Simple Template System in PHP

Manuel A. Perez-Quinones, Computer Science, VT, 2012

Updated: May 12, 2013

The tiny template is a very simple template language implemented in PHP to be used for small projects and classroom instruction. The language supports conditionals, repetitions, and calls to functions embedded in HTML.

Variables

Variables can be placed anywhere in the HTML by using the name of a variable and encapsulating it in curly braces. {name} for example, will replace the value of the name variable for its value.

Passing Variables

Variables are passed in an associate array. The index (key) is the name of the variable and the value associated with that key will be used in the substitution. A simple example follows:

Consider an HTML file, ex1.tmpl, that includes the following:

```
<html>
<body>
<h1>Welcome</h1>
<p>My name is {name} and this is home page.</p>
</body>
</html>
```

To use this template, we prepare an array with the following values.

```
$symbols = array('name' => "Joe");
```

Then we call the generate_template routine as shown below.

```
require_once("template.php");
$page = gen_template("ex1.tmpl", $symbols);
echo $page
```

Directives

The directives for the template are embedded in <% symbols %> and include if and repeat. The if statement, seen below, allows a condition to be expressed. The condition is a variable passed to the system as shown above. If that variable evaluates to True, then the text for the "then-block" will be included in the output. If the variable evaluates to False, then the "else-block" will be included in the output. Note that the else block is option.

```
<% if {condition} %>
    html
<% else %>
    more html
<% end %>
```

The repetition directive, shown below, is similar to the if in that it uses a variable to contribute the execution. In this case, however, the variable is a collection of data. The body of the repeat will be included in the ouptput once for each element of the collection.

```
<% repeat {collection} %>

    html

<% end %>
```

Lets consider a more complex example to see how repeat works. Consider an HTML file, ex2.tmpl, that includes the following:

```
<html>

<body>

<h1>Welcome</h1>

<p>My name is {name} and this is home page.</p>

<ul>

<% repeat {phones} %>

    <li>my {type} number is {number}</li>

<% end %>

</ul>

</body>

</html>
```

Using collections in repeat are a bit tricky. At the top level of your variables, you want to have the symbol used in the repeat line. In this example it is {phones}.

```
$symbols = array('name' => "Joe",
    'phones' => array( .. ));
```

The content of the collection, should be arrays themselves using the symbols mentioned in the repeated part of the body of the repeat statement. In this example, each element of the 'phones' collection should have 'type' and 'number'. Then completing the example, we have:

```
$symbols = array('name' => "Joe",
    'phones' => array(
        array('type' => "home", 'number' => "555-1234"),
        array('type' => "cellular", 'number' => "555-2345")
        ));
```

Additional variables

The repeat statement also supports additional variables that can be used in the loop body.

- 'loopfirst' evaluates to True in the first iteration of the loop. It is false the rest of the time. Perfect to generate output on the first time through the loop.
- 'looplast' is similar to 'loopfirst' but evaluates to True only on the last iteration through the loop.
- 'loopcount' contains the index of the loop counter
- 'loopodd' evaluates to true when 'loopcount' is an odd number
- 'loopeven' evaluates to true when 'loopcount' is an even number

With these, we could do a more complex example as shown below:

```
<% repeat {phones} %>

    <% if {loopfirst} %>

        <ul>

    <% end %>

    <li>my {type} number is {number}</li>

    <% if {looplast} %>

        </ul>

    <% end %>

<% end %>
```

Future changes (these have been implemented, better doc coming soon)

- add calling a function so that it returns HTML directly
- The template system should support an all-in-one file representation.

    ```
    <% data {json | xml | csv} %>
    _data goes here_
    <% end %>
    ```

- how is the data from statement above mixed with other data?
- add special case to repeat to support a csv array with the first row having the field names and the individual rows not being in an associative array.

```
<% repeat {} %>
<% end %>
```

- add support to template.php to take the file name as both command line argument (argv) and CGI.

To be done

- add support for file inclusion
- add calling a function so that it returns data that is integrated with the other data.

the end