SOW Overview
Revision: 1.1.1.1

Mark J. Olesen

Date: 2004/12/22 01:35:40

Abstract

SOW is an experimental project, which is loosely derived from the YAPIHO project concept. It's purpose is to simplify the process of embedding HTML code within a CGI program such as PHP. As an example, consider a novice HTML programmer such as a family member who regularly maintains simple HTML pages. However, they now desire to have dynamic pages, and they do not wish to use or learn PHP, XML, etc. So, as a favor, you decide to take on the project in PHP. Later, they use their favorite editor and are either unable to modify the HTML because it's now in some CGI language such as PHP. To complicate matters, the HTML was broken up into chunks such as headers, body, footers, and the parts reassembled in the script. By implementing the pages in CGI, you have made them dependent upon you or other experienced developers to maintain their site.

SOW attempts to resolve these simple issues by allowing the developer to separate CGI code and HTML. The way it works is similar to the way a mail-merge program works–embedded fields within a document are translated and filled in with dynamic values.

Copyright

1 Revision History

```
$
$Log: overview.tex,v $
Revision 1.1.1.1  2004/12/22 01:35:40  markjolesen
initial import.

$
```

2 Contact

SOW project is proudly hosted on SourceForge.NET. The project site can be found at URL http://sow2.sourceforge.net and the project page at http://sourceforge.net/projects/sow2/.

Report Writing Services, INC. can be located at URL http://ReportWritingServices.com.

3 Audience

SOW and it's accompanying documentation is intended for it's community of users, which are mainly developers.

4 Introduction

SOW is an experimental project, which means that it is not suitable for use in commercial products or products that depend upon reliability. It is likely to change often and is entirely driven on it's community of users and developers.

Creating simple HTML based web pages has become a fairly easy task. It is not uncommon for the hobbyist of any age to create a web site comprised entirely of HTML based web pages. However, creating dynamic web pages introduces certain complexities–it requires knowledge of a language capable of running under the Common Gateway Interface (CGI).

The purpose of SOW is to separate embedded HTML code within the scripting language. It accomplishes this by using the concepts of mail-merge programs. To be certain, SOW is not a language and it is not an XML replacement. There are other viable and efficient ways to produce dynamic HTML code.

5 Overview

SOW is nothing more than a library of routines. It is intended to be simple yet powerful. Let SOW handle the details, and let the scripting language control the flow. SOW should be able to handle any type and size of data limited only by the amount of system resources.

A SOW document is comprised of sections and fields, which is described in the sections below.

5.1 Sections

A section is embedded within HTML comments such as

⟨!– @@ .section(mysection) @@ –⟩

The start of a document until the first named section is given a default section name "START." Section names, must be unique. Since "START" is already taken, it may not be used. A section continues until another section is encountered (see diagram below). There is no imposed limit on the number of sections that can be used. Section names may only contain alphanumeric ASCII characters and underscores.

```
          /-   <head>
START     |    <body>
          |    ....
          \-
          /-   <!--@@ .section(mysection) @@ -->
MYSECTION |    ...
          \-
```

5.2 Fields

Fields are place holders for dynamic values. They may appear anywhere within a section. However, fields defined in an HTML comment defining a section are currently ignored. Fields are surrounded by double hash/pound symbols (e.g. ##myfield##). Field names may only contain alphanumeric ASCII characters and underscores.

6 Miscellaneous

Section and field names are limited to 64 characters in length. They must be in ASCII and only contain alphanumeric characters and underscores. Section and field names are case-insensitive, and must not be duplicated (unless specified in the same comment block).

The first section in an HTML comment block is the one used. Any remaining will be ignored. For example, in the comment block

⟨!– @@.section(mysection)@@ @@.section(mysection2)@@ –⟩

"mysection2" will be ignored.

The parser tries to be forgiving and flexible. Typically, white-space is not a problem. For example, the comment block

⟨!– start of section @@ . section (mysection) @@ –⟩

would not produce any problems.

An important remark is to keep in mind that SOW does not format any information.

7 Issues

HTML editors can cause embedding issues. This is especially true for tags that do not require quotes. For example, the "option" tag

⟨option value="##value##" ##selected##⟩##text##⟨/option⟩

may contain an unquoted "selected" attribute. An editor such as Mozilla Composer may reformat the tag such as

⟨option value="##value##" ##selected##=""⟩##text##⟨/option⟩

A resolution would be to use a single field for the entire option block (e.g. ##option##). The disadvantage would be that the user would not see the selection box in an editor.

A PHP Examples

A.1 Text

A.1.1 message.txt

Following is an example of an ASCII text file named message.txt.

```
##header##

##date##

Dear ##salute##

Your account ##account## is past due by ##past_due_days##.
To avoid further harassment, pay the full amount
of ##amount_due## now dammit!
```

A.1.2 message.php

Following is an example PHP script that uses SOW to read in the message.txt template and fill in the fields.

```
<?php
/* message.php example */
/* read, fill and output message.txt */

include 'dl_local.php';

dl_local("phpsowDL.so");

$ar= array();
$ar['header']= "Cruel Collection agency\nYou owe us, pay now!\n";
$ar['date']= 'December 01, 1938';
$ar['salute']= 'Mr. Right';
$ar['account']= '01010101';
$ar['past_due_days']= 'many many years';
$ar['amount_due']= '$0.01';

$env= new_env();
```

```
update_env($env,$ar);

$sow = new_sow('message.txt');
stroke_sow($sow, $env);
reap_sow($sow);
?>
```

A.1.3 Output

Following is the output SOW produces.

```
Cruel Collection agency
You owe us, pay now!

December 01, 1938

Dear Mr. Right

Your account 01010101 is past due by many many years.
To avoid further harassment, pay the full amount
of $0.01 now dammit!
```

To give the example a little more utility, you can extract the contents of the document into a string, and then use the PHP mail() routine to E-mail it to someone.

```
...
// reap_sow($sow); do not send to standard out

// read contents of document
$str= duplicate_buffer_sow($sow);

// use mail to send it off
mail(...);
```

A.2 HTML

A.2.1 sample.htm

Following is an example of an HTML file named sample.htm.

```
<html>
<head>
<meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
<title>SAMPLE.HTM</title>
</head>
<body>
This is a simple SOW HTML example that fills in option values.
<br>
Select a state:
<select name="states">
<!-- @@.section(options1)@@ -->
```

```
<option value="##value##" ##selected##>##text##</option>
<!-- @@.section(rest)@@ -->
</select>
Remainder...
__o0o__
</body>
</html>
```

A.2.2 sample.php

Following is an example PHP script that uses SOW to read in the sample.htm template and fill in the fields.

```
<?php
/* sample.php */
/* read, fill in and output sample.php */

include 'dl_local.php';

dl_local("phpsowDL.so");

$env= new_env();
$sow = new_sow('sample.htm');

// stroke start section
cur_stroke_sow($sow, $env);

// advance cursor to next section "option"
cur_next_sow($sow);

$st= array();
$st[]= "NY";
$st[]= "TX";
$st[]= "CA";
$st[]= "FL";

$ar= array();

foreach ($st as $key=>$v){
$ar['value']= "$key";
$ar['selected']= ($key) ? "" : "selected";
$ar['text']= "$v";
update_env($env, $ar);
cur_stroke_sow($sow, $env);
}

// advance cursor to next section "rest"
// (prevent stroke of current section in butt)
cur_next_sow($sow);
```

```
// auto-stroke through remaining sections
cur_butt_sow($sow, $env);

// output document
reap_sow($sow);
?>
```

A.2.3 Output

```
<html>
<head>
<meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
<title>SAMPLE.HTM</title>
</head>
<body>
This is a simple SOW HTML example that fills in option values.
<br>
Select a State:
<select name="states">

<option value="0" selected>NY</option>

<option value="1" ="">TX</option>

<option value="2" ="">CA</option>

<option value="3" ="">FL</option>

</select>
Remainder...
__oOo__
</body>
</html>
```

B dl_local

```
<?php
// stolen from from php.net messages
// endofyourself@yahoo.com 18-Oct-2003 04:12
function dl_local( $extensionFile ) {
    //make sure that we are ABLE to load libraries
    if( !(bool)ini_get( "enable_dl" ) || (bool)ini_get( "safe_mode" ) ) {
      die( "dh_local(): Loading extensions is not permitted.\n" );
    }

      //check to make sure the file exists
    if( !file_exists( $extensionFile ) ) {
      die( "dl_local(): File '$extensionFile' does not exist.\n" );
    }
    //check the file permissions
    if( !is_executable( $extensionFile ) ) {
      die( "dl_local(): File '$extensionFile' is not executable.\n" );
    }

  //we figure out the path
  $currentDir = getcwd() . "/";
  $currentExtPath = ini_get( "extension_dir" );
  $subDirs = preg_match_all( "/\//" , $currentExtPath , $matches );
  unset( $matches );

      //lets make sure we extracted a valid extension path
    if( !(bool)$subDirs ) {
      die( "dl_local(): Could not determine a valid extension path [extension_dir].\n" );
    }

  $extPathLastChar = strlen( $currentExtPath ) - 1;

    if( $extPathLastChar == strrpos( $currentExtPath , "/" ) ) {
      $subDirs--;
    }

  $backDirStr = "";
      for( $i = 1; $i <= $subDirs; $i++ ) {
      $backDirStr .= "..";
        if( $i != $subDirs ) {
          $backDirStr .= "/";
        }
    }

  //construct the final path to load
  $finalExtPath = $backDirStr . $currentDir . $extensionFile;

    //now we execute dl() to actually load the module
      if( !dl( $finalExtPath ) ) {
      die();
    }

  //if the module was loaded correctly, we must bow grab the module name
  $loadedExtensions = get_loaded_extensions();
  $thisExtName = $loadedExtensions[ sizeof( $loadedExtensions ) - 1 ];

  //lastly, we return the extension name
   return $thisExtName;
 }//end dl_local()
?>
```