

FormRequest: Dynamic Form Parameters

The Problem

Sometimes you want a particular form on a page to return different parameters or values depending on which button or javascript function is used to submit the form.

Here is a simple example:

```
<input type='submit' name='action' value='Stop' />
```

```
<input type='submit' name='action' value='Start' />
```

Whichever submit button is pressed is considered "active", and only its parameter and value are passed in the request ("action=Stop" or "action=Start").

If you want a graphic button instead of a text button, the example changes to something like this:

```
<button type='submit' name='action' value='Stop'>
```

```
<img src='stop.png' height='40' width='40' alt='Stop' /></button>
```

```
<button type='submit' name='action' value='Start'>
```

```
<img src='start.png' height='40' width='40' alt='Start' /></button>
```

This should work the same way as the textual submit buttons, and in some browsers it does. Unfortunately, some versions/document compatibility modes of Internet Explorer use the innerText of the button as its value, while IE6 considers all graphical submit buttons to be active, not just the one that was pressed (if any). Either way, it is likely that you will end up with an inappropriate request to the server.

Some Solutions

Where the use of javascript is acceptable, one work-around is to omit the name attribute from the button so that no value is directly passed in the request and to use a click handler to populate a hidden input according to which button has been pressed, as follows:

```
<input type='hidden' name='action' value="" />
<button type='submit' onclick='this.form.action.value="Stop"'>
<img src='stop.png' height='40' width='40' alt='Stop' /></button>
<button type='submit' onclick='this.form.action.value="Start"'>
<img src='start.png' height='40' width='40' alt='Start' /></button>
```

A side effect of this approach is that there will now always be at least an empty value for "action", even if the form is submitted by some other means (e.g. javascript code or another submit button not using the variable "action").

In a very complex form, there could be quite a few "conditional" parameters. Of course, it is possible to create hidden inputs in advance for every possible value you might wish to submit with the form, along with special values, additional parameters, or some other context for server-side logic to determine which values should actually be used.

A slightly more sophisticated approach (also using and dependent on javascript) is to dynamically add hidden inputs to the document object model (DOM) for exactly the values you need for a particular request.

The "formRequest" function created by Brian Katzung of Kappa Computer Solutions, LLC makes this easy and convenient.

Continuing with the example from above, the code would look something like this:

```
<script type='text/javascript' src='formrequest.js'></script>
...
<button type='button' onclick='formRequest(this, { "action": "Stop" })'>
<img src='stop.png' height='40' width='40' alt='Stop' /></button>
<button type='button' onclick='formRequest(this, { "action": "Start" })'>
<img src='start.png' height='40' width='40' alt='Start' /></button>
```

In this example, the call to `formRequest` will dynamically add a hidden input with name "action" and a value of "Start" or "Stop", depending on which button is pressed. Note that the button type is now "button", not "submit", as `formRequest` will handle submitting the form. Like the original example, the "action" input will not exist in the request unless one of these buttons is used to submit the form or it is created elsewhere by other means.

FormRequest Details

If the first parameter is either a DOM form element or any element within the form that has a "form" property referencing the enclosing form, that form will be the recipient of the dynamically added hidden inputs. Otherwise, the first form on the page (`document.forms[0]`) is used.

Subsequent parameters should be objects whose attributes and values will be used to dynamically generate the hidden inputs. If the object is an array, the elements in the array are interpreted alternately as names and values instead.

If you need to pass multiple values for the same parameter, you can use an array of values or duplicate the key value (in an array object) as follows:

```
{ "name": [ "value1", "value2" ] }
```

```
[ "name", [ "value1", "value2" ] ]
```

```
[ "name", "value1", "name", "value2" ]
```

Special names "`=action`", "`=enctype`", "`=method`", and "`=target`" can be used to override the action URL, enctype (encoding type), method (GET or POST), or target window of the form for specific calls to `formRequest`.

If the target of the form is another window (either because of a target attribute in the original request or because of an "`=target`" setting in a `formRequest`), or if the form generates an AJAX request, the page will not be replaced and it is therefore possible for the form to be submitted multiple times as the result of additional user action. Subsequent calls to `formRequest` will undo the previous changes before processing the new `formRequest`. If you need to undo the changes in any other context, simply call `undoFormRequest()`.

One final special name, "`=confirm`", prompts the user (using the value as the prompt) to confirm or cancel the form submission.

None of the special names or their values are passed in the request.