



WAP–Enabling a Website with PHP3

By Mike Banahan

All materials Copyright © 1997–2002 Developer Shed, Inc. except where otherwise noted.

Table of Contents

<u>Introduction</u>	1
<u>How it Works</u>	3
<u>Switching to WML</u>	5
<u>Conclusions</u>	7

Introduction

The [UK Geographic Search Engine / Guide](#) was an idea whose genesis occurred whilst our team were having their pre-Christmas drink in [The Park](#) on the day before Christmas Eve, 1999. We'd been kicking around a number of ideas about how many of the Internet's users didn't 'get' what it could really be made to do when the concept of removing the virtuality and injecting some *reality* just sort of bubbled up out of thin air. Fuelled with demob-happiness some crazy ideas got discussed and then the blinding flash of the obvious hit us: why can't you search the net based on how near things are to each other rather than the text that documents contain?

Going from the thought to the execution always takes time but we were lucky. As a hobby project (mainly to learn the Qt widget set) I had already built a [moving-map GPS package](#) to run on a Linux laptop while driving. Very basic, nothing flashy, but at least it had a crude mapping engine in it. About an hour of hackery on the code hooked it into our php3-based contact manager software and presto: all of our customers' offices popped up on the maps too!

By the end of January a rough prototype of Somewherenear had been built. The location data is stored in a [MySQL](#) database and distance-based searching performed very simply by using SQL 'select' statements with 'where' clauses that limit the latitude and longitude of the items sought. By late March we were ready to release the thing to the world and did, feeling pretty pleased to snaffle the domain name. We know that the radius-selection algorithms won't scale but after chatting with Stuart Green (thanks Stuart), ex Leeds University, we are confident that techniques used in molecular physics can be applied to give us very fast radius detection, at the cost of lots of disk space.

We had always planned to hook the database into location-aware devices. Whether they would be mobile phones or PDAs with GPS in them wasn't clear, but it was obvious that before long people would be out and about with devices that contain browsers and also know about their location. We knew that we would probably have to deliver the data out of the system in various formats. HTML was obvious, and probably WML too, along with other XML derivatives if we proved to have built a useful resource.

Our goal was to be able to go to a strange town, press the 'nearest pub' button on whatever the device was and to get a map showing establishments nearby, together with visitors' ratings. Having gone there, you should be able to add your own vote on the quality of the beer, the food, the surroundings and so on, then ask for the nearest good curry house and cashpoint to pay for the food. (Non UK-readers may not be aware of the mystic significance of curry and beer, but believe me, it is *extremely* important to the UK software industry. Curry houses are by far the most common food outlets in Britain. Even Macdonalds have had to introduce Chicken McTikka to be able to compete; it's just as appealing as their other offerings.)

We had an architecture already prepared for this kind of project, based on CGI programs and pseudo-HTML templates. A description of the [TrainingPages training search engine](#) was written up some time ago and has been described in a number of talks I have given. The architecture has some known deficiencies so we thought it was time to move it up to its next stage. Its greatest weakness is that HTML ends up embedded in the CGI scripts, making them specific to the delivery format, plus the lack of enough intelligence in the template language itself. Moving on to the next step, I decided to replace the template language with php3 but to enforce a rigid separation of responsibilities. **The CGI scripts must do all the data management, the templates must deal with presentation only.** This rule is an absolute winner and has proved to be hugely effective.

WAP–Enabling a Website with PHP3

We still write the CGI programs in C++. It's a proper heavyweight engineering language and frankly I still don't think that Java cuts it on the server side. C++ templates provide so much power to the designer and the ability to use exception–handling to catch errors has saved us acres of code, proving in my mind superiority over Perl or php3 for the core data engine.

This article copyright GBdirect Ltd 2000. All rights reserved. Reprinted from <http://www.gbdirect.co.uk/> with permission.

How it Works

The CGI program runs on response to every incoming query and is easily fast enough to fill our data pipe without having to resort to trickery like Apache modules or fastCGI. Each execution of the CGI inspects the incoming form and uses that to decide which presentation template should be used. These presentation templates (nothing to do with the C++ `template` construct) are php3 pages. The CGI program's output is essentially a long list of php3 data definitions; here's a fragment:

```
<? $logfile="/www/web.places/logs/dblog";
header("Set-cookie: name=3880d6350463; expires=Saturday,
23-Mar-2002 06:08:30 GMT; path=/");
$Mytemplate["formmethod"]="get";
$Mytemplate["sessionkey"]="3880d6350463";
$Mytemplate["retrieved_cookie_session"]="Fri Apr 21 20:11:05
2000 "; $Mytemplate["lasttimeon"]="956344265";
$Mytemplate["placetype"]="1";
$Mytemplate["QUERY"]="a=show_linkacetype=1";
$Mytemplate["mapimagewidth"]="292";
$Mytemplate["mapimageheight"]="219";
$Mytemplate["east"]="414996"; $Mytemplate["north"]="435499";
$Mytemplate["placename"]="Park"; $Mytemplate["title"]="Place
Details, Park"; $Mytemplate["OSGB"]="SE 149 354";
$Mytemplate["latlong"]="053 48\" 55'N 001 46\" 19'W";
```

... and a *lot* more like it.

After spitting out the data, the CGI program opens the php3 template file and tacks it onto the end of the data definitions, so it's acting much like the `cat` command. The whole pasted-together chunk is sent through a pipe into the php3 interpreter and the output of *that* is also piped asynchronously through `weblint`; if weblint picks up any errors, it emails us to warn that we are generating bad HTML. All this is done for every page generated. Running on an AMD K6/250 it looks as if we can do about one page per second, including the map generation before we would have to turn off the weblinting or look for more power.

Here is a fragment from the `place` php3 template, this is the one that executes if you click on the link to [The Park](#) pub:

```
<? require('stdhead.tpl'); require('stdplace.tpl'); $thisplace
= new PlaceDetail(); if (isset($place)) {
$thisplace->import($place[0]); } $thistypes = new TypeList();
if (isset($type)) { $thistypes->import($type); } $REenter =
"sessionkey=" . R("sessionkey") . ". R("rlimit") . ".
R("east") . ". R("north") . ". $thisplace->unique_id . ".
R("placetype"); if(I("o-east")){ $REenter .= ". R("o-east") .
". R("o-north"); } $basicformfields = "<input type=hidden
name=a value=\"show\">\n"; $basicformfields .= "<input
type=hidden name=placetype value=\"\" . R("placetype") .
\">\n"; $basicformfields .= "<input type=hidden
```

WAP-Enabling a Website with PHP3

```
name=sessionkey value=\" $K\">\n"; $basicformfields .= "<input
type=hidden name=rlimit value=\"\" . $Mytemplate["rlimit"].
"\n">\n"; if (I("o-east")){ $basicformfields .= "<input
type=hidden name=o-east value=\"\" . $Mytemplate["o-east"].
"\n">\n"; $basicformfields .= "<input type=hidden name=o-north
value=\"\" . $Mytemplate["o-north"] . "\n">\n"; }
$basicformfields .= "<input type=hidden name=east value=\"\" .
$Mytemplate["east"]. "\n">\n"; $basicformfields .= "<input
type=hidden name=north value=\"\" . $Mytemplate["north"].
"\n">\n"; $basicformfields .= "<input type=hidden
name=\"unique_id\" value=\"\" .s $thisplace->unique_id .
"\n">\n"; $votepics = array("twocross.gif", "onecross.gif",
"onetick.gif", "twotick.gif", "threetick.gif"); $votealts =
array(":<", ":((", ":", "|", ":", ":", ":", ">"); // and a lot more besides
...

```

It's probably obscure unless you know php3 but I thought I'd include it for those who know it. A lot of the calls like `R()`, `I()` and so on are functions defined in `stdhead.tpl` – they extract data from the predefined arrays but also report errors if expected data is missing, helping us to debug the interface between the CGI and the php.

That all works – the CGI→php3→weblint→user chain is highly effective. By having the logic in the C++ and the HTML generation in the php3 two people or teams can work simultaneously on the project without getting in each other's way much and what's more, if you are tinkering with layout you **know** you can't break the data engine or corrupt the database easily. This is typically **not** the case with 'traditional' php3 work where data management and layout get badly intermingled. At least they do when I'm writing the stuff, maybe there are some star performers who can manage to do better; I take my hat off to them if they exist.

This article copyright [GBdirect Ltd](http://www.gbdirect.co.uk/) 2000. All rights reserved. Reprinted from <http://www.gbdirect.co.uk/> with permission.



Switching to WML

After we got the HTML site going, the next task was to WAP it up. The first concern was how much we would have to hack php3 to tell it not to emit a first line that says

```
Content-type: text/html
```

Although I couldn't find it documented anywhere, I found to my deep and abiding joy that if you use the `php3 header()` function to output a different content-type header, that suppresses the default. Bingo! Every WML card deck now includes this php fragment at the top:

```
<? header("Content-type: text/vnd.wap.wml"); echo "<?xml
version=\"1.0\"?>\n"; echo "<!DOCTYPE wml PUBLIC
\"-//WAPFORUM//DTD WML 1.1//EN\" \" . \"
\"http://www.wapforum.org/DTD/wml_1.1.xml\">\n"; ?>
```

and that deals with all the usual crud you have to cough out to get started with WAP. From there on it was plain sailing. It took six hours to a) learn enough WML to get something done and b) implement it. I used Steve Mann's book "Programming Applications with the Wireless Application Protocol" (Wiley, ISBN 0-471-32754-9) as my reference, plus a few downloaded tutorials from places like [Waptastic](#) for backup. I wouldn't recommend any of them. Considering how restricted the WML language is, you would have thought it wouldn't take much explaining, but I have yet to see a decent tutorial on it. Maybe I am just too dim to pick it up easily. That might also account for how limited our current WAP version of the site is.

Of course you can't rely on WML being correct without testing it. If there is a WML weblint-alike, I'd like to find it, but haven't looked hard enough yet. It would be a big help to ensure that the WML is well-formed on the way out. You also have to check that your WAP site 'works', to the degree that you can make anything work through such a limited medium. There are a number of WAP phone simulators around. There is a web-based one at [Gelon](#) as well as others from [phone.com](#) and the marvellously named [Slob-Trot](#) in Finland. Sadly both the latter two only run under Windows so far as I can tell, but [vmware](#) came to the rescue. Or, at least it did for the Slob Trot browser, the phone.com version ran but didn't paint any images on the screen. I later found a native Windows machine and it seemed OK on that.

Using the simulators to test the WML allowed the php3 templates to be written very quickly. Here is the full 'place' template:

```
<? require("stdwaphdr.tpl"); require('stdplace.tpl'); ?> <?
$votepics = array("xx", "x", "+", "++", "+++"); ?> <card
title="Search results" id="first"> <p>
<?htmlTplacetyname();?>(s) near <? htmlT("placesearch");
?><br/> <? if(isset($places)): ?> <? $placelist = new
PlaceList; $placelist->import($places); $placelist->reset();
$count = 1; while($place = $placelist->next()): ?> <? echo
htmlspecialchars("\$place->name\", \" .
$place->county_address()); ?> <? printf("(%.2fm) \",
($place->distance_km()*100/1.6)/100); ?><br/> <?>
```

WAP-Enabling a Website with PHP3

```
if(isset($Objects[$place->unique_id . "votes"])){ $arrayref =
$Objects[$place->unique_id . "votes"]; // echo "Scores:";
$first = 1; while(list($key, $value) = each($arrayref)){ $idx
= $value - 1; if($idx < 0) continue; $vname =
$Objects["voteranktoshortnames"][$key]; if ($first) { $first =
0; } else { echo ", "; } if (($idx < sizeof($votepics)) &($idx
>= 0)) { echo "$vname $votepics[$idx]"; } else { error("Vote
value out of range"); } } echo "<br/>\n"; } else { // echo "No
votes yet<br/>\n"; } ?> <? endwhile; ?> <? else: ?> No
matches! <? endif; ?> Go back<anchor
title="back"><prev>back</prev></anchor> </p> </card> <?
require("stdwapftr.tpl"); ?>
```

This article copyright [GBdirect Ltd](http://www.gbdirect.co.uk/) 2000. All rights reserved. Reprinted from <http://www.gbdirect.co.uk/> with permission.

Conclusions

It's dead easy if you follow the path of separating the data management from the presentation. Our approach works fine for us and we can easily add features to the WAP part of the site to bring it up to a similar spec to the main HTML site. Whether we would want to is another matter. WML is pretty ropey and it's my own opinion that people used to HTML browsers will be extremely frustrated if they get suckered by the hype about surfing the net from your mobile phone. Screens the size of typical mobile phones are well-nigh unusable and devices with more real estate will have enough power to run HTML browsers. The data compression of the WAP protocol is another thing – couple that with HTML and you will have something much more interesting.

The drawbacks that I've seen from using php3 like this are minor. The `htmlspecialchars()` function in php3 doesn't deal correctly with the WML entities that need escaping, but I intend to write my own version and put it in the `stdwaphdr.tpl` include file. If php3 finds a syntax error it tries to be nice and tell the user via the browser – but it outputs HTML not WML while it does it. Hardly a show-stopper if you have got your php3 code right in the first place, but should it happen then the user's browser will choke. I'm told that WAP phones are even less reliable than Windows at the moment: an article in this month's Personal Computer World describes having to remove the battery from the phone when it gets wedged.

It works, it's simple and I'm sure we will be doing more with this.

Mike Banahan, GBdirect, 22nd April 2000

Update 22nd May 2000

If you want a copy of the templates used on the wap site, I have prepared a [gzipped tarfile](#) of them which you can use for information. Note that the server will serve them up as text/plain – just click on 'save as' and drop the file into an appropriate directory before unpacking it. The files will unpack into the current directory, so make sure you put the archive somewhere sensible first. If you spot any errors, I'd love to know.

In response to a number of comments I've received to this piece, I have written a [description of experiences developing WAP/WML pages](#) as a follow-up item.

This article copyright [GBdirect Ltd](#) 2000. All rights reserved. Reprinted from <http://www.gbdirect.co.uk/> with permission.