edit source view:

```
var app = Application;
while (app.CurrentView != 1) {
  app.Wait(100);
}
```

This is the same sample code in VBScript:

```
set app = Application
while app.CurrentView  1
  app.Wait (100)
wend
```

Toolbar and toolbutton methods

This section describes the toolbar manipulation methods available in the Application object.

A unique name identifies each toolbar. The name of the toolbar displays in the title bar caption when the toolbar is not docked. Toolbars are loaded from files in the toolbar directory, which can be obtained from the `ToolbarDir` property. The toolbar name is the same as its filename without the path or extension. For example, if the toolbar filename is Custom.tbr, then the toolbar name is Custom.

When you create a toolbutton, remember that a toolbutton label is limited to two characters.

AddAppToolbutton

Syntax `AddAppToolbutton(wsToolbarName, wsExeFile, wsCmdLine, wsHint: WideString): WordBool;`

Description Boolean. Adds a toolbutton for an external application to the passed toolbar. Fails if the toolbar does not exist or if the toolbutton could not be added. Returns `True` if the same toolbutton (based on `wsExeFile` and `wsCmdLine`) already exists on the toolbar, but does not add a duplicate toolbutton.

Example

```
function Main(){
  with (Application){
    AddAppToolbutton('Standard', 'Notepad.exe', '', 'NotePad');
  }
}
```

AddScriptToolbutton

Syntax
```
AddScriptToolbutton(wsToolbarName, wsScriptFile, wsHint, wsCaption,
wsImageFile: WideString): WordBool;
```

Description Boolean. Adds a script toolbutton (executes passed JScript or VBScript file when clicked) to the passed toolbar. Fails if toolbar does not exist. Returns `True` if a toolbutton already exists, but does not add a duplicate toolbutton.

Example
```
function Main() {
Var sToolBarName = "MyToolBar";

with (Application) {
     CreateToolbar (sToolBarName);
     AddScriptToolbutton (sToolBarName, "C:\\temp\scripts\hello.js",
        "hello.js",'A', "");
     ShowToolBar(sToolBarName);
}
}
```

AddTagToolbutton

Syntax
```
AddTagToolbutton(wsToolbarName, wsTagStart, wsTagEnd, wsHint, wsCaption,
wsImageFile: WideString): WordBool;
```

Description Boolean. Adds a tag toolbutton (inserts tag pair when clicked) to the passed toolbar. Fails if toolbar does not exist. Returns `True` if a toolbutton already exists, but does not add a duplicate toolbutton.

Example
```
function Main(){
  with (Application){
    AddTagToolButton('Common', TagCase('<blockquote>'),
        TagCase('</blockquote>'), 'Block quote', 'BQ', '');
  }
}
```

AddVTMToolbutton

Syntax
```
AddVTMToolbutton(wsToolbarName, wsScriptFile, wsHint, wsCaption,
wsImageFile: WideString): WordBool;
```

Description Boolean. Adds a VTM toolbutton (displays passed VTM dialog box when clicked) to the passed toolbar. Fails if the toolbar does not exist. Returns `True` if a toolbutton already exists, but does not add a duplicate toolbutton.

Example
```
function Main(){
  var sVTM_File = 'C:\\Program Files\\Macromedia\\ColdFusion Studio
        5\\Extensions\\TagDefs\\XHTML\\blockquote.vtm';
```

```
  with (Application){
    AddVTMToolbutton('Common', sVTM_File, 'Block quote', 'BQ', '');
  }
}
```

CreateToolbar

Syntax `CreateToolbar(wsToolbarName: WideString): WordBool;`

Description Boolean. Creates a new, undocked toolbar of the passed name. Fails if the toolbar of the same name already exists.

Example

```
function Main() {
Var sToolBarName = "MyToolBar";

with (Application) {
     CreateToolbar (sToolBarName);
     ShowToolBar(sToolBarName);
}
}
```

DeleteToolbar

Syntax `DeleteToolbar(wsToolbarName: WideString): WordBool;`

Description Boolean. Physically deletes the toolbar. Fails if the toolbar does not exist or if the toolbar is one of the built-in toolbars. Works only on custom toolbars; built-in toolbars can be hidden, but not deleted.

Example

```
function Main() {
Var sToolBarName = "MyToolBar";

with (Application) {
     DeleteToolbar (sToolBarName);
}
}
```

HideToolbar

Syntax `HideToolbar(wsToolbarName: WideString): WordBool;`

Description Boolean. Hides a toolbar. Fails if the toolbar does not exist.

Example

```
function Main() {
Var sToolBarName = "MyToolBar";

with (Application) {
     HideToolbar (sToolBarName);
}
}
```

SetToolbarDockPos

Syntax
```
SetToolbarDockPos(wsToolbarName: WideString; nDockPos: Integer): Word-
Bool;
```

Description Boolean. Sets the docking position of the toolbar. Fails if the toolbar does not exist.

The following values for `nDockPos` are allowed:

```
1 = Top
2 = Bottom
3 = Left
4 = Right
```

Example
```
function Main(){
  with (Application){
    SetToolbarDockPos('Standard', 2);
  }
}
```

ShowToolbar

Syntax
```
ShowToolbar(wsToolbarName: WideString): WordBool;
```

Description Boolean. Displays a toolbar if it is not already showing. Fails if the toolbar does not exist.

Example
```
function Main() {
Var sToolBarName = "MyToolBar";

with (Application) {
     CreateToolbar (sToolBarName);
     ShowToolBar(sToolBarName);
}
}
```

ToolbarExists

Syntax
```
ToolbarExists(wsToolbarName: WideString): WordBool;
```

Description Boolean. Returns `True` if the passed toolbar exists.

Example
```
function Main() {
with (Application) {

sToolBarName = InputBox (VersionText, "Enter the Toolbar name.",
        "MyToolbar");

  while (ToolbarExists(sToolBarName) != 0){
         sToolBarName = InputBox (VersionText, "Please choose
             another name.", "MyToolbar");
  }
}
}
```

Sample toolbar script 1

```
//*****************************************************//
// This script creates a toolbar named Apps if one does not exist,
// then adds two custom toolbuttons to it. The first toolbutton
// launches Windows Explorer, the second one opens Windows Explorer
// at the current folder in the Editor.
//*****************************************************//

function Main() {
  var TB_NAME = 'Apps';
  var app = Application;

  if (!app.ToolbarExists(TB_NAME)) {
    app.CreateToolbar(TB_NAME);
  }
  app.AddAppToolbutton(TB_NAME, 'c:\\windows\\explorer.exe', ",
  'Explorer');
  app.AddAppToolbutton(TB_NAME, 'c:\\windows\\explorer.exe',
    app.CurrentFolder, 'Explorer - Current Folder');
```

Here's the same code in VBScript:

```
Sub Main
  const TB_NAME = "Apps"
  Dim app

  set app = Application
  if not app.ToolbarExists(TB_NAME) then
    app.CreateToolbar TB_NAME
  end if

  app.AddAppToolbutton TB_NAME, "c:\windows\explorer.exe", "",
    "Explorer"
  app.AddAppToolbutton TB_NAME, "c:\windows\explorer.exe",
    app.CurrentFolder, "Explorer - Current Folder"
End Sub
```

Sample toolbar script 2

```
//*************************************************************//
// This script creates a toolbar which is capable of executing
// all of the scripts contained in the Document Cache
//*************************************************************//

function Main () {

var sToolBarName;
var Result;
var count;
var fname;
var fnamepath;

  with (Application) {fnamepath
     sToolBarName = InputBox (VersionText, "Enter the Toolbar name.",
        "MyToolbar");

     while (ToolbarExists(sToolBarName) != 0){
           sToolBarName = InputBox (VersionText, "Please chose another
       name.", "MyToolbar");
      }

     CreateToolbar (sToolBarName);
     ShowToolBar(sToolBarName);
     count = 0;

     while (count <= (DocumentCount-1)){

         fnamepath = DocumentCache(count).FileName
         fname = ExtractFileName(fnamepath);
         AddScriptToolbutton (sToolBarName, fnamepath, fname, count,
        "");
         count ++;

     }
  }
}
```

ActiveDocument object

Use the `ActiveDocument` object to refer to the document currently displayed in the Editor. To access an open document that is not active, use the `Application.DocumentCache` object.

Properties

CanRedo

Syntax `CanRedo: WordBool (read-only)`

Description Boolean. Returns `True` if changes can be re-done.

Example

```
function Main() {
var sMessage;

sMessage = "CanRedo : ";

with (Application) {
     If(ActiveDocument.CanRedo)
        sMessage = sMessage + "Yes";
     Else
        sMessage = sMessage + "No";

   MessageBox(sMessage, VersionText, 0);

}
}
```

CanUndo

Syntax `CanUndo: WordBool (read-only)`

Description Boolean. Returns `True` if changes can be undone.

Example

```
function Main() {
var sMessage;

sMessage = "CanUndo : ";

with (Application) {
     If(ActiveDocument.CanUndo)
        sMessage = sMessage + "Yes";
     Else
        sMessage = sMessage + "No";

   MessageBox(sMessage, VersionText, 0);
}
}
```

CaretPosX

Syntax `CaretPosX: integer (read-only)`

Description The X-axis caret position.

Example
```
function Main(){
  with (Application){
    if (ActiveDocument.CaretPosX > 1){
      ActiveDocument.CursorLineStart(false);
    }
  }
}
```

CaretPosY

Syntax `CaretPosY: integer (read-only)`

Description The Y-axis caret position.

Example
```
function Main(){
  with (Application){
    if (ActiveDocument.CaretPosY > 1){
      ActiveDocument.CursorDocStart(false);
    }
  }
}
```

Filename

Syntax `Filename: OleString (read-only)`

Description filename of the current document.

Example
```
function Main() {
var sMessage;

sMessage = "Your filename is: ";

with (Application) {

     sMessage = sMessage + ActiveDocument.Filename;

     MessageBox(sMessage, VersionText, 0);
}
}
```

LineCount

Syntax `LineCount: integer (read-only`

Description Number of lines in the current document.

Example
```
function Main() {
var sMessage;

sMessage = "Your file line count is: ";

with (Application) {
     sMessage = sMessage + ActiveDocument.LineCount;
     MessageBox(sMessage, VersionText, 0);
}
}
```

Lines

Syntax `Lines(Index: integer) OleString`

Description Gets and sets the text of the line at the passed index. Iterating through a document using the Lines property might be slow, especially for large documents. For best results, only use Lines to evaluate single lines of text. If you must use Lines to update many lines, you can increase performance by wrapping the update in a `BeginUpdate..EndUpdate` block.

Example
```
function Main(){
  with (Application){
    ActiveDocument.Lines(0) = '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
        4.01 Transitional//EN">';
  }
}
```

Modified

Syntax `Modified: WordBool (read-only)`

Description Boolean. Returns `True` if the document changed since it was last saved.

Example
```
function Main() {
var sMessage;

sMessage = "Modified : ";

with (Application) {
     If(ActiveDocument.Modified)
        sMessage = sMessage + "Yes";
     Else
        sMessage = sMessage + "No";
```

```
   MessageBox(sMessage, VersionText, 0);
}
}
```

ReadOnly

Syntax `ReadOnly: WordBool (read-only)`

Description Boolean. Returns `True` if the current document is read-only.

Example

```
function Main(){
  with (Application){
    if (ActiveDocument.ReadOnly){
      MessageBox('Current document is Read-Only', 'Information', 0);
    }
  }
}
```

SelLength

Syntax `SelLength: integer`

Description Gets and sets the length of the current selection.

Example

```
function Main() {
var sMessage;

sMessage = "The length of the selected text of your document is ";

with (Application) {
     sMessage = sMessage + ActiveDocument.SelLength;
     MessageBox(sMessage, VersionText, 0);
}
}
```

SelStart

Syntax `SelStart: integer`

Description Gets and sets the start of the current selection.

Example

```
function Main(){
  with (Application){
    with (ActiveDocument){
      // Select entire document
      SelStart = 0;
      CursorDocEnd(true);
    }
  }
}
```

SelText

Syntax `SelText: OleString`

Description Gets and sets the text in the current selection.

Example
```
function Main() {
var sMessage;

sMessage = "The length of the selected text of your document is ";

with (Application) {
     sMessage = sMessage + ActiveDocument.SelText;
     MessageBox(sMessage, VersionText, 0);
}
}
```

TabIndex

Syntax `TabIndex`

Description Gets and sets the tab index of the document tab.

Example
```
function Main(){
  with (Application){
    if (ActiveDocument.TabIndex != 0){
      DocumentIndex = 0;
    }
  }
}
```

Text

Syntax `Text: OleString`

Description Gets and sets the complete document text.

Methods

BeginUpdate

Syntax `BeginUpdate();`

Description Turns off screen updating for the current document. This is useful if your script needs to make several changes to the current document at once—turning off screen updating during the procedure might significantly speed up the process. To turn on updating again, use `EndUpdate`.

Use `BeginUpdate...EndUpdate` with caution. If you fail to call `EndUpdate` after a call to `BeginUpdate`, or if the script crashes before `EndUpdate` is called, the user cannot view any changes made in the Editor.

Clear

Syntax `Clear();`

Description Clears all text from the current document.

Close

Syntax `Close(wbPromptToSave: WordBool): WordBool;`

Description Boolean. Closes the current document. If `wbPromptToSave` is `True`, the user is prompted to save any changes. Returns `True` if the document was closed (that is, the user didn't cancel saving changes).

Cursor

Syntax `Cursor_ (wbSelect: WordBool);`

Description Boolean. Positions the cursor. If `wbSelect` is `True`, then the current selection is extended to the new cursor position.

The following values are allowed:

`CursorLeft, CursorRight, CursorWordLeft, CursorWordRight, CursorDown, CursorUp, CursorPageDown, CursorPageUp, CursorDocStart, CursorDocEnd, CursorLineStart, CursorLine End.`

Example

```
function Main(){
  with (Application){
    // Select the entire line
    ActiveDocument.CursorLineStart(false);
    ActiveDocument.CursorLineEnd(true);
  }
}
```

EndUpdate

Syntax `EndUpdate();`

Description Turns on screen updating for the current document.

Example

```
function Main(){
  var iTags = 0;
  with (Application){

    ActiveDocument.BeginUpdate();
    ActiveDocument.CursorDocStart(false);

    try{
      while (ActiveDocument.GotoNextStartTag(false)){
        iTags++;
      }
```

```
    }
    finally{
      ActiveDocument.EndUpdate();
    }

    MessageBox('Total tags: ' + iTags, 'Tag Count', 0);
  }
}
```

GetCaretPos

Syntax `GetCaretPos(var x, y: integer);`

Description Returns the caret pos (x=column, y=line).

GetCurrentChar

Syntax `GetCurrentChar(): OleVariant;`

Description Returns the current character.

GetNextChar

Syntax `GetNextChar(): OleVariant;`

Description Returns the next character. If you use this function, along with `GetPreviousChar`, in long loops, the code can run slowly.

GetPreviousChar

Syntax `GetPreviousChar(): OleVariant;`

Description Returns the previous character.

GetTagAtCursor

Syntax `GetTagAtCursor(wbSelect: WordBool): WideString;`

Description Boolean. Returns the tag in which the cursor is currently located in the Editor. If a tag cannot be identified, the method returns an empty string.

If `wbSelect` is `True`, the tag is highlighted in the document.

Example

```
// Displays a message box with the current tag
function Main() {
with (Application) {
MessageBox(ActiveDocument.GetTagAtCursor(false), 'Current Tag At
        Cursor', 0);
}
}
```

GotoNextEndTag

Syntax `GotoNextEndTag(wbSelect: WordBool): WordBool;`

Description Boolean. Moves the next end tag, and selects it if `wbSelect` is `True`. Returns `False` if no tag found.

GotoNextStartTag

Syntax `GotoNextStartTag(wbSelect: WordBool): WordBool;`

Description Boolean. Moves the next starting tag and selects it if `wbSelect` is `True`. Returns `False` if no tag is found.

GotoPreviousEndTag

Syntax `GotoPreviousEndTag(wbSelect: WordBool): WordBool;`

Description Boolean. Moves the previous end tag and selects it if `wbSelect` is `True`. Returns `False` if no tag is found.

Example

```
function Main(){
  with (Application){
    ActiveDocument.GotoPreviousEndTag(false);
  }
}
```

GotoPreviousStartTag

Syntax `GotoPreviousStartTag(wbSelect: WordBool): WordBool;`

Description Boolean. Moves the previous starting tag and selects it if `wbSelect` is `True`. Returns `False` if no tag is found.

Example

```
function Main(){
  with (Application){
    ActiveDocument.GotoPreviousStartTag(true);
  }
}
```

Indent

Syntax `Indent();`

Description Indents the current selection.

InsertTag

Syntax `InsertTag(sStartTag, sEndTag: OleVariant; wbOverwriteSelection: WordBool);`

Description Boolean. Inserts the passed tag pair at the current cursor position, overwriting the selection if `wbOverwriteSelection` is `True`. The cursor is positioned between the start and end tags after this operation. If `wbOverwriteSelection` is `False`, the tags surround the current selection.

Example
```
function Main(){
  with (Application){
    ActiveDocument.CursorDocEnd(false);
    ActiveDocument.InsertTag('<a href="http://www.macromedia.com">',
        '</a>', true);
  }
}
```

InsertText

Syntax
```
InsertText(InsertStr: OleVariant; wbOverwriteSelection: WordBool);
```

Description Boolean. Inserts the passed string at the current cursor position. If `wbOverwriteSelection` is `True`, this function overwrites the selection.

LastSavedDate

Syntax
```
LastSavedDate();
```

Description Returns the datetime value for the last save of the current document file.

Example
```
function Main(){
  with (Application){
    SetStatusText('Last save: ' + ActiveDocument.LastSavedDate());
  }
}
```

Print

Syntax
```
Print(wbNoPrompt: WordBool);
```

Description Boolean. Prints the current document. If `wbNoPrompt` is `False`, prompts the user for print settings.

Redo

Syntax
```
Redo();
```

Description Performs a single `redo` operation.

Reload

Syntax `Reload(wbPromptToSave: WordBool);`

Description Boolean. Reloads the current document. If `wbPromptToSave` is `True`, prompts the user to save changes.

ReplaceAll

Syntax `ReplaceAll(strSearch, strReplace: OleVariant; bMatchCase: WordBool): Integer;`

Description Boolean. Replaces all occurrences of `strSearch` with `strReplace`, matching case if `bMatchCase` is `True`. Returns the number of replacements made.

Save

Syntax `Save: WordBool();`

Description Boolean. Saves changes to the current document. Returns `True` if successful.

SaveAs

Syntax `SaveAs(wsFileName: widestring): WordBool;`

Description Boolean. Saves changes to the current document to the file specified in the `wsFileName` parameter.Returns `True` if successful.

If `wsFileName` is empty, the standard Save As dialog box displays. This function overwrites existing files, so use it with caution.

Example

```
function Main(){
  var sFile = '';

  // Insert a link and create the file in one step
  with (Application){
    sFile = InputBox('Add File Link', 'Enter a filename to create and
        link to:', sFile);

    if (sFile == ''){
      return;
    }

    ActiveDocument.InsertTag('<a href="' + sFile + '">', '</a>',
        false);
    NewDocument(true);
    ActiveDocument.SaveAs(CurrentFolder + '\\' + sFile);
  }
}
```

SelectAll

Syntax `SelectAll();`

Description Selects all the text in the current document.

SelectCurrentLine

Syntax `SelectCurrentLine();`

Description Highlights the current line.

SelectLine

Syntax `SelectLine(Index: Integer);`

Description Highlights the passed line.

SetCaretPos

Syntax `SetCaretPos(x, y: Integer);`

Description Sets the current column/line.

TextPosToEditorPos

Syntax `TextPosToEditorPos(var nPos:OleVariant):WordBool;`

Description Boolean. Converts an index in a text string in a VBScript script to the corresponding editor position. Takes into account tabs and newlines, which count as two characters in the text but only as one in the Editor. Returns the `nPos` parameter as an `OleVariant`, but its actual type is `integer`.

This method is only supported in VBScript.

Undo

Syntax `Undo();`

Description Performs a single undo operation.

Unindent

Syntax `Unindent();`

Description Removes the indent in the current selection.

DocumentCache object

It is very important to understand the DocumentCache object when using the VTOM. Although HomeSite+ for Dreamweaver MX enables you to open dozens of files at once, only the current document stays in memory. When a document becomes inactive, that is, when the user switches to a different document in the Document window, the previously current document is cached to conserve resources.

Every open document has an element in the `Application.DocumentCache` array. To refer to a specific cached document, use `Application.DocumentCache(Index)`, where `Index` is the index of the document in the Document tab.

Example

```
//*****************************************//
//This JScript shows how to loop through the array:
//*****************************************//
var app = Application;
for (idx = 0; idx < app.DocumentCount; idx++) {
  sFile = app.DocumentCache(idx).Filename;
  }
```

Example

```
//*****************************************//
//This VBScript shows how to loop through the array:
//*****************************************//
set app = Application
for idx = 0 to app.DocumentCount - 1
sFile = app.DocumentCache(idx).Filename
next
```

Example If you know the filename of an open document, you can retrieve its index by using the `Application.GetTabIndexForFile` function, like this:

```
var app = Application;
idx = app.GetTabIndexForFile('c:\docs\file.htm');
  bReadOnly = app.DocumentCache(idx).ReadOnly;
```

To access more information about a cached document, you must first make it the current document and refer to it using the Application object's `ActiveDocument` property. To do this, set the `Application.DocumentIndex` property to the index of the cached document.

Properties

Filename

Syntax `Filename: OleString (read-only)`

Description Filename of the cached document.

Example

```
function Main() {
var sMessage;

sMessage = "Your filename is: ";

with (Application) {
     sMessage = sMessage + DocumentCache (0).Filename;
}
}
```

Modified

Syntax `Modified: WordBool (read-only)`

Description Boolean. Returns `True` if the cached document changed since it was last saved.

Example

```
function Main() {
var sMessage;

sMessage = "Modified: ";

with (Application) {
     If(DocumentCache (0).Modified)
        sMessage = sMessage + "Yes";
     Else
        sMessage = sMessage + "No";

   MessageBox(sMessage, VersionText, 0);

}
}
```

ReadOnly

Syntax `ReadOnly: WordBool (read-only)`

Description Boolean. Returns `True` if the cached document is read-only.

Example

```
function Main() {
var sMessage;

sMessage = "Read-only: ";

with (Application) {
     If(DocumentCache (0).ReadOnly)
        sMessage = sMessage + "Yes";
     Else
        sMessage = sMessage + "No";

   MessageBox(sMessage, VersionText, 0);

}
}
```

Text

Syntax `Text: OleString (read-only)`

Description File contents of the cached document.

Example

```
function Main() {
var sMessage;

sMessage = "Your document contains the following text: \n";

with (Application) {
     sMessage = sMessage + DocumentCache (0).Text;
}
}
```

Project object

The `Project` object provides a basic set of properties and methods for scripting project maintenance tasks.

The `ProjectManager` and `DeploymentManager` objects extend the capabilities of the `Project` object. For more information, see "ProjectManager object" on page 252 and "DeploymentManager object" on page 258.

Properties

ActiveProjectFile

Syntax `ActiveProjectFile (read-only)`

Description Filename of the active project or empty string if no project is open.

Methods

AddFileToProject

Syntax
```
AddFileToProject(const wsFilename: WideString): WordBool;
```

Description Boolean. Adds the passed file to the active project. The main project folder or a subfolder of the main project must contain the file.

CloseProject

Syntax
```
CloseProject(wbCloseOpenFiles: WordBool): WordBool;
```

Description Boolean. Closes the active project, if any. If `wbCloseOpenFiles` is `True`, then all open files are closed.

OpenProject

Syntax
```
OpenProject(const wsProjectFile: WideString): WordBool;
```

Description Boolean. Opens the passed project file, making it the active project. Pass an empty string to display the Open Project dialog box.

RemoveFileFromProject

Syntax
```
RemoveFileFromProject(const wsFilename: WideString): WordBool;
```

Description Boolean. Removes the passed file from the active project.

ShowLastProjectError

Syntax `ShowLastProjectError();`

Description Displays an error message for the last project-related error.

UploadProject

Syntax `UploadProject(const wsTargetDir: WideString; const wbForceLCase, wbUploadOnlyNewer, wbEncryptCFML: WordBool): WordBool;`

Description Boolean. Uploads a project based on the passed criteria. The `wbEncryptCFML` parameter is specific to ColdFusion Studio.

UploadProjectDlg

Syntax `UploadProjectDlg(): WordBool;`

Description Boolean. Displays the upload project dialog box for the active project.

ProjectManager object

The `ProjectManager` object provides extensive and granular scripting capabilities.

This section groups the `PropertyManager` methods by function.

Properties

FileCount

Syntax `FileCount: Integer (read-only)`

Description The number of files in the project.

IsDirty

Syntax `IsDirty: WordBool (read-only)`

Description Boolean. Differences exist between the project as viewed in HomeSite+ for Dreamweaver MX and the underlying disk structure.

IsFileSelected

Syntax `IsFileSelected: WordBool (read-only)`

Description Boolean. Selects the file in the project.

Path

Syntax `Path: OleVariant (read-only)`

Description Default project path.

SelectedFile

Syntax `SelectedFile: OleVariant (read-write)`

Description Gets and sets variant property.

Methods

AddFile

Syntax `AddFile(Filename: OleVariant);`

Description Adds a file to the project.

AddFolder

Syntax `AddFolder(Folder: OleVariant; FolderType: TFolderType; FolderPath: OleVariant; Parent: OleVariant);`

Description Adds a new folder to the project root. To add a subfolder to an existing folder, use a folder-level method.

CheckedIn

Syntax `CheckedIn: WordBool;`

Description Boolean. Checks a project into source control.

CreateProject

Syntax `CreateProject(ProjectName: OleVariant; ProjectPath: OleVariant);`

Description Creates a new project.

DeployScriptList

Syntax `DeployScriptList: OleVariant;`

Description Lists scripts associated with the project.

DeployServerList

Syntax `DeployServerList: OleVariant;`

Description Lists servers associated with the project.

FolderList

Syntax `FolderList: OleVariant;`

Description Lists project subfolders.

Open

Syntax `Open(ProjectName: OleVariant);`

Description Opens a new project.

RemoveFile

Syntax `RemoveFile(Filename: OleVariant);`

Description Removes a file from the project.

RemoveFolder

Syntax `RemoveFolder(FolderName: OleVariant);`

Description Removes a folder from the project.

Save

Syntax `Save;`

Description Saves the current project to disk in WDDX format.

SelectFile

Syntax `SelectFile(Filename: OleVariant);`

Description Selects a file for source control operations.

Folder methods

FolderAddFile

Syntax `FolderAddFile(Folder: OleVariant; Filename: OleVariant);`

Description Add a file to the specified folder.

FolderAllFiles

Syntax `FolderAllFiles(Folder: OleVariant): OleVariant;`

Description Lists all files within the specified project.

FolderChangeType

Syntax `FolderChangeType(Folder: OleVariant; FolderType: TFolderType);`

Description Changes the folder type.

FolderContainsFile

Syntax `FolderContainsFile(Folder: OleVariant; Filename: OleVariant): WordBool;`

Description Boolean. Determines if a file is contained within a project folder.

FolderDeployTarget

Syntax `FolderDeployTarget(Folder: OleVariant; out FolderType: Integer; out FolderTarget: OleVariant);`

Description Gets the folder deployment target and type.

FolderFileCount

Syntax `FolderFileCount(Folder: OleVariant): Integer;`

Description Counts the number of files in the project folder.

FolderRemoveFile

Syntax `FolderRemoveFile(Folder: OleVariant; Filename: OleVariant);`

Description Removes file from specified folder.

FolderRenameFile

Syntax `FolderRenameFile(Folder: OleVariant; OldName: OleVariant; NewName: OleVariant);`

Description Renames a file within a project.

FolderSubFolderExists

Syntax `FolderSubFolderExists(Folder: OleVariant; SubFolderName: OleVariant): WordBool;`

Description Boolean. Determines if a subfolder exists.

FolderSubFolders

Syntax `FolderSubFolders(Folder: OleVariant): OleVariant;`

Description Lists all subfolders within the project folder.

FolderType

Syntax `FolderType(Folder: OleVariant): Integer;`

Description Returns the specified folder type.

Deployment methods

DeploymentScriptAdd

Syntax `DeploymentScriptAdd: OleVariant;`

Description Adds a deployment script to the project.

DeploymentScriptCount

Syntax `DeploymentScriptCount: Integer;`

Description Returns the number of scripts.

DeploymentScriptList

Syntax `DeploymentScriptList: OleVariant;`

Description Lists all project deployment scripts.

DeploymentScriptRemove

Syntax `DeploymentScriptRemove(ScriptName: OleVariant);`

Description Removes a deployment script from the project.

DeploymentServerAdd

Syntax `DeploymentServerAdd(ServerName: OleVariant; ServerNum: Integer);`

Description Adds a deployment server to the project.

DeploymentServerCount

Syntax `DeploymentServerCount: Integer;`

Description Returns the number of deployment servers.

DeploymentServerList

Syntax `DeploymentServerList: OleVariant;`

Description Lists deployment servers.

DeploymentServerRemove

Syntax `DeploymentServerRemove(ServerName: OleVariant);`

Description Removes a deployment server from a project.

DeploymentManager object

The `DeploymentManager` object is a scriptable interface to the Project Deployment engine. The `DeploymentManager` object provides a collection of methods and properties that enable you to write highly customized scripts to control the deployment process.

For examples of `DeploymentManager` Object syntax, see the "Sample deployment script" on page 263.

Properties

CreateFolder

Syntax `CreateFolder: WordBool (read-write)`

Description Boolean. Determines whether the deployment engine creates the missing folders on the target server.

EncryptCFML

Syntax `EncryptCFML: WordBool (read-write)`

Description Boolean. Determines whether the deployment engine encrypts all CFML files.

FolderCount

Syntax `FolderCount: Integer (read-only)`

Description The count of folders associated with the current open project.

ForceLowerCase

Syntax `ForceLowerCase: WordBool (read-write)`

Description Boolean. Determines whether the deployment engine forces lower-case filenames.

IsLocalDeployment

Syntax `IsLocalDeployment: WordBool (read-write)`

Description Boolean. lets you to perform a local deployment by overriding the assigned deployment server list. Uses the actual deployment pathnames assigned to the folders.

ServerCount

Syntax `ServerCount: Integer (read-only)`

Description Counts the deployment servers associated with the current open project, including any new servers that were added temporarily using the `AddServer` method.

UploadOnlyIfNewer

Syntax `UploadOnlyIfNewer: WordBool (read-write)`

Description Boolean. Determines whether the deployment engine only uploads a file if it has a newer date/time stamp than the target file.

Methods

Server login failures result in an automatic cancellation of the deployment.

AddServer

Syntax
```
AddServer(const wsServerName:WideString,ITServerType:Integer);
```

Description Temporarily adds a machine-level server to the list of deployment servers. This server does not become part of the project's stored deployment server list, but is added temporarily for custom deployment tasks.

The following ITServerType values are allowed:

```
1 = FTP
2 = RDS
```

CheckServerFolders

Syntax
```
CheckServerFolders(sServerName: WideString);
```

Description Inserted in the generated VBScript/JScript deployment scripts at the appropriate points, the CheckServerFolders method iterates through all of the assigned deployment folders for a project, verifying their existence on the target server (`sServerName`). Creates a folder if it does not exist.

Older scripts that do not contain the `CheckServerFolders` call will still work. The `CreateFolder` property gets passed to the internal `CopyFileExtended` call and tells `FileProxy` to create the directories that do not exist.

ClearServerList

Syntax
```
ClearServerList();
```

Description Clears the internal list of servers assigned to the project. Use this method to override the default project deployment server list.

CopyFile

Syntax `CopyFile(const SourceFile: WideString; const TargetFile: WideString): Integer;`

Description Copies the source file to the target file location.

CopyFileExtended

Syntax `CopyFileExtended(const SourceFile: WideString; const TargetFile: WideString; CreateFolder: WordBool; UploadOnlyIfNewer: WordBool; EncryptCFML: WordBool): Integer;`

Description Boolean. Copies the source file to the target file location and provides additional arguments.

CreateDir

Syntax `CreateDir(PathName: WideString): Integer;`

Description Directory creation primitive, returns 0 for success, or RDS error code for failure.

FileExists

Syntax `FileExists(const wsFileName: WideString): WordBool;`

Description Boolean. Checks to see if a file exists.

GetDeployServerName

Syntax `GetDeployServerName(nServer:integer): WideString;`

Description Returns the name of the server in the server list based on the index `nServer`. The internal server list array starts from a zero base.

GetDeployServerStatus

Syntax `GetDeployServerStatus(nServer:integer):WideString;`

Description Returns the index of the server in the server list based on the index `nServer`. The internal server list array starts from a zero base.

GetDeployTargetName

Syntax `GetDeployTargetName(const wsServerName:Widestring, const wsFolderName:Widestring, nFileIndex:Integer): WideString;`

Description Calculates the target deployment filename using the passed-in server name, folder name, and an integer representing the folder file to use. Use this method is mainly when iterating through all the files in a folder. It calculates a server file target path,

such as "Rds://localhost/D:/main/images/TestImage.jpg" which you can pass as the second argument to the `UploadFile` method.

GetFolderDeployPath

Syntax `GetFolderDeployPath(const wsFolderName:String): WideString;`

Description Returns the deployment path of the passed folder name. Use this path in conjunction with the `GetDeployServerName` and `GetFolderName` methods.

GetFolderFileCount

Syntax `GetFolderFileCount(const wsFolderName:String): Integer;`

Description Returns the number of files tracked by the passed project folder name.

GetFolderFileExt

Syntax `GetFolderFileExt(const wsFolderName: WideString; nIndex: Integer): WideString;`

Description Returns the extension of a folder file based on the passed folder name. For details, see “Project folder names” on page 263.

GetFolderFileName

Syntax `GetFolderFileName(const wsFolderName: WideString; nIndex: Integer): WideString;`

Description Returns the name of a folder file based on the passed folder name.

GetFolderName

Syntax `GetFolderName(nFolder:Integer): WideString;`

Description Returns the project folder name of the folder represented by the position index `nFolder`. Project folders are numbered consecutively.

GetLastErrorCode

Syntax `GetLastErrorCode(): Integer;`

Description Tests the result of an `UploadFile` call by returning the last error code as an integer. Use both `GetLastError` methods to programmatically abort a script or provide other forms of rudimentary error handling.

GetLastErrorMessage

Syntax `GetLastErrorMessage(): WideString;`

Description Tests the result of an `UploadFile` call by returning last associated RDS error message.

IsFolderDeployable

Syntax `IsFolderDeployable(const wsFolderName:WideString): WordBool;`

Description Boolean. Returns the deployment status of the passed folder name. Use this method to skip folders that are designated as Do Not Deploy folders.

OpenProject

Syntax `OpenProject(const wsProjectName: WideString);`

Description Opens the project specified in the passed `OleString`. The project specification must be a fully qualified path to the existing project file.

PathExists

Syntax `PathExists(const wsFolderName: WideString): WordBool;`

Description Boolean. Checks to see if a path exists.

SetDeployState

Syntax `SetDeployState(sServerName:WideString,bServerStatus: WordBool);`

Description Boolean. Enables and disables the deployment server specified by `sServerName` during deployment. This allows some servers to be turned on or off during the deployment process.

SetLogFileName

Syntax `SetLogFileName(sLogFileName: String);`

Description Changes the name of a generated deployment log file. The default name is Deployment.log and its default location is the main installation directory for HomeSite+ for Dreamweaver MX. Specify a fully qualified path and filename.

SetLogging

Syntax `SetLogging(bDoLogging: WordBool);`

Description Boolean. Turns logging on and off. This method only has an effect prior to opening a project for deployment.

UploadFile

Syntax `UploadFile(const wsFile:Widestring, sTargetFile: WideString);`

Description Uploads an individual file to the server. The first `OleString` represents the fully qualified path of an individual file to upload. The second `OleString` represents the fully qualified target name, such as rds://localhost/main/index.html.

UploadProject

Syntax `UploadProject(const wsProjectName: WideString);`

Description Uploads an entire project based on the fully qualified project name passed as an `OleString`. This triggers the default deployment engine processing loop which iterates through each server assigned to the project, each folder within the project, and uploads all the files contained within each folder. During this process, skips any folders that are configured as Do not deploy folders.

Project folder names

Project folder names are stored in the following format: Project/Folder/Subfolder1[/SubFolder2… SubFolderN], where:

Project represents the name of the project

Folder represents the topmost folder in the project

SubFolder1 represents a folder stored relative to Folder

SubFolder2..SubFolderN represent folders that are stored relative to SubFolder1 in a hierarchical fashion.

To use any of the existing folder-related `DeploymentManager` calls, you must pass the fully qualified folder name starting from the project name itself. For example, if you have a project called Project1 and a Main folder that contains an Images folder, and you must retrieve the count of how many files you have in the Images folder, the proper call is:

```
ICount = GetFolderFileCount("MyProject/Main/Images");
// Returns count of files in Images
```

Sample deployment script

```
// ====================================================================
// This sample script was generated by the Deployment Wizard,
// using the Project Element Iterator Script option.
// ====================================================================
// Project Name:
// Date/Time Generated:
```

```
// ====================================================================
// Server List:
//
// Local Deployment
// ====================================================================

function Main()
{
 var app = Application;

 var DeploymentManager = Application.DeploymentManager;

   var i,j,n,
        sServerName,sFolderName,sDeployPath,sFromFile,sTargetFile;

  //====================================================================
   // Logging Options
  //====================================================================
   DeploymentManager.SetLogging(true);
   DeploymentManager.SetLogFileName("C:\\Program
        Files\\Macromedia\\ColdFusion Studio 5\\Deployment.log");
  //====================================================================
   // End Logging Options
   // ================================================================
  // ====================================================================
   // Open the project...
  //====================================================================
   DeploymentManager.OpenProject("D:\\Projects\\Release Notes\\version
        5\\CFS\\Test1.apf");

  //====================================================================
   // Bypass servers and perform local deployment
  //====================================================================
   DeploymentManager.IsLocalDeployment = true;
  //====================================================================
   // Project Server Selections
  // ====================================================================
   // Project Server
   DeploymentManager.SetDeployState("",true);
  //====================================================================
   // Set Deployment Flags...
  //====================================================================
   DeploymentManager.CreateFolder = true;
   DeploymentManager.UploadOnlyIfNewer = true;
   DeploymentManager.EncryptCFML = false;
   DeploymentManager.ForceLowerCase = false;
  //====================================================================
   // Iterate through Deployment Servers
  //====================================================================
   for (i = 0; i <= DeploymentManager.ServerCount-1; i++) {
      if (DeploymentManager.GetDeployServerStatus(i)) {
         sServerName = DeploymentManager.GetDeployServername(i);
                  DeploymentManager.CheckServerFolders(sServerName);
```

```
                // ========================================================
                 //          Iterate through Project Folders
                 //========================================================
                 for (j=0; j <= DeploymentManager.FolderCount-1; j++) {
                    sFolderName = DeploymentManager.GetFolderName(j);
                    sDeployPath = DeploymentManager.GetFolderDeployPath(j);

                   //========================================================
                    //  isFolderDeployable reflects the Deployment Options you
                //chose for the folder
                  //========================================================
                    if (DeploymentManager.IsFolderDeployable(sFolderName)) {
                      // ======================================================
                       // Iterate through Folder Files
                     //========================================================
                       for (n=0; n <=
                DeploymentManager.GetFolderFileCount(sFolderName)-1; n++) {
                            sFromFile =
                DeploymentManager.GetFolderFileName(sFolderName,n);
                            sTargetFile =
                DeploymentManager.GetDeployTargetName(sServerName,sFolderName,
                n);
                            DeploymentManager.UploadFile(sFromFile,sTargetFile);
                       } // n...

                    } // if (IsFolderDeployable(sFolderName))...
                 } // j...
              } // if GetDeployServerStatus...
              } // i...

             DeploymentManager.CloseProject();


          }
```

HTTPProvider object

The HTTPProvider object is a general purpose HTTP protocol object. You can use it in VTOM scripts for low-level HTTP operations. The HTTPProvider object is an alternative to simplified HTTP-related methods, such as the GetURL method of the main Application object, which has a limited number of customizable HTTP parameters. Using the HTTPProvider object, you can initialize specific HTTP provider properties (such as Proxy, ProxyPort, Username, Password), and execute GET, POST, and HEAD HTTP method requests. For detailed information on many of the properties in this section, see http://www.w3.org/Protocols/http://www.w3.org/Protocols//a .

Properties

Agent

Syntax `Agent: OleVariant`

Description Sets and gets the identification of the client that initiates a request. Use this property to identify yourself as a client type or emulate a browser.

AuthorizationRequest

Syntax `AuthorizationRequest: OleVariant (read-only)`

Description The WWW-Authenticate response-header field. The 401 (unauthorized) response messages include this field. The field value consists of at least one challenge that indicates the authentication scheme(s) and parameters applicable to the Request-URI.

ContentLength

Syntax `ContentLength: Integer (read-only)`

Description Specifies the length of the received content stream.

ContentType

Syntax `ContentType: OleVariant (read-only)`

Description Specifies the MIME content type of the received content stream.

ContentTypePost

Syntax `ContentTypePost: OleVariant`

Description Sets and gets the Content-Type entity-header field, which indicates the media type of the Entity-Body sent to the recipient. For the HEAD method, indicates the media type that is sent if the request is a GET.

Cookie

Syntax `Cookie: OleVariant`

Description Sets and gets the Cookie header element. Use this property to send a set of client cookies to the server along the HTTP request.

Example **Sample cookie script**

```
function Main ()
{
var app = Application;
var httpPro = app.HTTPProvider;
httpPro.URL    = "http://127.0.0.1/GetCustomerRegistration.cfm";
httpPro.Cookie = 'Customer="John_Doe"; $Path="/myapp";Cust_ID="4567";
        $Path="/myapp"';
httpPro.Get();
}
```

DocName

Syntax `DocName: OleVariant (read-only)`

Description The document name segment from the requested URL.

LastResponse

Syntax `LastResponse: OleVariant (read-only)`

Description The most recent response content block when content stream is sent from the server in multiple responses.

Location

Syntax `Location: OleVariant (read-only)`

Description The response-header field which defines the exact location of the resource that was identified by the Request-URI. During redirection, this is the final URL of the resource returned.

ModifiedSince

Syntax `ModifiedSince: OleVariant`

Description Sets and gets the Modified-Since request-header field.

MultiThreaded

Syntax `MultiThreaded: WordBool`

Description Boolean. Sets and gets whether the HTTPProvider uses multithreading when executing HTTP requests.

NoCache

Syntax `NoCache: WordBool`

Description Boolean. Sets and gets the NoCache request-header field.

Password

Syntax `Password: OleVariant`

Description Sets and gets the web server access password.

Proxy

Syntax `Proxy: OleVariant`

Description Sets and gets the proxy server. Use the `GetApplicationSetting()` function with the following setting constants (50 and 51) to extract the users' proxy server settings:

```
var app = Application;
var httpPro = app.HTTPProvider;
httpPro.Proxy      = app.GetApplicationSetting(50);
httpPro.ProxyPort  = app.GetApplicationSetting(51);
```

ProxyPassword

Syntax `ProxyPassword: OleVariant`

Description Sets/gets the proxy server password.

ProxyPort

Syntax `ProxyPort: OleVariant`

Description Sets/gets the proxy server port.

ProxyUsername

Syntax `ProxyUsername: OleVariant`

Description Sets/gets the proxy server username.

RcvdCount

Syntax `RcvdCount: Integer (read-only)`

Description The size of the content stream received from the server. Use this property to display progress during asynchronous `GET` operations. Use the `ContentLength` property value extracted from the document header to get the total length of the incoming content stream.

ReasonPhrase

Syntax `ReasonPhrase: OleVariant (read-only)`

Description The Reason-Phrase element provides a short textual description of the Status-Code. The Status-Code is used by automata and the Reason-Phrase is for the human user.

The following are some of the Status-Code, Reason-Phrase pairs:

```
200 - OK
201 - Created
202 - Accepted
204 - No Content
301 - Moved Permanently
302 - Moved Temporarily
304 - Not Modified
400 - Bad Request
401 - Unauthorized
403 - Forbidden
404 - Not Found
500 - Internal Server Error
501 - Not Implemented
502 - Bad Gateway
503 - Service Unavailable
```

ReceivedHeaderAsString

Syntax `ReceivedHeaderAsString: OleVariant (read-only)`

Description The header elements as a string. The header elements are separated on different lines.

ReceivedStreamAsString

Syntax `ReceivedStreamAsString: OleVariant`

Description Sets and gets the `RECEIVED` stream as a string. Use `SaveReceivedStreamToFile` to save the received stream into a file.

Reference

Syntax `Reference: OleVariant`

Description Sets and gets the Referer request-header field. This field allows the client to specify, for the server's benefit, the address (URI) of the resource from which the Request-URI was obtained. This allows a server to generate lists of back-links to resources for interest, logging, optimized caching, and so on. It also allows obsolete or mis-typed links to be traced for maintenance. The Referer field must not be sent if the Request-URI was obtained from a source that does not have its own URI, such as input from the user's keyboard.

Sender

Syntax `Sender: OleVariant`

Description Sets and gets the sender parameter.

SendStreamAsString

Syntax `SendStreamAsString: OleVariant`

Description Sets and gets the SEND stream as a string.

SentCount

Syntax `SentCount: Integer (readonly)`

Description The size of the content stream sent to the server. Use this property to display progress during asynchronous `POST` operations.

State

Syntax `State: TAllaireHTTPProviderState (read-only)`

Description The state of the `HTTPProvider` object.

The enumerated state values are as follows:

```
0 - httpReady
1 - httpNotConnected
2 - httpConnected (browse)
3 - httpDnsLookup
4 - httpDnsLookupDone
5 - httpWaitingHeader
6 - httpWaitingBody
7 - httpAborting
```

StatusCode

Syntax `StatusCode: Integer (read-only)`

Description The HTTP request Status-Code element. This is a 3-digit integer result code of the attempt to understand and satisfy the request.

URL

Syntax `URL: OleVariant`

Description Sets and gets the URL location of the resource on which an HTTP method is to be applied.

Username

Syntax `Username: OleVariant`

Description Sets and gets the web server access username.

Methods

Abort

Syntax `Abort();`

Description Aborts the current HTTP operation.

Get

Syntax `Get();`

Description Performs an HTTP `GET` method request. Uses the appropriate properties of the `HTTPProvider` object to set HTTP request parameters, such as proxy server settings, username and password.

Example

```
//*****************************************//
// This script demonstrates a basic GET method
// against the Macromedia home page.
//*****************************************//
// Message box constants
var hsOKInfo = 64;
function Main () {

var app = Application;
var httpPro = app.HTTPProvider;
```

```
httpPro.URL        = "http://www.macromedia.com";

httpPro.Get();

app.MessageBox( httpPro.ReceivedHeaderAsString ,"Received HTTP Header",
        hsOKInfo);
app.MessageBox(
  "URL='     "+ httpPro.URL + "'\n" +
  "ProxyUsername='"   + httpPro.Username + "'\n" +
  "ProxyPassword='"   + httpPro.Password + "'\n" +
  "Proxy='"           + httpPro.Proxy + "'\n" +
  "ProxyPort='"       + httpPro.Proxyport + "'\n" +
  "ProxyUsername='"   + httpPro.ProxyUsername + "'\n" +
  "ProxyPassword='"   + httpPro.ProxyPassword + "'\n" +
  "Sender='"          + httpPro.Sender + "'\n" +
  "Agent='"           + httpPro.Agent + "'\n" +
  "Reference='"       + httpPro.Reference + "'\n" +
  "NoCache='"         + httpPro.NoCache + "'\n" +
  "ModifiedSince='"   + httpPro.ModifiedSince + "'\n" +
  "Cookie='"          + httpPro.Cookie + "'\n" +
  "ContentTypePost='" + httpPro.ContentTypePost + "'\n" +
  "MultiThreaded='"   + httpPro.MultiThreaded + "'\n"+
  "State='"           + httpPro.State + "'\n"+
  "ContentLength='"   + httpPro.ContentLength + "'\n"+
  "ContentType='"     + httpPro.ContentType + "'\n"+
  "RcvdCount='"       + httpPro.RcvdCount + "'\n"+
  "SentCount='"       + httpPro.SentCount + "'\n"+
  "StatusCode='"      + httpPro.StatusCode + "'\n"+
  "ReasonPhrase='"    + httpPro.ReasonPhrase + "'\n"+
  "AuthorizationRequest='"   + httpPro.AuthorizationRequest + "'\n"+
  "DocName='"         + httpPro.DocName + "'\n"+
  "Location='"        + httpPro.Location + "'\n"
  ,"HTTP Provider Diagnostics", hsOKInfo);

var sOutput = httpPro.ReceivedStreamAsString;
app.activeDocument.Text = sOutput;
}
```

GetAsync

Syntax `GetAsync();`

Description Performs an HTTP `GET` method request asynchronously.

Head

Syntax `Head();`

Description Performs an HTTP `HEAD` method request.

HeadAsync

Syntax `HeadAsync();`

Description Performs an HTTP `HEAD` method request asynchronously.

Post

Syntax `Post();`

Description Performs an HTTP `POST` method request.

Example **Sample POST script**

```
//*****************************************//
// This script illustrates a POST method by which
// three FORM variables are submitted to a ColdFusion page.
//*****************************************//

function Main () {
var app = Application;
var httpPro = app.HTTPProvider;
httpPro.URL        = "http://127.0.0.1/httptest.cfm";
var CustomerID  = "John Doe";
var ProductID   = "3456";
var DateSold    = "10/10/99";
var PostStream ='Customer_ID='     + httpPro.URLEncode( CustomerID ) +
           '&ProductNumber='   + httpPro.URLEncode( ProductID ) +
           '&SaleDate='        + httpPro.URLEncode( DateSold );
httpPro.SendStreamAsString = PostStream; httpPro.Post();
var sOutput = httpPro.ReceivedStreamAsString;
app.activeDocument.Text = sOutput;
}
```

PostAsync

Syntax `PostAsync();`

Description Performs an HTTP `POST` method request asynchronously.

SaveReceivedStreamToFile

Syntax `SaveReceivedStreamToFile(FilePath: OleVariant; bOverwrite: wordbool): OleVariant;`

Description Boolean. Saves the received stream into a file and returns the error message if an error occurred. The `bOverwrite` parameter specifies whether to overwrite any existing files or return an error.

The following error messages are predefined. Check for the error strings to detect these error cases:

- File already exits—Returned when file specified in FilePath exists and `bOverwrite` is set to `False`.
- Path does not exist—Returned when the path specified in `FilePath` does not exist.

Example **Sample download script**

```
//*****************************************//
// This script downloads a ZIP file using
// SaveReceivedStreamToFile
//*****************************************//

function Main () {
  var hsOKInfo = 64;
  var app = Application;
  var httpPro = app.HTTPProvider;
  httpPro.URL      = "http://127.0.0.1/test.zip";
  httpPro.Get();
  var bOverwrite = false;
  var sErrorMsg =
  httpPro.SaveReceivedStreamToFile("d:\\downloads\\test.zip",
    bOverwrite );
  if ( sErrorMsg != "" )
  {
    app.MessageBox( "A error occured :" + sErrorMsg ,"HTTPProvider
    Error", hsOKInfo);
  }
}
```

URLEncode

Syntax `URLEncode(const wsValue: WideString): WideString;`

Description Returns a URLEncoded form of the wsValue string. Use this function when populating URL or FORM data.

Example HTTPProvider script

```
//*********************************************//
// ActiveScripting example (JScript)

//*********************************************//
// This script contacts the site specified by the user URL,
// copies its source code and displays the page in the internal
// browser of the application

//*********************************************//
function Main (){

  var sDocName;
        var sSource;
  var sMessage;
```

```
with (Application){

    if (HTTPProvider.State == 0){
     // Set the URL property of HTTPProvider Object.
     HTTPProvider.URL = InputBox(VersionText, "Please Enter the URL.",
         "http://www.yahoo.com");
     // Perform HTTP Get Request
     HTTPProvider.Get();
     sSource = '';
     if (HTTPProvider.ReasonPhrase == "OK") // Check if the requested
         URL can be displayed
        sSource = HTTPProvider.ReceivedStreamAsString;  //Save the page
         source into the string
     else{
         sSource = '<font size="+2" color="#0000ff" >'  //If the page
         cannot be displayed, post error message and show a reason.
         sSource =  sSource + 'The request could not be completed:
         <br><br><li>'
         sSource =  sSource + HTTPProvider.ReasonPhrase + '.</font></
         li>';
         }
     sMessage = "The source of the requested page has been copied
         successfully,\n";
     sMessage = sMessage + 'the page now will be displayed ';
     sMessage = sMessage + 'in the browse window of \n' + VersionText +
         '!';
     MessageBox (sMessage, VersionText, 0);
    }
    else{
     sSource  = 'You are not connected to Internet properly, ';
     sSource =  sSource + 'please check the connection and try again';
     }
MessageBox("Some images may not be downloaded properly!","Warning!",0);
  NewDocument (false);     // Initialize a new document.
  ActiveDocument.InsertText(sSource, false); // Insert the received
        source code into new document.
  CurrentView = 2;// Change to the browse mode.
  }
}
```

ZIPProvider object

The `ZIPProvider` object is the general purpose PKZIP services object used by HomeSite+ for Dreamweaver MX for ZIP file composition and extraction. You can use this object in your VTOM scripts for low-level PKZIP operations from within HomeSite+ for Dreamweaver MX. Because Macromedia uses a licensed control for its ZIP operations, you cannot use `ZIPProvider` outside of HomeSite+ for Dreamweaver MX.

Properties

CompressionLevel

Syntax `CompressionLevel: Integer`

Description Sets and gets the compression level use to compress the archive file. You can set the value from 0 to 9; 0 represents no compression, 1 is fastest compression, and 9 is the slowest but most size-efficient compression.

ExtractionDir

Syntax `ExtractionDir: OleVariant`

Description Sets and gets the directory path where the files are extracted from an existing archive file using the Extract method.

FileCount

Syntax `FileCount: Integer (read-only)`

Description The number of elements in the archive.

Overwrite

Syntax `Overwrite: WordBool`

Description Boolean. Sets and gets whether the newly extracted files overwrite any existing file in the `ExtractionDir` directory.

Password

Syntax `Password: OleVariant`

Description Sets/gets the password for the archive file. Set this property when extracting password protected files or to password- protect files in an archive file that you just created.

ZIPComment

Syntax `ZIPComment: OleVariant`

Description Sets and gets the comment for the archive file. Use this property to read a comment of an existing archive file, or set the comment for a new archive file that you created.

ZIPFile

Syntax `ZIPFile: OleVariant`

Description Sets and gets the archive file path against which the operation is to be applied. To create a new archive file set this property to the file path of the new file and then call the Add method to populate the file with content. To add, delete, or extract files from an existing archive file, set the property to the file path of the file and call `Add`, `Delete`, or `Extract` methods on the file.

Example

```
//************************************************//
// The following example uses the ZIPFile property to extract a file
// (somefile.exe) from another file (d:\\downloads\\test.zip):
//************************************************//

function Main () {
  var hsOKInfo = 64;
  var app = Application;
  var ZIPPro = app.ZIPProvider;
  ZIPPro.ExtractionDir = 'd:\\ExtractHere\\'; ZIPPro.ZipFile =
    'd:\\downloads\\test.zip';
  var ResultCode = ZIPPro.Extract('somefile.exe');
  app.MessageBox( "Return Code :" + ResultCode ,"ZIPProvider",
    hsOKInfo);
}
```

Methods

Add

Syntax `Add(Files: OleVariant; bRecurse, bIncludeDirs, bIncludeHiddenFiles, bIncludeVolumeLabels: WordBool): Integer;`

Description Boolean. Adds files to an archive. Use the Files parameter to narrow down the group of files to add. You can specify a single file, a set of files delimited by line breaks, or an entire directory using wildcards. Use the bRecurse parameter to add subfolders. Use `bIncludeDirs` to specify whether directory information should be stored in the archive. Use `bIncludeHiddenFiles` and `bIncludeVolumeLabels` to restrict the inclusion of hidden files or volume labels. Do not set the `bIncludeVolumeLabels` parameter to `True` unless you specify the drive letter as the first element in the `Files` parameter. The `Files` parameter may contain multiple elements separated by line

breaks. The function returns a status code that can be investigated to determine the success of the operation.

Example **Sample Add script**

```
function Main () {
  var hsOKInfo = 64;
  var app = Application;
  var ZIPPro = app.ZIPProvider;
  //Identify the files to be compressed
  var ZIPFiles = 'd:\\projects\\hs4\\scripts\\test.htm';
  ZIPFiles = ZIPFiles + '\n' + 'd:\\projects\\hs4\\scripts\\test.zip';
  ZIPFiles = ZIPFiles + '\n' + 'd:\\projects\\hs4\\scripts\\*.js';

  //Add method options
  var bRecurse = true;
  var bIncludeDirInfo = true;
  var bIncludeHidden = true;
  var bIncludeVolume = false;
  //Create the ZIP file and execute Add method
  ZIPPro.ZipFile = 'd:\\newzips\\newarchive.zip';
  var nStatusCode = ZIPPro.Add( ZIPFiles , bRecurse, bIncludeDirInfo,
    bIncludeHidden, bIncludeVolume );

  //Interpret and display the return code
  var sMessage = '';
  if (nStatusCode == 0){sMessage = "Operation Succesful";}
    else if (nStatusCode == 10){sMessage = "General Warning";}
    else if (nStatusCode == 0){sMessage = "Success";}
    else if (nStatusCode == 10){sMessage = "WarningGeneral";}
    else if (nStatusCode == 30){sMessage = "WarningNoZipFile";}
    else if (nStatusCode == 40){sMessage = "WarningFilesSkipped";}
    else if (nStatusCode == 50){sMessage = "WarningEmptyZipFile";}
    else if (nStatusCode == 100){sMessage = "ErrorNoZipFile";}
    else if (nStatusCode == 110){sMessage = "ErrorZipStruct";}
    else if (nStatusCode == 120){sMessage = "ErrorMemory";}
    else if (nStatusCode == 130){sMessage = "ErrorBadCall";}
    else if (nStatusCode == 140){sMessage = "ErrorNothingToDo";}
    else if (nStatusCode == 150){sMessage = "ErrorDiskFull";}
    else if (nStatusCode == 160){sMessage = "ErrorEOF";}
    else if (nStatusCode == 180){sMessage = "ErrorLibInUse";}
    else if (nStatusCode == 190){sMessage = "ErrorUserAbort";}
    else if (nStatusCode == 200){sMessage = "ErrorTestFailed";}
    else if (nStatusCode == 210){sMessage = "ErrorZeroTested";}
    else if (nStatusCode == 240){sMessage = "ErrorDLLNotFound";}
    else if (nStatusCode == 250){sMessage = "ErrorInternalLogic";}
    else if (nStatusCode == 280){sMessage = "ErrorTempFile";}
    else if (nStatusCode == 290){sMessage = "ErrorRead";}
    else if (nStatusCode == 300){sMessage = "ErrorWrite";}
    else if (nStatusCode == 310){sMessage = "ErrorCantCreateFile";}
    else if (nStatusCode == 350){sMessage = "ErrorParentDir";}
    else if (nStatusCode == 370){sMessage = "ErrorNameRepeat";}
    else if (nStatusCode == 380){sMessage = "ErrorLatest";}
    else if (nStatusCode == 400){sMessage = "ErrorDOSError";}
```

```
    else if (nStatusCode == 410){sMessage = "ErrorMultidisk";}
    else if (nStatusCode == 420){sMessage = "ErrorWrongDisk";}
    else if (nStatusCode == 430){sMessage = "ErrorMultidiskBadCall";}
    else if (nStatusCode == 440){sMessage = "ErrorCantOpenBinary";}
    else if (nStatusCode == 450){sMessage =
      "ErrorCantOpenSfxConfig";}
    else if (nStatusCode == 460){sMessage =
      "ErrorInvalidEventParam";}
    else if (nStatusCode == 470){sMessage = "ErrorCantWriteSfx";}
    else if (nStatusCode == 490){sMessage = "ErrorBinaryVersion";}
    else if (nStatusCode == 500){sMessage = "ErrorNotLicensed";}
    else if (nStatusCode == 510){sMessage = "ErrorCantCreateDir";}

  app.MessageBox( sMessage ,"ZIPProvider", hsOKInfo);
}
```

Delete

Syntax `Delete(Files: OleVariant): Integer;`

Description Deletes files from an archive. Use the `Files` parameter to narrow down the group of files to delete. You can specify a single file, a set of files delimited by line breaks or an entire directory using wildcards. If left an empty string, all files are extracted. The function returns a status code that you can investigate to determine whether the operation succeeded.

Extract

Syntax `Extract(Files: OleVariant): Integer;`

Description Extracts files from an archive. Use the `Files` parameter to narrow down the group of files to extract. You can specify a single file, a set of files delimited by line breaks or an entire directory using wildcards. If left an empty string, all files are extracted. The function returns a status code that you can investigate to determine whether the operation succeeded.

FileDate

Syntax `FileDate(nIndex: Integer): WideString;`

Description Returns the datetime of an existing archive file element (file/directory/volume) by index. The `index` value can be from `0` to `FileCount-1`.

FileIsDirectory

Syntax `FileIsDirectory(nIndex: Integer): WordBool;`

Description Boolean. Use this function to determine whether a specific element in an archive is a directory. The `nIndex` value can be from `0` to `FileCount-1`.

FileIsHidden

Syntax `FileIsHidden(nIndex: Integer): WordBool;`

Description Boolean. Use this function to determine whether a specific element in an archive is a hidden file. The `index` value can be from `0` to `FileCount-1`.

FileIsReadOnly

Syntax `FileIsReadOnly(nIndex: Integer): WordBool;`

Description Boolean. Use this function to determine whether a specific element in an archive is read-only. The `index` value can be from `0` to `FileCount-1`.

FileIsSystem

Syntax `FileIsSystem(nIndex: Integer): WordBool;`

Description Boolean. Use this function to determine whether a specific element in an archive is a system file. The `index` value can be from `0` to `FileCount-1`.

FileIsVolume

Syntax `FileIsVolume(nIndex: Integer): WordBool;`

Description Boolean. Use this function to determine whether a specific element in an archive is a volume label. The `index` value can be from `0` to `FileCount-1`.

FileName

Syntax `FileName(nIndex: Integer): WideString;`

Description Returns the name of an existing archive file element (file/directory/volume) by index. The `index` value can be from `0` to `FileCount-1`.

Example

```
//*********************************************//
// This code illustrates a loop over the content
// of the archive, looking for a specific filename.
//*********************************************//

function Main () {
var hsOKInfo = 64;
var app = Application;
var ZIPPro = app.ZIPProvider;
ZIPPro.ZipFile = 'd:\\zipfiles\\test.zip';
```

```
// Loop through the contents of the ZIP file
for ( x = 0; x &lt; ZIPPro.FileCount; x++ ) {
  if ( ZIPPro.FileName(x) == 'cfabort.vtm' ) {
    app.MessageBox( &quot;File found in the
      archive.&quot;,&quot;ZIPProvider&quot;, hsOKInfo);
    }
  }
}
```

FileSize

Syntax `FileSize(nIndex: Integer): Double;`

Description Returns the size of an existing archive file element (file/directory/volume) by index. The `nIndex` value can be from `0` to `FileCount-1`.

ActiveScripting examples

JScript

```
<html>
<head>
<title>JScript Example</title>
</head>
<body>
<pre>
//*******************************************************//
// ActiveScripting example (JScript)
//*******************************************************//
// Generates a table of the filenames of all open documents,
// listing their modified status and read-only property.
//*******************************************************//
function Main (){
  var newline;
  var br;
  var tab;
  var sTable;
  var count;

  newline = '\n';
  br = '<BR>';
  tab = '\t';
with (Application){
  // start the table
  sTable = '<B><FONT size="+1" color="Blue">';
  sTable = sTable + 'Names and properties of all open documents:';
  sTable = sTable + '</FONT></B><br><br><br>';
  sTable = sTable + '<TABLE border="1" width="500">' + newline;
  sTable = sTable + newline + '<TR><TD><B>Document Name</B></TD>';
  sTable = sTable + '<TD><B><CENTER>Modified</CENTER></B></TD>';
  sTable = sTable + '<TD><B><CENTER>Read-Only</CENTER></B></TD>';
  sTable = sTable + '</TR>' + newline;

    count = 0;

  // loop through all open documents and put their
  // names and properties into the table.

  while (count <= (DocumentCount - 1)){

      // get document's name

      fname = GetDisplayName(DocumentCache(count).FileName);

      //extracting the document's name only without its path

      sTable = sTable + tab + '<tr><td>' + ExtractFileName(fname);
      sTable = sTable + '</td><td>';
```

```
            //Is this document Modified?

        if (DocumentCache (count).Modified)
         sTable = sTable + '<center> Yes'+'</td><td></center>';
        else
           sTable = sTable + '<center> No' + '</td><td></center>';

        //Is this document Read-Only?

        if (DocumentCache (count).ReadOnly)
         sTable = sTable + '<center>Yes'+ '</td><td></center> ';
        else
           sTable = sTable + '<center>No' + '</td><td></center> ';

        // close row.

        sTable = sTable + '</td></tr>' + newline;

        count ++;  // increment count by 1.
    }

    sTable = sTable + newline + '</table><br><br>';

    sTable = sTable + '<b><i><font color="#0000ff"> You ran this script
          inside </font></i></b>';
    sTable = sTable + '<b><font color="#ff0000">'+ VersionText + '.</
          font></b>';

    NewDocument (false);

    ActiveDocument.InsertText (sTable, false);

    CurrentView = 2;

      }
}

//*****************************************************//
// If the filename is an empty string, either set it
// to untitled or don't change it.
//*****************************************************//

function GetDisplayName (fname){
  if (fname == '')
    return '(untitled)';
  else
      return fname;
}
</pre>
</body>
</html>
```

VBScript

```
<html>
<head>
  <title>VBScript Example</title>
</head>

<body>

<pre>

//*********************************************************//
// Displays a table containing file information
//*********************************************************//

Sub Main
    dim app
    dim idx
    dim sTable
    dim newline, fname, br, tab
    dim nCurrentIdx
    newline = chr(13) + chr(10)
    br = "<br>"
    tab = chr(9)

    ‘ create app reference.  note that the Application object is only
    ‘ available from within HomeSite and Studio - to create the app
       object from an
    ‘ external script, use
       CreateObject("AllaireClientApp.TAllaireClientApp")
    set app = Application

    ‘ save the index of the current document so it can be returned to
    nCurrentIdx = app.DocumentIndex

    ‘ start the table
    sTable = "<b><font color=Blue>Names of all open"_
           + " documents:</font></b>"_
           + br + newline + br + newline _
           + "<table border=1 width=460>"_
           + newline + "<tr>"_
           + "<td><b>Document</b></td>"_
           + "<td><b>Modified</b></td>"_
           + "<td><b>Read-Only</b></td>"_
           + "</tr>" + newline

    ‘ loop through all open documents (note that DocumentCount is
       1-based,
    ‘ whereas DocumentCache() is 0-based)
    for idx = 0 to app.DocumentCount - 1
         ‘ get document name (uses function example)
        fname = GetDisplayName(app.DocumentCache(idx).Filename)
         sTable = sTable + tab + "<tr><td>" + fname
         sTable = sTable + "</td><td>"
```

```
            ' is this document modified?
            if app.DocumentCache(idx).Modified then sTable = sTable + "Yes"
            sTable = sTable + "</td><td>"

            ' is this document read-only?
            if app.DocumentCache(idx).ReadOnly then sTable = sTable + "Yes"

            ' close row
            sTable = sTable + "</td></tr>" + newline
        next
        sTable = sTable + newline + "</table>"

        ' add a new document (False = don't create from default template -
            blank)
        app.NewDocument(False)

        ' insert the table (note that the ActiveDocument will be the new
            document
        ' created above)
        app.ActiveDocument.InsertText sTable, False

        ' switch to browse mode
        app.CurrentView = 2

        ' wait for user to re-enter edit mode
        while app.CurrentView <> 1
            ' Wait is a home-grown routine to make up for the loss of
            ' DoEvents in VBScript. It will pause for a given number of
            ' milliseconds without locking the user interface
            app.Wait(100)
        wend

        ' return to the original document
        app.DocumentIndex = nCurrentIdx

        MsgBox "Script Completed."

        ' free the references
        set app = nothing
    End Sub

    ' function example
    function GetDisplayName(fname)
        if fname = "" then
           GetDisplayName = "(untitled)"
        else
            GetDisplayName = fname
        end if
    end function
    </pre>

    </body>
    </html>
```

Third-party add-ins

Scripting support offers many possibilities for third-party developers. For the latest news on third-party add-ins, visit the VTOM Scripts section of the Macromedia Developer Exchange at www.macromedia.com/desdev/developer/.

Running a script at startup

If you want to distribute an add-in, you can register it to run a script the first time the program loads.

At start-up, the program executes any scripts it finds listed in this Windows Registry key:

HKEY_CURRENT_USER\Software\Macromedia\HomeSite+\RunOnce

You can create this key and enter the script name. After this key is read, the program deletes the entries so that they do not execute again.

Sample startup script

```
//************************************************//
// This script adds a toolbar for an application.
// Add a string entry whose value contains the
// absolute path to the script you want to run
//************************************************//
"MyAppScript"="c:\\MyApp\\MyAppScript.bas"

Sub Main
  const TB_NAME = "MyApp"
  Dim app

  set app = Application

  ' delete toolbar if it already exists
  if app.ToolbarExists(TB_NAME) then
    app.DeleteToolbar(TB_NAME)
  end if

  ' recreate toolbar
  app.CreateToolbar TB_NAME

  ' dock it to the bottom
  app.SetToolbarDockPos TB_NAME, 2

  ' add app toolbutton
  app.AddAppToolbutton TB_NAME, "c:\MyApp\MyApp.exe", "", "Click to run
    MyApp"
  ' add tag toolbutton
  app.AddTagToolbutton TB_NAME, "<div>", "</div>", "DIV Toolbutton",
    "DV", ""
    ' add script toolbutton
  app.AddScriptToolbutton TB_NAME, app.AppPath + "test.bas" , "Script
```

```
    Toolbutton", "SC", ""
    ' add VTM toolbutton
  app.AddVTMToolbutton TB_NAME, app.AppPath +
    "Extensions\TagDefs\HTML\div.vtm" , "VTM Toolbutton", "VT", ""

End Sub
```

Table of CommandID values

The following table lists the set of currently exposed commands. For backward compatibility, these values will not change, but some value might become obsolete and be removed from the set.

Command ID	Value
CMDID_cmdNone	0
CMDID_cmdFileOpen	3
CMDID_cmdFileClose	4
CMDID_cmdFileCloseAll	5
CMDID_cmdFileNew	6
CMDID_cmdFileNewWizard	7
CMDID_cmdFileSave	8
CMDID_cmdFileSaveAs	9
CMDID_cmdFileSaveAsTemplate	10
CMDID_cmdFileSaveAll	11
CMDID_cmdFileReload	12
CMDID_cmdFileInsert	13
CMDID_cmdFileConvertTextFile	14
CMDID_cmdFilePrint	15
CMDID_cmdFileOpenFromWeb	16
CMDID_cmdProjectNew	17
CMDID_cmdProjectOpen	18
CMDID_cmdProjectClose	19
CMDID_cmdProjectAddFile	20
CMDID_cmdProjectRemoveFile	21
CMDID_cmdProjectSync	22
CMDID_cmdProjectUpload	23
CMDID_cmdCursorLeft	24
CMDID_cmdCursorRight	25
CMDID_cmdCursorWordLeft	26
CMDID_cmdCursorWordRight	27
CMDID_cmdCursorDown	28
CMDID_cmdCursorUp	29
CMDID_cmdCursorPageDown	30
CMDID_cmdCursorPageUp	31

Command ID	Value
CMDID_cmdCursorDocStart	32
CMDID_cmdCursorDocEnd	33
CMDID_cmdCursorLineStart	34
CMDID_cmdCursorLineEnd	35
CMDID_cmdEditBackspace	36
CMDID_cmdEditCut	37
CMDID_cmdEditCopy	38
CMDID_cmdEditPaste	39
CMDID_cmdEditUndo	40
CMDID_cmdEditRedo	41
CMDID_cmdEditRepeatLastTag	42
CMDID_cmdEditCodeFormat // CodeSweeper	43
CMDID_cmdEditToggleInsertMode	44
CMDID_cmdEditDelete	45
CMDID_cmdEditDeleteLine	46
CMDID_cmdEditDeleteToEOL	47
CMDID_cmdEditDeleteWordLeft	48
CMDID_cmdEditDeleteWordRight	49
CMDID_cmdEditSelectAll	50
CMDID_cmdEditConvertTagCase	51
CMDID_cmdEditGotoLine	52
CMDID_cmdEditGotoPreviousStartTag	53
CMDID_cmdEditGotoNextStartTag	54
CMDID_cmdEditGotoPreviousEndTag	55
CMDID_cmdEditGotoNextEndTag	56
CMDID_cmdEditCurrentTag	57
CMDID_cmdEditWordWrap	58
CMDID_cmdEditSetBookmark	59
CMDID_cmdEditGotoNextBookmark	60
CMDID_cmdEditTriggerCodeTemplates	61
CMDID_cmdEditIndent	62
CMDID_cmdEditUnindent	63
CMDID_cmdSelectionConvertToUL	64
CMDID_cmdSelectionConvertToOL	65
CMDID_cmdSelectionConvertToTable	66

Command ID	Value
CMDID_cmdSelectionAddBR	67
CMDID_cmdSelectionStripTags	68
CMDID_cmdSelectionUCase	69
CMDID_cmdSelectionLCase	70
CMDID_cmdSearchFind	71
CMDID_cmdSearchReplace	72
CMDID_cmdSearchAgain	73
CMDID_cmdSearchFindEx	74
CMDID_cmdSearchReplaceEx	75
CMDID_cmdSearchFindMatchingTag	76
CMDID_cmdSearchReplaceDoubleSpacing	77
CMDID_cmdSearchReplaceExtChars	78
CMDID_cmdToolsTagChooser	79
CMDID_cmdToolsSpellCheck	80
CMDID_cmdToolsSpellCheckAll	81
CMDID_cmdToolsSpellMark	82
CMDID_cmdToolsDocumentWeight	83
CMDID_cmdToolsThumbnails	84
CMDID_cmdToolsVerifyLinks	85
CMDID_cmdToolsValidateDoc	86
CMDID_cmdToolsValidateTag	87
CMDID_cmdToolsPalette	88
CMDID_cmdToolsExecAsScript	89
CMDID_cmdToolsStyleEditor	90
CMDID_cmdToolsScriptWizard	91
CMDID_cmdToolsTableSizer	92
CMDID_cmdViewFullScreen	93
CMDID_cmdViewToggleResTab	94
CMDID_cmdViewToggleResTabFocus	95
CMDID_cmdViewPreviousDocument	96
CMDID_cmdViewNextDocument	97
CMDID_cmdViewInternalBrowser	98
CMDID_cmdViewExternalBrowser	99
CMDID_cmdViewDefaultExternalBrowser	100
CMDID_cmdViewDreamweaver	101

Command ID	Value
CMDID_cmdViewEditSource	102
CMDID_cmdViewToggleEditPreview	103
CMDID_cmdViewToggleEditDesign	104
CMDID_cmdViewToggleTagInspectorFocus	105
CMDID_cmdViewToggleSpecialChar	106
CMDID_cmdViewTagInsight	107
CMDID_cmdViewTagTip	108
CMDID_cmdViewEditorToolbar	109
CMDID_cmdViewEditorTab	110
CMDID_cmdViewToggleQuickBar	111
CMDID_cmdViewResults	112
CMDID_cmdOptionsSettings	113
CMDID_cmdOptionsCustomize	114
CMDID_cmdOptionsOrganizeQuickBar	115
CMDID_cmdOptionsCodeFormatSettings	116
CMDID_cmdLinkBotVerifyPage	117
CMDID_cmdLinkBotVerifyProject	118

Table of SettingID values

The following table lists currently exposed setting IDs. For backward compatibility, these values will not change, but some might become obsolete and be removed from the set.

Setting ID	Value	Description
SET_FLAG_READONLY_FILES	0	Flag read-only files in the file list
SET_WARN_READONLY_OPEN	1	Warn when opening read-only files
SET_RELOAD_TYPE	2	What to do when another application modifies the current document
SET_STARTUP_FOLDER_TYPE	3	Start in last opened folder or specific folder
SET_DEFAULT_EXTENSION	4	Default extension for new documents
SET_DEFAULT_TEMPLATE	5	Default template to use when creating new documents
SET_SHOW_RESTAB	6	Display the resource tab
SET_CURRENT_RESTAB	7	Page in resource tab active when closed (reset at startup)
SET_RESTAB_ALIGN	8	Set Resources tab alignment (deprecated in version 4.5)
SET_RESTAB_TABPOS	9	Tab position (top/bottom)
SET_RESTAB_WIDTH	10	Set Resource tab width
SET_PALETTE_FILE	11	Set current palette file
SET_DYNAMIC_REFRESH	12	Dynamically refresh the file list when current directory changed
SET_FILELIST_TIMEOUT	14	Milliseconds to wait before cancelling retrieval of files in folder (internal use only)
SET_LOAD_CUSTOM_SHORTCUTS	15	If set, loads custom command table at startup
SET_LINEAR_TAG_OUTPUT	16	Output tags from tag editors on a single line
SET_TITLE_AS_LINK_DESC	17	Use the `title` tag as the description for dropped links
SET_INTERNAL_BROWSE_MSIE	18	Use MSIE as the internal browser

Setting ID	Value	Description
SET_MSIE_AUTO_REFRESH	19	Automatically refresh MSIE after navigating (needed for IE3.x)
SET_ANCHORDLG_NAMES	20	Show named links in anchor dialog box
SET_USE_SHORT_TEMPFILE_NAME	21	Use short filename for temp files when shelling to external browser
SET_RESTORE_FILES	22	Restore files at startup
SET_FILE_FILTER	23	File filter for file list
SET_REMOTE_FILE_FILTER	24	File filter for remote file list
SET_TAG_INSIGHT	26	Use tag insight
SET_INSIGHT_SHOWTAGLIST	27	Use tag insight for tags (immediately after < is pressed)
SET_TAG_COMPLETION	28	Toggle tag completion
SET_TAGHELP_DELAY	29	Milliseconds to delay tag insight
SET_LOWERCASE_TAGS	30	Lowercase inserted tags
SET_HEX_COLORS	31	Always use hex values instead of color names
SET_DHTMLED_ALLOW_FRAMES	32	If True, allow DHTMLEdit view to display frames (version 1.0 of DHTMLEdit did not support frames)
SET_STANDARD_OPEN_DLG	33	Use TOpenDialog instead of TfFileOpenDlg (common dialog box instead of custom)
SET_OPEN_DLG_PREVIEW	34	Enable preview in TfFileOpenDlg
SET_OPEN_DLG_DETAILS	35	Show details in TfFileOpenDlg (ViewStyle vsReport instead of vsList)
SET_USER_DICTIONARY_FILE	37	User dictionary (for integrated spell checker)
SET_SPELLCHECK_SKIPTAGS	38	Skip tags and <script> blocks when spell checking
SET_SHOW_QUICKBAR	39	Show QuickBar
SET_TAG_VALIDATION	40	Validate when > is pressed

Setting ID	Value	Description
SET_USE_CSE_VALIDATOR	41	Use CSE validator for full-page validation.Never change the key for this setting, because Alsoft may update it during the CSE install
SET_SHOW_EDITORTOOLBAR	42	Show Editor toolbar
SET_SHOW_EDITORTAB	43	Show the tabset below the Editor
SET_WARNING_BEEP	44	Beep when a warning message displays
SET_AUTOCONVERT_SPECHAR	45	Automatically convert special characters (> #128) when typed
SET_RUNONCE_ENABLED	46	At startup, check the gREG_ROOT\RunOnce key and executes any scripts
SET_TAGTREE_HEIGHT	47	Set tag tree height in pixels
SET_SHELLTREE_HEIGHT	48	Set height in pixels for files tab pane
SET_PROXY_NAME	50	Set the proxy port name for link verification and open from web
SET_PROXY_PORT	51	Set the proxy port for link verification and open from web
SET_CONNECT_TIMEOUT_MS	52	Timeout (in milliseconds) for link verification and open from web
SET_SAVE_FILE_FORMAT	53	File format for saving (pc, unix, mac)
SET_WEB_DOCUMENT_EXTENSIONS	54	Configure the extensions list for recognized web document formats
SET_WEB_IMAGE_EXTENSIONS	55	Configure the extensions list for recognized web image formats
SET_BROWSE_MAPPINGS	56	Enable internal browser mappings
SET_SHOW_HELP_SEPARATE_PANE	57	Use main integrated browser for Help
SET_HELP_BROWSER_HEIGHT	58	Set the height in pixels for the Help browser
SET_QUICKBAR_LEFT	60	Set the left position of the QuickBar

Setting ID	Value	Description
SET_SHOW_SKIPPED_FILES _IN_SEARCH	66	See dxSearchResults.AddSkipItem
SET_LOWERCASE_LINKS	67	Lowercase IMG, A, and other dropped links
SET_ALLOW_MULTIPLE_INSTANCES	79	Only for debugging. Multiple instances use same reg settings and would conflict
SET_CLOSE_PARA_TAGS	80	Add when toolbutton is clicked
SET_PROMPT_ON_UNKNOWN_FILES	81	Show warning before opening unknown files
SET_PROMPT_BROWSE_UNSAVED _REMOTE	82	When browsing remote files, prompt user to save changes
SET_FILEDLG_SHOW_CURRENT	83	When displaying the open or save dialog box, default to the current local dir
SET_STARTUP_BLANK_DOC	84	Start with a blank document (overrides default template)
SET_SHOW_SHORTCUT_IN_HINTS	85	Show keyboard shortcuts in toolbutton hints
SET_INTERNAL_BROWSE_MOZILLA	86	Use MOZILLA as the internal browser
SET_OEM_CONVERSION	87	Control call to CharToOemBuff()
SET_FULL_PATH_ON_SAVE	88	Puts absolute path in filename edit box when saving
SET_CHECK_UPDATE_TYPE	100	Check for product update every time app loads
SET_CHECK_UPDATE_DATE	101	Date last check for update run
SET_CHECK_UPDATE_TIMEOUT _SHORT	102	Timeout for update check at startup (milliseconds)
SET_CHECK_UPDATE_TIMEOUT_LONG	103	Timeout for update check initiated by user
SET_CHECK_UPDATE_LAST_DATE	104	Check date of last product update
SET_CENTER_START_TAG	105	Start tag to use when Align Center toolbutton is clicked
SET_CENTER_END_TAG	106	End tag to use when Align Center toolbutton is clicked

Setting ID	Value	Description
SET_USE_VTM_IMAGE_DIALOG	107	Use vtml image dialog
SET_USE_VTM_ANCHOR_DIALOG	108	Use vtml anchor dialog
SET_USE_VTM_BODY_DIALOG	109	Use vtml body dialog
SET_TEMPFILE_PREFIX	110	Prefix to use for temp files
SET_FILELIST_COLS_ACTIVE	120	Set local columns as active
SET_FILELIST_COL_SIZES	121	Local FileList Column Size Settings
SET_REMOTE_FILELIST_COLS_ACTIVE	122	Set remote columns as active
SET_REMOTE_FILELIST_COL_SIZES	123	Remote FileList Column Size Settings
SET_REMOTE_FPTREE_HEIGHT	125	Set height for remote file list (deprecated in version 4.5)
SET_PROJECT_VERIFIER_EXCL_EXTS	130	Setting for exclusion extension list
SET_SPECIALCHAR_LAYOUT	131	Layout orientation of special character toolbar
SET_DREAMWEAVER_INTEGRATION	140	Set Dreamweaver MX integration and file save prompt
SET_DREAMWEAVER_LAUNCH_TYPE	149	Set Dreamweaver MX launch
SET_DIR_LOCAL_USER	150	Set location of local user directory
SET_DIR_SNIPPETS_PRIVATE	151	Set location of private snippets directory
SET_DIR_SNIPPETS_SHARED	152	Set location of shared snippets directory. Blank for no shared snippets
SET_DIR_HELP	153	Set location of Help files directory
SET_DIR_PALETTES	154	Set location of palette file directories
SET_DIR_TOOLBARS	155	Set location of toolbars directory
SET_DIR_TOOLBUTTON_IMAGES	156	Set location of images directory for toolbars when no path specified in TBR file
SET_DIR_PARSERS	157	Set location of color coding parsers directory
SET_DIR_STARTUP	158	Set location of startup directory

Setting ID	Value	Description
SET_DIR_WIZARDS	159	Set location of wizard root directory
SET_DIR_CUSTOM_TEMPLATES	160	Set location of custom templates directory
SET_DIR_OLD_TEMPLATES	161	Set location of old templates directory
SET_DIR_OLD_TAGEDITORS	162	Set location of old tag editors directory
SET_DIR_EXTENSIONS	163	Set location of extensions directory
SET_DIR_TAGDEFS	164	Set location of tag definitions directory
SET_DIR_CUSTOM_TAGDEFS	165	Set location of custom tag definitions directory
SET_DIR_OUTLINEPROFILES	166	Set location of outline profiles directory
SET_DIR_SPELLING	167	Set location of spelling checker directory
SET_DIR_AUTOFORMATTERS	168	Set Codesweeper profile directory
SET_DIR_SAVED_SEARCHES	169	Set location of saved searches directory
SET_DIR_HTML_HELP	170	Set location of html Help files directory
SET_DIR_THUMBNAILS	199	Set location of stored thumbnails (deleted at close)
SET_TAGCHOOSER_FILE	200	Set location of tag chooser file
SET_EXPBUILDER_FILE	201	Set location of Expression Builder file
SET_USER_SHORTCUTS_FILE	202	Set location of key shortcuts file
SET_SNIPPETS_SHORTCUTS_FILE	203	Set location of snippets shortcut file
SET_SPECIAL_CHAR_FILE	204	Set location of special character file
SET_TOOLBAR_POS_FILE	205	Set location of file containing data about the positions of toolbars and tool windows
SET_TOOLBAR_DATA_FILE	206	Set location of file containing data about the active toolbars

Setting ID	Value	Description
SET_QUICKBAR_DATA_FILE	207	Set location of file containing data about what toolbars are in the QuickBar
SET_INSIGHT_TAGLIST_FILE	208	Set location of tag insight file
SET_STYLE_EDIT_EXE	209	Set location of styleedit executable
SET_CODETEMPLATE_FILE	210	Set location of code template file
SET_AUTOREPLACE_FILE	211	Set location of code template replace file
SET_TAGCOMPLETION_FILE	212	Set location of tag completion file
SET_BREAKPOINTS_FILE	213	Set location of debugger breakpoint file
SET_DEFAULTAUTOFORMAT_FILE	214	Default formatting used by the Codesweeper
SET_VERITY_HELP_SEARCH_HELPFILE	215	Used to display search tips
SET_STYLE_EDIT_WND_CLASS	216	Used to dynamically invoke a style editor (StyleEd or TopStyle)
SET_STYLE_EDIT_OLE_OBJECT	217	Used to dynamically invoke a style editor (StyleEd or TopStyle)
SET_EDITOR_FONTNAME	300	Set editor font name
SET_EDITOR_FONTSIZE	301	Set editor font size
SET_EDITOR_TABWIDTH	302	Set editor tab width
SET_EDITOR_SHOW_RMARGIN	303	Set editor right margin
SET_EDITOR_RMARGIN	304	Set editor left margin
SET_EDITOR_AUTOINDENT	305	Set editor indent
SET_EDITOR_FOREGROUND	306	Set editor foreground color
SET_EDITOR_BACKGROUND	307	Set editor background color
SET_EDITOR_SHOW_GUTTER	308	Set editor gutter display
SET_EDITOR_WORDWRAP	309	Set editor word wrap
SET_EDITOR_USE_TABS	310	Set editor
SET_EDITOR_AUTOREPLACE	311	Set editor
SET_EDITOR_AUTOREPLACE _TRIGGERS	312	Set editor
SET_EDITOR_AUTOQUOTE	313	Automatically close quotes

Setting ID	Value	Description
SET_EDITOR_AUTOCLOSE_ASP	314	Add %> when <% entered
SET_EDITOR_AUTOCOMMENT	315	Add --> when <!-- is entered
SET_EDITOR_HTML_DROP	316	Allow HTML to be dragged-and-dropped from compliant apps (inserted as text otherwise)
SET_EDITOR_SHOW_BROWSER	317	Show web browser below editor
SET_EDITOR_BROWSER_HEIGHT	318	Set height for browser below editor
SET_EDITOR_BROWSE_UPDATE _INTERVAL	319	Set update interval for browser below editor
SET_EDITOR_BROWSE_AUTOUPDATE	320	Set auto-update for browser below editor
SET_EDITOR_ALLOW_DRAGDROP	321	Allow drag/drop inside editor
SET_EDITOR_ALLOW_UNDO_AFTER _SAVE	322	Allow saves after undo
SET_EDITOR_ADJUST_POS_FOR _WORDWRAP	323	See fAllaireClientMain.ShowEditor Pos
SET_EDITOR_LINE_NUMBERS	324	Set line numbers in editor gutter
SET_EDITOR_AUTOPOUND	325	Set auto insertion of pound sign
SET_EDITOR_CHARSET	326	Set editor character set
SET_EDITOR_OUTLINECURRLINE	327	Set outline on current line
SET_SITEVIEW_GRADIENT	400	Set gradient fill for site view
SET_SITEVIEW_SHOWTITLE	401	Toggle site view title
SET_SITEVIEW_STYLE	402	Set tree or chart view
SET_SITEVIEW_AUTONAV_TO_LINK	403	Navigate to the current link when user clicks on a node in site view
SET_PROJECT_RESTORE_AT_START	500	Restore last opened project at startup
SET_PROJECT_LAST_OPENED	501	Set project to absolute path to last opened project
SET_PROJECT_CLOSE_ALL_ON_OPEN	502	Close all files when another project is opened
SET_AUTOFORMAT_CURRENT _PROFILE	601	Set the path to the currently selected Codesweeper

Setting ID	Value	Description
SET_AUTOFORMAT_SHOW_PROMPT	602	Show prompt when user clicks toolbutton
SET_CFS_HELP_SERVER CFStudio only	61	Server to map to in dxWebBrowser.DoMSIENewWindow when user browses snippets
SET_CFS_ADJUST_HELP_EXAMPLES CFStudio only	62	If `False`, then Help examples are not adjusted. See dxWebBrowser.DoMSIENewWindow.

Glossary

This section contains brief definitions of terms that you might encounter while learning to use the product.

ASCII

American Standard Code for Information Interchange, developed by the American National Standards Institute (ANSI). System of representing an alphabetic, numeric, or special character with a 7-bit binary number (string of seven 0s or 1s), which defines 128 possible characters. UNIX and DOS-based operating systems use ASCII for text files; while Windows NT and 2000 uses Unicode.

attribute

Characteristic of a tag; for example, the `src` attribute of the `<image>` tag has a value that is equal to the location of the image to display.

breakpoint

Point in your code where the debugger stops processing and waits for your input.

browser

Program that displays markup language documents and other documents, based on a document's structure. Different Web browsers have different rules for the code they accept and how they display the content of different tags.

class

In CSS, a class is a style definition that is a subset of another style definition. For example, you could define a class of the paragraph tag for contact information, to make contact information display centered in large, red, bold text.

In object-oriented programming, a class is a user-defined type (as opposed to *int*, which is a built-in type). Classes have defined characteristics and behaviors, known as **attributes** and **methods**. A class serves as a blueprint for an object.

A POSIX character class is a code that represents a set of characters; for example `alpha` represents all alphabetic characters ([A-Za-z]).

client

A computer that is requesting something from a server; for example, a computer that is asking for the results of a database query.

code template

User-defined text that is inserted by typing a user-defined keyword. HomeSite+ for Dreamweaver MX has a few pre-set code templates; for example, typing "scriptj" inserts a JavaScript block.

CSS

Cascading Style Sheets. Defines formatting "styles" for page elements such as paragraphs and tables. Styles are usually defined in a separate file (or **style sheet**), with Web pages referencing this file. Alternatively, you can embed styles in a Web page by creating a `<style>...</style>` block. Embedded styles affect the display of the tags that are between the start and end `<style>` tags.

data source

Entry point for database operations.

deployment

Uploading files to a location that others can access. For example, your relatives across the country cannot see the online photo album that you created on your computer until you deploy the files to a Web server. You can also deploy to an **intranet** or **extranet**, so that only a select group of people have access. Many companies have their own intranet Web site, and most business-to-business e-commerce Web sites are extranets, accessible only by paying member companies.

domain name

Name that locates an organization or other entity on the Internet; for example, www.macromedia.com. The domain name consists of three parts: the host server name (www); the name describing the organization or entity (macromedia); and the type of domain (.com). Domains can be commercial (.com), non-profit (.org), Internet Service Providers (.net), four-year colleges or universities (.edu), U.S. government (.gov), U.S. military (.mil), or non-U.S. countries (fr for France, .de for Germany, and so on).

DTD

Document Type Definition. Specification that defines the rules of a Standard Generalized Markup Language (SGML) language; for example, its tags and required attributes and allowed values for its tags. A DTD complies to the rules of SGML, and enables an SGML compiler to correctly handle documents of the type it defines.

encoding

Format that defines how the values of certain types are represented, in terms of bits. Examples: American Standard Code for Information Interchange (ASCII), a 7-bit character set; and Unicode, a 16-bit character set.

entity name

Also known as an **entity reference**. Code for a special or extended character. Entity names prevent browsers from misinterpreting these characters. For example, the lesser than and greater than signs (<, >) are used to form tags in the code. Using the entity names `<` and `>` instead of < and > prevents browsers from interpreting the characters as tag brackets.

FTP

The File Transfer Protocol (FTP) defines an Internet standard for transferring files between two computers over a given network.

GIF

Graphic Interchange Format. Type of image file (.gif) that has a small file size but less detail than PNG and JPG files. PNG files also have a smaller file size than GIF files, but do not support animation. GIFs are perfect for animated graphics, and are fine for simple images that need to download quickly onto a user's computer.

host name

Name of a computer that has full two-way access to other computers on the Internet.

HTTP

Hypertext Transfer Protocol. Enables transfer of text in hypertext form, such as text written in HTML. HTTP servers manage the traffic of HTTP requests and responses. (Users make an HTTP request every time they enter a URL beginning with http://.)

image map

Image with more than one associated link. For example, a corporate home page could have an image of a site map. Users could click different parts of the site map image to display different sections of the corporate Web site.

IP address

Internet Protocol address, also known as an IP. A 32-bit number that identifies each sender or receiver of information across the Internet. Enables a computer to participate in the Internet.

ISP

Internet Service Provider. Examples: America On-Line (AOL), Global On-Line (GOL).

JPG

Also known as **JPEG**. Image file format (.jpg) created by the Joint Photographic Experts Group. JPG files support complex compression algorithms, enabling you to fine-tune the balance between file size and picture quality. JPG files have a larger file size than the other Web image alternatives, GIF and PNG files. However, when quality matters more than a quick download time, JPGs are ideal.

link

Also called a **hyperlink**. Text in a Web document that you can click to display another Web document. Hyperlinks are the connecting strands in the World Wide Web.

localhost

Host name for the computer that you are currently using.

mailto

Tag that creates an e-mail link. This link opens the user's e-mail program, if he has one, and begins a new message to the e-mail address contained in the `<mailto>` tag.

mapping

Association between the physical directories where your files are stored and the server that processes the files.

markup language

Language that structures information for display purposes, like HTML, or for data storage and retrieval, like XML. All markup languages are a subset of the Standard General Markup Language (SGML).

palette

Group of colors that you can use in developing your Web site. A "browser-safe" palette only has colors that are rendered the same across browsers.

path

Route that a computer takes to access or upload a file. Paths can be **absolute** or **relative**. An absolute path leads to a single possible location for the file to be accessed or uploaded; for example, C:\Program Files\Macromedia\HomeSite+ for Dreamweaver MX\UserData. A relative path leads to the file based on the location of something else; for example, code snippets are written to \UserData, on whatever drive and directory that you installed HomeSite+ for Dreamweaver MX. Other examples of paths: URLs, such as http://www.macromedia.com; IP addresses, such as 132.96.219.3; and *localhost* to signify the computer that you are currently using.

PNG

Portable Network Graphics. Type of image file (.png) that has a small file size but less detail than the JPG file. PNG files are perfect for images that need to download quickly onto a user's computer. They are smaller in file size and of higher quality than GIF files, but unlike GIFs they do not support animation.

project

In HomeSite+ for Dreamweaver MX, a project organizes the files in a Web site or Web application by grouping them in a central location. (This location can be physical or virtual.) Using projects simplifies deployment and maintenance tasks such as link verification and extended search and replace.

RDS

Remote Development Services. Proprietary protocol that is built into HomeSite MX for Dreamweaver. RDS does everything that FTP does, and also lets developers browse server databases, debug remote applications, and use server-side source control. The RDS server also manages security. The RDS server is always the ColdFusion Server.

RGB value

Code for a color, in hexadecimal (`[0-9A-Fa-f]`) notation. The code is comprised of three pieces, which signify the level of red, green, and blue in the color. Each of these pieces is either a 2-digit number from 00 to 99, or a 2-letter combination from AA to FF. Absence of a color is 00 or AA; the maximum concentration of a color is 99 or FF. For example, pure red can be denoted as 990000, 99AAAA, FF0000, and so on. Green is 00FF00 and blue is 0000FF. The Web has several great resources for getting the RGB value of colors; for example, see www.hypersolutions.org/pages/rgbhex.html.

schema

Structure or model for a database. A schema can be a graphical representation or a formal text description.

selector

Page element, or tag. TopStyle Lite CSS Editor uses this term in its user interface.

server

A computer or software program with the role of serving. For example, a database server (or **transaction server**) receives incoming requests from client computers and responds by "serving" either the requested data or an error. A Web server such as Apache or IIS returns HTML pages in response to requests, such as when a user enters the Web page's URL. An application server such as ColdFusion or JRun Server handles the business logic of a Web application and can process all sorts of Web languages beyond HTML; for example Perl, CGI scripts, servlets, ActiveServer Pages, Enterprise JavaBeans, and so on. An application server can also act as a Web server, but most developers use a specialized Web server like Apache to get extra features.

SSL

Secure Socket Layer. An Internet protocol for managing the security of a message transmission.

tag editor

Special type of dialog box created in the Visual Tools Markup Language (VTML) that enables users to complete information for a new or existing tag. These dialog boxes are slightly different from standard Windows dialog boxes; for example, they all have embedded Help and many of them have tabs. The tag editors for Anchor `<a>`, Body `<body>`, Image `<img>` tags are standard dialog boxes unless you select to use the VTML tag editors in **Options > Settings > Markup Languages > HTML/XHTML**.

thumbnail

Miniature depiction of an image. Usually, thumbnails for several images are placed together, so users can preview the thumbnails and decide what to do with the images, if anything.

Unicode

Unicode Worldwide Character Standard. System of setting up binary codes for text or script characters, so that the characters from the principal written languages of the world can be displayed and processed.

UNC

Universal Naming Convention. A UNC path allows you to identify a shared file on a computer without specifying the storage device it is on. A UNC path only works within a specified network; for example, in your Network Neighborhood. The UNC format is: \\servername\sharename\path\filename; for example, \\printserver\floor2printer.

URL

Uniform Resource Locator. A unique identifier or "address" for an Internet file. The URL contains at least the protocol (e.g., http or ftp) and the host name, but can also include a port number, lengthy file paths if the file is deeply nested on a server, and complicated query strings, especially if the file is dynamically generated like with JavaServer Pages (JSP). For security purposes, some URLs are hidden from the user.

UTF-8

USC Transformation Format (USC is the Universal Character Set). Encoding format that enables computers to handle both ASCII and Unicode.

W3C

World Wide Web Consortium. Creates specifications, guidelines, software, and tools to achieve its vision for the Web—that all users would equally experience the Web regardless of their browsers, operating systems, or physical disabilities; and that the Web would be a free flow of information, commerce, and communication.

watch

Expression or variable that you are monitoring, so that you can see its value at given points while the debugger steps through your code.

WDDX

Web Distributed Data Exchange, an open source XML-based technology that enables the exchange of complex data between Web programming languages, creating what some refer to as 'Web syndicate networks'. WDDX consists of a language-independent representation of data based on an XML 1.0 DTD, and a set of modules for a wide variety of languages that use WDDX. WDDX can be used with HTTP, SMTP, POP, FTP and other Internet protocols that support transferring textual data. For more information, see the WDDX Web site at www.OpenWDDX.org.

XHTML

Extensible Hypertext Markup Language. Reformulation of HTML as an XML application. It is almost identical to HTML 4.01, but more strict and clean. It is designed to replace HTML, and it works on most existing HTML browsers. For more information, see the W3C HyperText Markup Language Home Page at www.w3.org/MarkUp/.

XML

Extensible Markup Language. Defines the structure of information, or how the information could be stored in a database. For example, an XML language for cooking could have a `<recipe>` tag, and `<recipe>` could contain tags for `<header>`, `<ingredients>`, and `<instructions>`. The power of XML lies in the fact that the

information can be stored and retrieved from a database, rendered in a variety of formats, and used for a variety of purposes—for example on a Web page, hand-held computer, or cell phone; or for a cookbook, recipe card club, or book about cilantro.

Index

A

ActiveDocument Object 236
ActiveScript
  configuring product for 85
  examples in VTOM 282
ActiveX 85
advanced search operators 38
anchoring regular expressions to strings 95
answers, finding xvi
APF files 124
Application Object 205
arguments, inserting with Tag Insight 56
ASP
  color coding for 96
  support for 82
  validating code with 105
attributes
  viewing 55
Auto Backup
  options 47
  using 47
Auto Completion, about 57

B

backing up files 46
backups, managing 47
bookmarks
  creating in Help 36
  organizing 36, 49
breakpoints, setting 141
Browse tab
  about 29
  setting browser for 10
browsers
  configuring 9
  copying text from 59
  internal 10
  removing 13
  setting file saving behavior 13
  viewing list of 12
building SQL statements 118

C

Cascading Style Sheets. *See* CSS
CFML
  support for 82
character classes in regular expressions 93
Chinese, support for 3
classes, in CSS 78
code
  editing in Tag Inspector 76
  preserving with CodeSweepers 98
  selecting blocks 69
  tools generating 58
code snippets. *See* snippets
code templates
  about 57
  editing 57
  using 57
CodeSweepers
  about 98
  configuring 102
  controlling white space with 102, 103
  creating 100
  deleting 101
  editing 100
  for XHTML 89
  removing tags with 103
  running 99
  running unattended 102
  setting default 100
coding in XHTML 85
coding tools for XHTML 87
ColdFusion Administrator for data sources 114
ColdFusion, web resources xiii
collapsing text 70
color coding
  about 96
  changing schemes 97
compatibility of FTP & RDS node 14
configuring
  browsers 9
  CSE HTML validator 108
  data sources 114
  debugging sessions 137
  deployment servers 148
  Function Insight 56
  Mozilla 10
  product 1
  projects 124
  remote servers 114
  Resource Level Monitor 7
  servers 9
  spell checker 165
  Tag Insight 55
  Tag Tree 75
  validator 105
connecting to
  data sources 114
  FTP servers 15
  RDS servers 18
  servers 15
contacting Macromedia xvi
container/control examples 187
controlling source 134
converting
  case in tags 83, 102
  text files to HTML 59
copying
  text from browsers 59

web content 49
CSE HTML Validator, about 108
CSS
about 78
benefits of using 78
classes 78
managing site with 79
syntax for 79
CSS, color coding for 96
custom user dictionary, using 165
customizing
deployments 146
Help 39
QuickBar 52
toolbars 32
workspace 31

D

data sources
accessing 113
adding 114
opening 115
database tools 114
database, native drivers 114
DBCS 3
debugger
about 136
output of 141
running 140
using 136
windows 141
debugging
about 135
breakpoints for 141
evaluating expressions 142
form submittal 142
setting up for 137
watches for 142
declarations, DOCTYPE 83
default template, editing 59
defaults
changing for browsers 13
changing for security 19
for CodeSweepers 100
for deployment 144
for mappings 24
for projects 124
defining
code templates 57
server mappings 23
deleting
browsers from list 13
CodeSweepers 101
custom toolbars 34
projects 129
servers 19
snippets 62
deploying
adding server for 148
basic method 145
editing server properties 149
files 145
folders 145
projects 150
projects, custom deployment 146
removing servers for 149
reviewing results of 153
scripts for 150
setting default options 144
to multiple servers 145
with wizard 149
DeploymentManager Object 258
developer resources xii
development mappings. *See* mappings
dialog definition files 179
directories
default for snippets 62
default for templates and wizards 59
for FTP servers 16
displaying
contents of Title tag 171
external links 171
source control toolbar 134
text files in Help tree 40
toolbars 32
docking
QuickBar tabs 32
toolbars 32
DOCTYPE declarations 83
document templates 58
Document Type Definition. *See* DTD
Document window, creating files in 29
documentation
about xiv
accessing 35
adding tag Help 186
adding to Help References 39
editing tag Help 35
extending Help system 38
in Tag Chooser 35
in tag editors 35
opening favorites 36
printing xv
using index 37
using search engine 37
viewing xiv
DocumentCache Object 247
double-spaced lines, replacing 163
download times, testing 172
downloading web pages 49
dragging files from Windows Explorer 48
Dreamweaver
inserting MS Office content with 60
integration with 33
visual editing and prototyping in 58, 60
drive mappings 22
DTD
color coding for 96
support in Tag Tree 75

E

Edit tab 29
editing pages 67
Editor toolbar, options 68
enabling
Function Insight 56
non-ANSI file encoding 43
RDS security 19
SSL for FTP connections 17
Tag Insight 55
XHTML support 87
encoding
about 43
enabling in non-ANSI files 43
opening encoded files 44
saving files with 46
evaluating expressions 142
event handler scripts 77
examples
ActiveScripting 282
Add script 278
ApplicationType script 205
containers and controls 187
Cookie script 267
CurrentView script 207
deployment script 263
download script 274
POST script 273
RDS file mapping 21
regular expressions 95
startup script 286
toolbar scripts 234
validating for HTML 4.0 106

validating for XHTML 1.0 89
wizard definition pages 193
Expression Builder
about 58, 177
creating regular expressions in 91
editing definition files for functions 56
location of files 297
expressions. *See* regular expressions
extended characters. *See* special characters
extended find and replace 160

F

favorites. *See* bookmarks
file saving 13
File Transfer Protocol. *See* FTP
file transfers, file size after 17
file types
modifying color coding for 96
setting for project folders 128
files
changing display of 48
deploying 145
editing included 72
encoding, about 43
inserting into documents 59
managing 41
managing in projects 127
on remote servers 14
opening 44
working with 44
Files tabs, about 42
filtering file list 48
finding answers xvi
Fireworks, integration with 71
folders
adding to favorites 48
adding to Help 39
adding to projects 126
deploying 145
managing in projects 127
formatting
with CodeSweepers 98
with CSS 78
forms, testing submittal of 142
freeing up resources 6
FTP & RDS node 14
FTP servers
connecting to 15
enabling SSL for 17
setting root URLs for 173
Function Insight
about 56
configuring 56
inserting arguments with 56

G

GMT time format 17
Graphical User Interface. *See* workspace
graphics, integration with Fireworks MX 71
GUI. *See* workspace

H

HDML, support for 82
Help. *See* documentation
hiding toolbars 32
HTML Tidy
about 98
creating CodeSweepers 100
deleting CodeSweepers 99, 101
editing CodeSweepers 99, 100
upgrading 98
HTTPProvider Object 266
hyperlinks. *See* links

I

ICW. *See* initial configuration wizard
Image Map Editor 58
images
adding to files 45
editing 71
image maps 58
thumbnails 71
using Fireworks MX for 71
IMFL, support for 82
importing
DTD information 76
outline profiles 75
previous configurations 3
included files, editing 72
Initial Configuration Wizard 3
inline coding tools 55
inserting
arguments with Function Insight 56
code snippets 62
column names 115
files 59
MS Office content 59
SQL queries 119
table names 115
tags from QuickBar 52
tags with Tag Insight 55
installing
Mozilla 10
newer version of HTML Tidy 98
product 3
technical support for xii
integrating
browsers 12
CSE HTML validator 108
projects with VSS 133
interactive debugger.*See* debugger
internal browser, configuring 10
Internet Explorer as internal browser 10

J

Japanese, support for 3
Java
color coding for 96
support for 82
JavaScript
color coding for 96
reference tree 58
wizard 58
JRun, support for 82
JScript, color coding for ASP with 96
JSP, support for 82

K

keyboard shortcuts
about 61
creating 61
for snippets 63
printing list of 61
Korean, support for 3

L

language support 82
languages, detection of 83
links
adding to open file 45
checking in Site View 171
editing 171
verifying 169
local paths, RDS file mapping 21
localhost, deploying projects to 150
lowercase
converting tags to 102
setting conversion to 83

M

Macintosh, saving files formatted for 45

Macromedia
headquarters xvi
sales xvi
website xii
Macromedia CodeSweepers. *See* CodeSweepers
managing
backups 47
bookmarks 36
code snippets 62
CodeSweepers 100
ColdFusion security for RDS 19
deployment scripts 152
files 41
projects with source control 132
servers 19
snippets 62
mappings
adding 23
editing 24
for debugging 20
for page processing 20
setting default 24
switching 24
media content, adding 40
Microsoft Office, inserting content from 59
Microsoft VSS, using in projects 133
modems, testing download times 172
monitoring system resources
about 6
responding to warnings 7
setting options for 7
moving
QuickBar tabs 32
toolbars 32
Mozilla, integrating with product 10
multi-character regular expressions 94
multi-user projects in VSS 134

N

namespace precedence
about 109
setting 109
native database drivers 114
navigating document structure 74
Netscape, integrating with product 10
Network Neighborhood, using for mappings 22
non-ANSI file encoding 43

O

OLE-DB drivers 114
opening
data sources 115
encoded files 44
files 44
Help in tag editors and Tag Chooser 35
included files for editing 72
pages from Web site 49
projects 129
recent files 45
source control application 134
SQL Builder 116
tag editors 73
options
Auto Backup 47
clipboard limit 69
CodeSweepers 102
debugging breakpoints 140
default CodeSweeper 100
default mappings 24
defaults for deployment 144
defaults for projects 124
Editor 68
file types for color coding 96
folder deployment 147
importing from previous versions 3
initial project resources 130
internal browser 10
link verification 169
markup languages 83
namespace precedence 109
project deployment 146
projects 124
root URL for FTP servers 173
spell checker 165
startup 44
Tag Tree display 97
validator rules 105
watches for debugger 142
workspace 27
XHTML 87
order
changing in Help tree 39
changing in QuickBar tabs 31
changing in toolbar buttons 33
of namespaces 109
organizing
favorite folders 49
QuickBar tabs 31
toolbars 31
outline profiles
about 74
creating 74
editing 74
editing list of 75
importing 75
output of debugger 141

P

passive mode, server connections 17
passwords for FTP servers 16
pdf documentation xv
Perl, color coding for 96
PHP
color coding for 96
support for 82
validating code with 105
POSIX character classes 93
preserving code formatting 98
printing
Help topics 35
keyboard shortcuts 61
pdf documentation xv
priority of namespaces 109
project (APF) files 124
Project Object 250
ProjectManager Object 252
projects
about 122
adding files to 127
closing 129
creating 124
custom deployment of 146
deleting 129
deleting resource types from 131
deploying with wizard 150
deployment options for 144
deployment servers 148
editing folders in 127
editing properties in 129
editing resource types in 131
integrating with VSS 133
managing with source control 132
opening 129
options for 124
project (APF) file 124
removing files from 128
removing folders from 127

removing resources from 131
setting defaults for 124
types of folders in 122, 123
using VSS with 133
verifying links in 170
viewing resources for 130
working with 128
properties
deployment scripts 152
for RDS servers 19
of deployment servers 149
of projects 129
proxy servers 49

Q

queries
editing 120
inserting new 119
inserting saved 119
saving 118
testing 120
QuickBar
customizing 52
inserting tags from 52
moving 32
organizing 31

R

RDS
about mappings for 21
file mapping examples 21
mapping with 23
servers, connecting to 18
recent files, opening 45
recovering unsaved files 47
referenced files, editing 71
RegExp. *See* regular expressions
regular expressions
about 91
back references in 94
examples of 95
searching with 164
single-character 92
using 91
using Expression Builder for 91
releasing system resources 6
remote servers
adding 114
removing 19
removing
buttons from toolbars 34
deployment scripts 152
deployment servers 149
files from project folders 128
HTML Tidy profiles 99
project folders 127
project resources 131
replacing
double-spaced lines 163
extended and special characters 163
text 159, 161
required server information 15
resource level monitor
about 6
configuring 7
responding to warnings 7
using 7
resource tabs, about 28
resources
adding from Projects tab 131
releasing system 6
working with 130
Resources window, working in 28
Results window
about 30
working in 156
results, generating document of 157
reverting to installed certificate and key 18
reviewing deployment results 153
root URLs, setting for FTP servers 173
RTML, support for 82
rules
CodeSweeper 103
validator 105

S

Sample 274
saving
code as snippets 62
deployment results 153
files 45
files with encoding enabled 46
queries 118
search results 157
search text 158
text to clipboard 69
schemes, for color coding 97
scripts
editing deployment 152
removing deployment 152
running deployment 152
searching
about 158
all open documents 160
current document, basic 159
current document, extended 160
extended find and replace 160
replacing double-spaced lines 163
replacing special characters 163
saving search text 158
selecting text for 158
with regular expressions 164
with saved search string 159
working with results of 156
Secure Socket Layer. *See* SSL
security, RDS settings for debugger 141
selecting
code and text 69
files for deployment 146
search text 158
server mappings. *See* mappings
servers
adding for remote data access 114
configuring 9
deleting configuration for 19
editing configuration of 19
editing properties for deployment 149
setting up. *See* configuring
settings. *See* options
sharing
project files in VSS 134
snippets 63
Shell Namespace Extension. *See* SNE node
shortcut keys. *See* keyboard shortcuts
showing toolbars 32
site view, checking page links in 171
SMIL 82
SNE node
about 14
compatibility of 14
snippets
creating 62
default directory for 62
deleting 62
editing 62
inserting 62
keyboard shortcuts for 63
managing 62
sharing folders of 63

source control
benefits of using 132
from within product 134
managing projects with 132
supported systems for 132
toolbar for 134
special characters
replacing 163
using in regular expressions 92
using palette for 58
spell checker
about 165
configuring 165
options for 167
running 168
user dictionary, about 165
SQL
color coding for 96
inserting into pages 119
statements, changing type of 118
where clauses, using variables in 118
SQL Builder
opening 116
saving queries in 118
select statements, writing in 118
statements, writing in 117
testing queries in 120
user interface, about 117
using 116
SSL
enabling for FTP connections 17
reverting to installed certificate and key 18
startup options 44
style editor
about 79
styles
CSS syntax 79
supported
languages 82
source control systems 132
supports 108
switching mappings 24

syntax and usage Help, accessing 73
system requirements 2

T

tag case conversion
for current document 102
setting Editor option for 83
Tag Chooser
customizing Help in 35
customizing with VTML 177
using 53
viewing Help in 35
Tag Completion, using 56
tag definitions
about 109
creating 180
editing 110
file structure of 180
priority of 109
tag editors
completing 54
creating 182
customizing Help in 35
editing tags with 73
opening 73
using 73
viewing Help in 35
VTML versus standard 73
Tag Insight
about 55
configuring 55
editing tags with 55
inserting tags with 55
Tag Inspector
about 76
editing tag definitions with 110
viewing property sheet for tags 76
tag sets, priority of 109
Tag Tree
about 74
configuring display of 75
importing outline profiles for 75
modifying outline profiles in 75
outline profiles, about 74
outline profiles, creating 74
tags
adding Help for 186
completing with tag editors 54
editing with tag editors 73
editing with Tag Insight 55
editing with Tag Inspector 110
inserting with Tag Insight 55
requiring attributes for 106
setting CodeSweeper rules for 103
validating 107
viewing attributes for 55
viewing Help for xiv, 73
templates
about 58
creating for wizard output 196
default directory for 59
editing default 59
saving document as 59
testing
page download times 172
queries 120
web pages 155
text, selecting 69
thumbnails, displaying 71
To 49
toolbars
adding 34
adding buttons to 32
deleting custom 34
moving 32
organizing 31
removing buttons from 34
showing and hiding 32
tutorials
resources for xiii

U

UNC paths, using for mappings 22
Unicode Big Endian, support for 43
Unicode, support for 43
UNIX, saving files formatted for 45
upgrading product 3
uppercase
converting tags to 102
setting conversion to 83
user dictionary, about 165
user interface. *See* workspace
UTC time format 17
UTF-8, support for 43

V

validating
code 104
configuring for XHTML 89
configuring validator 105
current document 107
requiring attributes when 106
rules for XHTML 90
single tag 107
using default validator 104
XHTML 89

VBScript, color coding for 96
verifying
  links 169
  links in document 169
  links in project 170
version source control, supported systems 132
viewing
  data sources 115
  favorite folders 49
  integrated browsers 12
  links in site view 171
  online Help xiv
  pages in external browsers 60
  pages linked from document 171
  project deployment servers 149
  results of extended searches 162
virtual folders 123
visual editing 60
Visual SourceSafe. *See* VSS
VSS, using in projects 133
VTML
  attrib tag variables 183
  attribute categories, defining 181
  bound controls 193
  building custom wizards with 192
  color coding for 96
  defining controls with 182
  dialog definition files 179
  dynamic expressions in tags 193
  examples of containers and controls 187
  examples of wizard definition pages 193
  extending language support with 82
  generating tags from tag editors 183
  populating dialog boxes with data 183
  Tag Chooser, customizing with 177
  tag definitions, creating with 180
  tag editors, creating with 182
  tag Help, creating with 186
  TAGDATAUnknownAttributes tag 185
  WIZ tags, about 197
  wizard definition page library 198
  wizard definition pages 193
  wizard output templates 196
VTML tag editors, about 73
VTOM
  ActiveDocument Object 236
  Application Object 205
  CommandID values 288
  creating toolbar for custom scripts in 204
  DeploymentManager Object 258
  DocumentCache Object 247
  example, Add script 278
  example, ApplicationType script 205
  example, cookie script 267
  example, CurrentView script 207
  example, deployment script 263
  example, download script 274
  example, POST script 273
  example, startup script 286
  examples of ActiveScripting 282
  examples, toolbar scripts 234
  executing scripts in 202
  HTTPProvider Object 266
  Project Object 250
  ProjectManager Object 252
  scripts, third-party add-ins 286
  SettingID values 292
  toolbuttons, creating custom in 204
  writing scripts in 202
  ZIPProvider Object 276

W

watches, setting in debugger 142
WDDX, CodeSweeper for 98
web
  downloading pages from 49
  languages, using 81
where clauses, using variables in 118
white space, controlling with CodeSweepers 102, 103
Windows Explorer
  dragging files from 48
WIZ tag set, about 197
wizards
  about 58
  building with VTML 192
  default directory for 59
  definition page library for 198
  definition pages for 193
  deployment 149
  initial configuration 3
  JavaScript 58
  JavaScript Tree 58
  output templates for 196
WIZML tag set, about 193
WIZML, about 196
WML, support for 82
workspace
  about 25
  components of 26
  customizing 31
writing
  SQL statements 117
  web content 51
WYSIWYG editor, using 60

X

XHTML
  about 86
  CodeSweeper for 89
  coding in 85
  coding tools for 87
  color coding for 88, 96
  configuring validator for 89
  enabling support for 87
  setting options for 87
  support for 82
  validating 89
  validation rules for 90

Z

ZIPProvider Object 276