ACME Script Widgets'

Mother's Little Helper

Mother's Little Helper (MLH) is a tool for AppleScript CGI writers which makes AppleScript CGI scripts much neater and easier to use. It acts as a filter between WebStar and the AppleScript applet combining all the parameters passed to the CGI into a single record, and parsing the post_args into a list of {field, value} pairs. Additionally it provides terminology for AppleScript which allows your CGIs to finally read more like AppleScript.

Mother's Little Helper (MLH) catches all incoming events to the applet of type "WWWΩ" and ID "sdoc", and bundles the parameters to the event into a single AppleScript record which can then be easily passed from one handler to another. Your AppleScript CGIs need never again contain code which looks like:

```
on «event WWWΩsdoc» path_args given «class post»:post_args ... etc.
```

Instead you define a handler for the WebStar event which looks like this:

```
on WWWEvent wParms
 ...
end WWWEvent
```

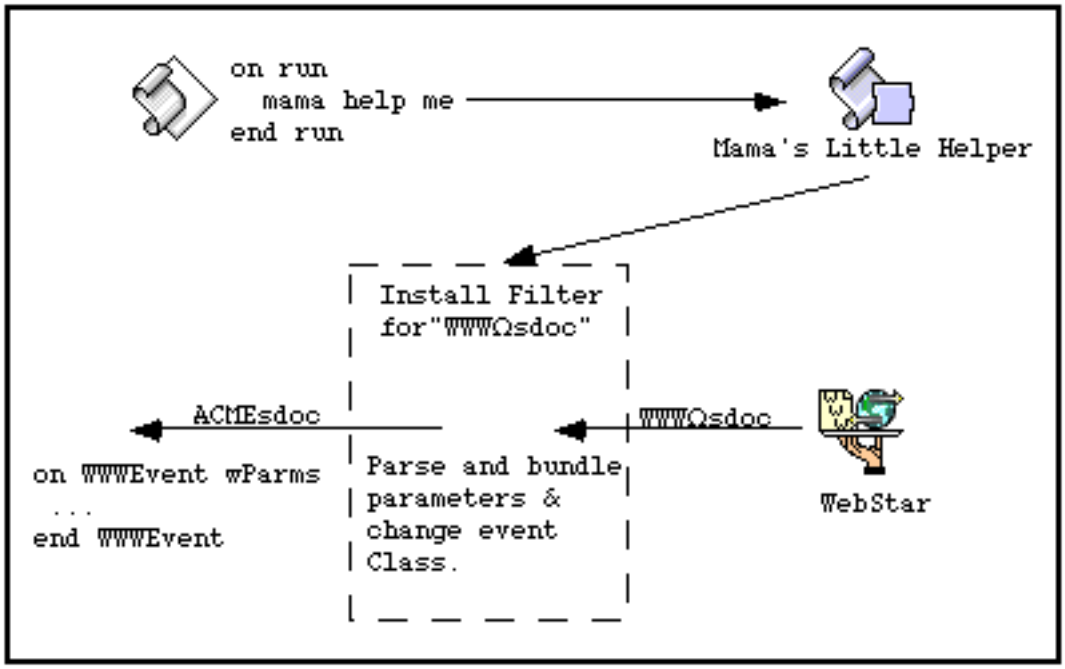
wParms now contains a record with the following properties:

Class cgi_params:

```
parsed_args  a list of lists  -- the parsed post arguments
path_args  string   -- path_args
http_search_args  string   -- http_search_args
post_args  string  -- post_args
method  string  -- method
client_address  string  -- client_address
username  string  -- uesr_name
password  string  -- password
from_user  string  -- from_user
server_name  string  -- server_name
server_port  string  -- server_port
script_name  string  -- script_name
```

```
content_type  string  -- content_type
referer  string  -- referer
user_agent  string  -- user_agent
action  string  -- action
action_path  string  -- action_path
client_ip  string  -- client_ip
full_request  string  -- full_request
connection_id  integer  -- thread_id
```

Conceptually, the process resembles that depicted in the drawing below.



The applet starts up and calls MLH to install the filter.

When the applet receives events from WebStar, they pass through the filter and get converted into an event for the on WWWEvent handler (**note:** WWWEvent is not the class and ID of the event; the event's class is 'ACME' and the ID is 'sdoc'). The arguments from the incoming event are bundled into a record of class cgi_params (defined in the 'aete' in the osax file), and the post_args parameter is parsed and added to the cgi_params as another property (parsed_args). The parsing done on post_args returns a format exactly like that of the ACME Parse Args Widget.

Thus, before the CGI's handler ever receives the event, all pre-processing has been done for it.

Technical Information

The osax file contains the MLH osax code resource, and a terminology resource (‘aete’) which not only provides an english language equivalent for the new event named WWWEvent, but also provides terminology for the parameters passed from WebStar. This means that if you open up one of your CGI/ACGIs written with the old style handler, all the parameters will now have intelligent names. In fact, what you may see is the following:

```
on «event WWWΩsdoc» path_args ¬
      given http_search_args:|http_search_args|, post_args:|post_args|,
... etc.
```

The vertical bars border the variable names because they have the same name as a keyword known to AppleScript. This is normal. There is however a side-effect from this behavior: if you have the MLH osax installed you’ll either have to refer to your variables (in old style CGIs) with the vertical bars around them, or change the variable names. In other words, to use the http_search_args variable in your old scripts, you’ll need to reference it as |http_search_args|, or else change the name to something else (like httpSearchArgs). Or, remove the MLH osax.

You only need to call MLH a single time when the applet runs, and the best place to do that is in the *run* handler. In AppleScript, if you do not declare a run handler with “on run ... end run,” all the statements outside of any other handlers are executed as if they were inside a run handler. Calling MLH twice will not hurt anything, but it will generate an error, so try not to do it.

MLH **ONLY** works when **called from within an AppleScript applet!** Make certain you understand that. It doesn’t work when you are running your script within an AppleScript development environment. If you call it from within Script Editor or other development environment, it will generate an error, but otherwise won’t hurt anything.

Up And Running...

Here are the steps you need to follow to write a CGI/ACGI using Mother's Little Helper.

1) Declare a handler with the following syntax (or any allowable variation which works for you:

```
on WWWEvent wParms
 ...
end WWWEvent
```

This is the new entry point to your CGI.

2) Inside this handler, refer to any of the parameters passed from WebStar as properties of the variable **wParms**. For example:

```
post_args of wParms
parsed_args of wParms
method of wParms
etc.
```

[wParms is just a variable and not a reserved word, so you can name the variable anything you want: foo, henry, muffy, etc.]

3) Don't make any calls to Parse CGI, ACME Parse Args, or the old Tokenize/DePlus/DecodeURL troika. This is no longer necessary since the arguments will already be parsed for you. They are in the **parsed_args** property of the wParms variable.

4) Add the following line to your run handler, or outside of any other handlers:

```
mama help me
```

(*She will if you ask her, but only then.*)

5) Remove the old "on «event WWWΩsdoc» path_args given..." handler from your script. Strictly speaking, you don't need to, but the whole point of MLH is to clean up and simplify the CGI code. If you have both handlers in your applet, the one which will get called depends on whether you ask *your mama to help* you or not. If you call MLH at any time while the applet is running, the old style handler will no longer field the events since the filter will have been installed.

6) Have fun.

Things to Remember...

For people not familiar with ACME Parse Args or the format of the parsed poop coming out of it, it works in the following manner.

It creates a list of lists, where each sublist contains the form field label name, and the value returned for that field (if any). If the same field name is defined multiple times on the html form, all values for that field will be combined into a list. So given the following contrived html form:

```
<HTML>
<head>
<TITLE>Sample Register Form</TITLE>
</head>
<BODY>
<form method=POST action="ACME_Parse_Args.acgi">

Your Name: <INPUT NAME="Name" SIZE="40"><P>
Other Name: <INPUT NAME="Name" SIZE="40"><P>

E-mail Address: <INPUT NAME="email"> <P>

</BODY>
</HTML>
```

The parsed arguments (with the user's values) would be:

{ {"Name", {"Wayne", "Walrath"}} , {"email", "ACME@kagi.com"} }

You get a list of two lists with the results of parsing the form above which contained three input fields. Since two of them had the same name, "Name", the values entered by the user are combined together in a list ({"Wayne", "Walrath"}). You can either loop through all the items in the main list, or use ACME Lookup Field (for sale as part of the ACME Script Widgets package) to quickly get the value of any field in the list.

Legal Stuff

Comments and bug reports should be sent to ACME@kagi.com. Please include enough information and sample scripts, data, etc. for me to reproduce the problem.