Using Coldfusion to generate JQuery validation scripts

Posted At : 17 January 2010 23:08 | Posted By : Shaun McCran
Related Categories: JQuery, Coldfusion, Frameworks, Javascript, AJAX

One of the ideas that we have been throwing around the office is creating a platform that will create the form-validation-database cycle automatically. Something akin to ORM, but not. ORM looks good, but we want more control over what is happening.

In this article I am exploring the idea of automatically creating JQuery validation from a simple Coldfusion input. In this case a list of required fields. I'll say up front Ray Camden's blog entry on Jquery Validation (http://www.coldfusionjedi.com/index.cfm/2009/2/10/An-Introduction-to-jQuery-and-Form-Validation-2) has been an invaluable help.

The principle behind this is that you can create a generic validation object routine, and simple provide it with a set data object (list or struct, haven't decided yet) and have it match against a form and validate it. So with that in mind we will create a simple form.

```
<form name="form" id="form" method="post" action="index.cfm?inline=#url.inline#">

                    <label for="name">Name</label><br/>
      <input type="text" name="name" id="name" class="form-field"><p/>

                <label for="telephone">Telephone</label><br/>
  <input type="text" name="telephone" id="telephone" class="form-field"><p/>

                  <label for="email">Email</label><br/>
      <input type="text" name="email" id="email" class="form-field"><p/>

       <label for="favouriteSandwhich">Favourite Sandwhich</label><br/>
t type="text" name="favouriteSandwhich" id="favouriteSandwhich" class="form-field"

            <input type="submit" name="action" value="Submit">

                                  </form>
```

As you can see from above, this is a simple form. Next we will include references to the Google code repository for the AJAX library, and out JQuery validation plugin.

```
'text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.mi
   <scr/ipt type="text/javascript" src="jquery.validate.pack.js"></script>
```

Next we have the JQuery script to validate our fields. Firstly we create a list of the fields that are required. By doing this it is incredibly easy to add another required field to the model.

```
<cfset variables.requiredList = "name,email">
```

The next phase of this will probably accept a structure rather than a list, in that way I can combine field lengths and any other custom criteria, rather than merely the fact that it is required. The JQuery validation starts off in exactly the same way as usual, we create a function to catch the validation request, and use the id of the form as the reference. We declare the error container style of "#error". Next we loop over our list inside the "rules:" element. This creates the required rules for each field. I've also added a basic check for the value 'email'. This will create an email validation rule. Next we create custom messages in the "message:" element. Again this catches the email value and adds an appropriate message.

```
                                    <scri/pt>
                                  $(document).ready(function(){
                                         $("#form").validate({

                                      errorContainer: "#error",
                                    errorLabelContainer: "#error ul",
                                           wrapper: "li",

                                              rules: {
                                                 <cfoutput>
                                <cfloop list="#variables.requiredList#" index="variab
#variables.index#: {required: true <cfif findNoCase('email', variables.index, '1'
                                                       </cfloop>
                                                    </cfoutput>
                                                            },

                                             messages: {

                                                    <cfoutput>
                                <cfloop list="#variables.requiredList#" index="variab
                             #variables.index#: {required: "The #variables.index# fiel
oCase('email', variables.index, '1')> email: "Email addresses are of the form user
                         minlength: jQuery.format("You need to use at least {0} charac
                                                            },

                                                    </cfloop>
                                                 </cfoutput>
```

```
                                                                    }
                                                                        }
                                                                );
                                                            });
                                                        </script>
```

Lastly we can create a style for the #error container we declared above.

```
                                        <style>
                                    /* Error handling styles */
                                #error {-moz-background-clip:border;
                                     -moz-background-inline-policy:continuous;
                                          -moz-background-origin:padding;
                                               background:#FFE7DF;
                                         background-position: 5px 8px;
                                           border: #FF3F00 solid 1px;
                                                color:#440000;
                                              margin:10px 0 1em;
                                            padding:0px 7px 7px 7px;
                                                display:none;
                                                 width: 90%;}

                                    /* padding for the list */
                    #error ul {list-style-type:none; padding: 0px 0px 0px 0px;}

                                /* padding for the list items */
                            #error ul li {padding: 4px 0 2px 16px;}

l; list-style-type:none; padding: 0px 0px 0px 0px; width: 198px; color:#440000; ba
                                    li {list-style-type:none;}
                        .form-field {width: 200px; padding: 0px 0px 0px 0px;}
                                        </style>
```

This creates a totally dynamic validation routine - all fed from a list. I think it forms a good basis to build a more dynamic rules driven model, where you can set field lengths as well.

There is an example of the complete code here, along with a variation on the inline or external placing of the validation messages.

Follow this link for a demo of the JQuery validation model .