Records / Manager Form

Posted At : June 29, 2010 9:30 AM | Posted By : Steve
Related Categories: com.sebtools, sebtags

Much of this entry has been updated as **Super Easy CRUD/File Management**.

I did a rough introduction to Records.cfc last time - just enough to show how it can make really basic CRUD operations easy. This time I want to show how you can take advantage of it to help keep your applications DRY.

In addition to the methods that I discussed last time, Records.cfc components (that is, components that extend Records.cfc) also get another method that I would like to discuss today:

- getFieldsStruct: This provides a structure of the fields in the primary table being managed by the component.

The structure returned is actually a structure of structures. Each key in the outer structure represents a field name. Each field has a key for every attribute of that field.

Let's put it to some use.

Last time we put together a very simple form to add a widget.

```
<form action="action.cfm" enctype="multipart/form-data" method="post">
   <div>
      <label for="WidgetName">Widget</label><br>
      <input type="text" name="WidgetName">
   </div>
   <div>
      <label for="WidgetImage">Image</label><br>
      <input type="file" name="WidgetImage">
   </div>
   <div>
      <input type="submit" value="Submit">
   </div>
</form>
```

Let's get some information so that we can make this form a bit easier to maintain.

```
<cfset sFields = Application.Widgets.getFieldsStruct()>
<cfoutput>
<form action="action.cfm" enctype="multipart/form-data" method="post">
   <div>
      <label for="WidgetName">
         #sFields.WidgetName.Label#
         <cfif StructKeyExists(sFields.WidgetName,"required") AND sFields.WidgetName.required IS true>
         *
         </cfif>
         :
      </label>
      <br>
      <input
         type="text"
         name="WidgetName"
         <cfif StructKeyExists(sFields.WidgetName,"Length")> maxlength="#sFields.WidgetName.Length#"</cfif>
         <cfif StructKeyExists(sFields.WidgetName,"size")> size="#sFields.WidgetName.size#"</cfif>
      >
```

```
   </div>
   <div>
      <label for="WidgetImage">
         #sFields.WidgetImage.Label#
         <cfif StructKeyExists(sFields.WidgetImage,"required") AND sFields.WidgetImage.required IS true>
         *
         </cfif>
         :
      </label>
      <br>
      <input type="file" name="WidgetImage">
   </div>
   <div>
      <input type="submit" value="Submit">
   </div>
</form>
</cfoutput>
```

The point of this isn't the actual code, but that you can determine information about each field from the component itself. You could check for a required field add any JavaScript and/or server side validation that you wanted as a result.

You could even add extra information in the form, like an "extensions" list and check for that in both JavaScript and ColdFusion code.

```
<cfset sFields = Application.Widgets.getFieldsStruct()>
<cfif StructKeyExists(Form,"WidgetImage")>

   <cfset folder = Application.Widgets.getFolder("WidgetImage")>
   <cfset dir = Application.FileMgr.getDirectory(folder)>
   <cffile action="UPLOAD" filefield="WidgetImage" destination="#dir#" nameconflict="overwrite">
   <cfset Form["WidgetImage"] = CFFILE.ServerFile>
   <cfif StructKeyExists(sFields.WidgetImage,"extensions") AND NOT
ListFindNoCase(sFields.WidgetImage.extensions,CFFILE.ServerFileExt)>
      <cffile action="delete" file="#dir##CFFILE.ServerFile#">
      <cfset StructDelete(Form,"WidgetImage")>
   </cfif>
</cfif>
<cfset Application.Widgets.saveWidget(argumentCollection=Form)>
<cflocation url="index.cfm" addtoken="no">
```

The big advantage here is that you can centralize the information about your data structure in the XML and have those changes available throught your application.

Personally, I love to use cf_sebForm with Records components. It will do all of the functionality demonstrated or discussed on this page with the following code:

```
<cf_sebForm
   CFC_Component="#Application.Widgets#"
   UploadFilePath="#ExpandPath('/files/')#"
   UploadBrowserPath="/files/"
   forward="index.cfm"
>
   <cf_sebField name="WidgetName">
   <cf_sebField name="WidgetImage">
   <cf_sebField type="Submit">
</cf_sebForm>
```

To clarify, that will create a form that has the appropriate labels and length limitations, checks for required fields on the client and server, verifies file type on the client and server, and handles the upload itself.

Records.cfc and Manager.cfc are part of the **com.sebtools package** which is open source and free for any use. The tags are part of **SebTags**, which are also open source and free for any use.