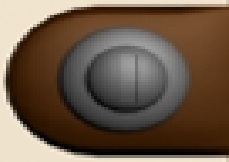
S.I.

Beginners Guide



AIML

Scripted Intelligence

Beginners Guide AIML

A simple guide to building an A.I. Brain

ScriptedIntelligence@hotmail.com

S.I. Beginners Guide AIML was brought to you by Donald Nadeau Jr.

Donald is a Artificial Intelligence, Home Automation, Robotics, and Programming enthusiast. Donald is also the CEO of SyberShot Studios, and owner of Scripted Intelligence, and Quantum Oculus

Table of contents

What is AIML?

AIML (Artificial Intelligence Markup Language) is a data structure/markup language that's used by a number of popular Internet chat bots. It's a "subset" of XML (Extensible Markup Language), and was created and developed by Dr. Richard Wallace in 1992. An A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) AIML tag set has been released under GNU General Public License (GPL).

I'm not going to get into to many details apart from what was stated above about what AIML is, reasons being is that this a beginners guide to scripting AIML for beginning Bot masters. Most people that are going to read this publication already know what AIML is and what it is used for.

To put it in the simplest of terms, AIML is the scripting language used by chatbot engines to reply to a user’s input. Example: If a user types in the word “hello” the chatbot engine searches through the AIML for the pattern ~hello~ and responds to the user with a predetermined reply that the Bot master has placed for the said pattern.

Creating your first AIML:

AIML can be created in virtually any text editor, for this tutorial we will be using Windows Notepad. First open windows notepad text editor and add the following lines of code below the paragraph, you can either type them or copy and paste them. In the upper toolbar click “File” then click “Save As”. You should now have a pop up “Save As” window. Choose a directory where you would like to save this this AIML file, the directory I usually use is “My Documents”. Now under “File Name” you name your file anything you desire just be sure to save it with an “aiml” extension by adding .aiml at the end of your chosen name. (Here Is An Example: MyBrain.aiml) Now Change the encoding to UTF-8 and click “Save”.

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">

</aiml>
```

You just created your first AIML file, but note this is a blank AIML file. It is the bare minimum code required. Let's take a look at line by line.
The first line (<?xml version="1.0" encoding="UTF-8"?>) is your doc type, it declares what Doc type is being used. You can also use (<?xml version="1.0" encoding="ISO-8859-1"?>) as your Doctype but I prefer the later for it is shorter.
The second line is the base code known as an AIML tag (<aiml version="1.0">)
All AIML files end with a closing </aiml> tag. The ending tag means the complete structure of the code ends there. Note you can have AIML tags within another, for example:

```
<aiml version="1.0">
<category>
</category>
</aiml>
```

Categories:

The category always begins with a category tag, and ends with a closed category tag. The pattern is what one person will write within the input field of a chatbot and also always ends with the </pattern> tag. **Note:** it is crucial that the text between pattern tags must always be capitalized. Template tag is the bots response to the pattern. Your AIML should have many Categories, containing one subject only to keep your work organized.

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
<category>
<pattern>HI</pattern>
<template>Hello, what's up?</template>
</category>
</aiml >
```

One will rarely want to write a chatbots brain from scratch, for it would be a very time consuming and daunting task to create all the categories needed for a bot to hold a decent conversion. The Alice AIML set is a great starting place for a pre-built brain, but one will have to go through and replace responses line by line to acquire uniqueness, it is a time consuming task, but far less daunting than starting from scratch.

Random responses:

Let’s take a look at this code first before I explain.

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
<category>
<pattern>HELLO THERE</pattern>
<template>
<random>
<li>Hi there, How are you doing on this beautiful day?</li>
<li>Hello, what's going on?</li>
<li>Hi,  What would you like to chat about?</li>
</random>
</template>
</category>
</aiml>
```

Now, what does random mean? In plain terms it means a random answer. The bot can simply choose an answer from the library of random answers and pick one that the bot will respond with. Depending on your chatbot engine it will usually respond in the order which the random replies are created in, but with a good chatbot engine replies are literally chosen randomly. The <li> tag basically means library. It's a full library of what the bots responds could be for a certain pattern. The library tag also always ends with the tag.

Fun and useful tags

Here are a few tags that I find that can be fun and useful at the same time. I'm sure you will find some amazing things to do with these tags with a little practice.

<explode> The purpose of the <explode> tag is to separate a word into individual letters, or a number into individual digits.
<explode>EXLODE</explode> = E X P L O D E

<date> The date tag allows your bot to give the date, time, and day. Here is a small example.

```
<category><pattern>WHAT YEAR IS THIS</pattern>
<template><date format="%Y"/>.</template>
</category>
```

<condition> This is an extremely fun and useful tag that allows conditions to be set in-order for a proper reply.

```
<pattern> HOW DO I SMELL</pattern>
  <template> I think you smell
    <condition name="gender" value="male"> gruesome </condition>
    <condition name="gender" value="female"> sweet </condition>!
  </template>
```

<that> The funnest tag for me is the <that> tag, even though it is a more advanced AIML technique it is still a very useful and fun to use tag once you grasp the concept. The following example is from my friend Dave Morton (AKA GeekCaveCreations) in his post here at Program-o.com

```
<category>

 <pattern>WHAT DID YOU SAY</pattern>

 <template>I said "<that index="1"/>"</template>

</category>
```

```
<category>
  <pattern>WHAT DID YOU SAY BEFORE THAT</pattern>
  <that>I SAID *</that>
  <template>My response was "<that index="3"/>"</template>
</category>
```

Input: Hello.

Output: Greetings, Seeker!

Input: Say something smart.

Output: Something smart.

Input: What did you say?

Output: I said "Something smart."

Input: What did you say before that?

Output: My response was "Greetings, Seeker!"

These are just a few basic steps on creating a chatbots brain. Now, there are a lot of advanced options and tags to go through. If you find it interesting, please continue reading.

Advanced AIML techniques:

AIML is one of the most advanced building blocks of modern chatbots. It creates the structure and style that surrounds your bots cognitive content and is capable of making your chatbot a joy to chat with or a pain in the rear. Mastering AIML is one of the most important things a Bot master can do, and has really become an essential criteria for being a successful Bot master.

Learning to implement these advanced techniques will not only make your bot more enjoyable to chat with, but they will also help you out as a Bot master in more ways than one as well. Let's look at a few of these techniques to see how helpful they can actually be.

So far you learned about matching only one input and not another form of the same question that might be asked by the user. This would be a tedious task, for any Bot master, however AIML has a couple of tags that can be beneficial for this scenario and save you plenty of time scripting your bot.

Two of those tags are the wild card, they are used within the pattern and the SRAI tag used within the template. In the next sub chapter of this section we will look at the wild cards.

wildcards:

A wildcard is a character that looks like this * and this _
Let's say you want to provide the same template/answer for
"WHAT IS SCRIPTEDINTELLIGENCE DOT COM",
"WHAT IS SCRIPTEDINTELLIGENCE WEBSITE" and
"WHAT IS SCRIPTEDINTELLIGENCE URL" this means normally you would have to write three different patterns. Using a wild card you can save yourself some scripting by using a the * character like so.

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
<category>
<pattern> WHAT IS SCRIPTEDINTELLIGENCE *? </pattern>
<template>Scripted Intelligence is a website about artificial intelligence and chatbots</template>
</category>
</aiml>
```

Note the wild card will match infinite number of words and, of course, input questions like WHAT IS SCRIPTEDINTALLIGENCE ALL ABOUT? as well.

The _ wild card works the same way but is the most divine of the two, for the _ takes precedent even when it comes to an exact match. Be very careful when using _ for it could easily dominate many of your categories.

Example:

The apples cost * each.
The apples cost _ each.
The apples cost a dollar twenty five each.

~The apples cost _ each.~ would override the other two patterns.

SRAI:

The srai tag is very useful in the situation when you want to redirect different sets of questions that are similar in meaning to one answer in the above category. For a clear example just how this works look at the script below.

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
<category>
<pattern> WHAT IS SCRIPTEDINTELLIGENCE *? </pattern>
<template>
<random>
<li> ScriptedIntelligence.com is a website about artificial intelligence, chatbots, and home of the S.I. Chat Bot League</li>
<li> A website that is home to the world's first chatbot league</li>
<li> <a href="http://www.ScriptedIntelligence.com " target="_blank">ScriptedIntelligence.com</a> website that is about artificial intelligence, chatbots and home of the S.I. Chat Bot League</li>
</random>
</template>
</category>
<category>
<pattern>WHERE CAN I FIND A CHAT BOT COMPETITION?</pattern>
<template><srai>WHAT IS SCRIPTEDINTELLIGENCE<star/></srai></template>
</category>
</aiml>
```

When a user now asks "where can I find a chat bot competition?" the template will be redirected to the above pattern "WHAT IS SCRIPTEDINTELLIGENCE *?" **NOTE:** the wild card character * must be replaced with <star/> tag within srai tags. Meaning that the two questions or more if you will add another srai tag are synonymous rephrased questions, and therefore the chatbot has only one template.

The answer will be randomly given by a chatbot one at a time as the identical pattern of question is being repeated by the user.

Using Variables:

You will find that variables are great and can be extremely useful, here is an example of using variables.

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
  <category>
        <pattern>OUR STORES</pattern>
        <template>
        <think>
        <set name="store1">Female Cloths</set>
        <set name="store2">Male Cloths</set>
        </think>
        </template>
</category>

<category>
        <pattern>GET *</pattern>
        <template>
        Sorry, that store is empty or non-existent.
        </template>
</category>

<category>
        <pattern>GET 1</pattern>
        <template>
                Store = <get name="store1"/>
        </template>
</category>

<category>
        <pattern>GET 2</pattern>
        <template>
                Store = <get name="store2"/>
        </template>
</category>
```

```
<category>
        <pattern>WHAT IS IN STORE *</pattern>
        <template>
        <srai>get<star/></srai>
        </template>
</category>
</aiml>
```

This will give you:
Input: What is in store 1?
Output: Female Clothes
Input: What is in store 2?
Output: Male Clothes
Input: What is in store 3?
Output: Sorry, that store is empty or non-existent.

Further education:

There are still many AIML tags that can be utilized in making an AIML file. Don't forget to use AIML variables which are really fascinating and can be used for many constructive uses. The following links will serve as good resources to those who want to delve more into AIML and Chatbots.

Alicebot.org
Chatbots.org
Aidreams.co.uk
knytetrypper.proboards.com
ScriptedIntelligence.com