Adding Fields to StarterCart

Posted At : February 17, 2011 10:30 AM | Posted By : Steve
Related Categories: StarterCart

Every shopping cart that I have tried or reviewed has some generic fields in their products that are meant to handle any product data that wasn't otherwise covered by their existing fields. I never liked this approach. It makes the GUI user somehow responsible for data structure and probably doesn't eliminate the need for custom code anyway.

With **StarterCart**, I took a different approach. Instead of having several product fields as well as some generic fields, I have only the bare bones - fields for the name, price, description. I honestly debated about whether or not to include a description field.

So, how do you add fields?

I have **covered this a bit** in the **docs**, but I thought I would do it again here from only a slightly different perspective.

Let's add an image field and a thumbnail field.

Creating a Custom Starter Cart component.

The first thing that we need to do is make a customized version of StarterCart (expect to have to do this for every cart you install).

To do this, we will open up /admin/cart/model/ and create StarterCart_Example.cfc (the part after the "_" will indicate your site):

```
<cfcomponent extends="StarterCart">

</cfcomponent>
```

Next we will need to open up /_config/components.cfm and find the entry for StarterCart. It should look like this:

```
<component name="StarterCart" path="admin.cart.model.StarterCart">
   <argument name="Manager" />
   <argument name="cfpayment" />
   <argument name="NoticeMgr" ifmissing="skiparg" />
   <argument name="Scheduler" ifmissing="skiparg" />
   <argument name="ErrorEmail" arg="Cart_ErrorEmail" />
</component>
```

Change it to this (Adding the "_Example" to the path):

```
<component name="StarterCart" path="admin.cart.model.StarterCart_Example">
   <argument name="Manager" />
   <argument name="cfpayment" />
   <argument name="NoticeMgr" ifmissing="skiparg" />
   <argument name="Scheduler" ifmissing="skiparg" />
   <argument name="ErrorEmail" arg="Cart_ErrorEmail" />
</component>
```

Then **refresh the framework** so it will know about this change.

Now we have a customized version of StarterCart running on our site.

Adding Fields

Now we need to add the product fields to StarterCart by editing our StarterCart_Example.cfc file. Let's have a large image with a maximum size of 400X400 and a thumbnail of that with a maximum size of 80X80. Ideally, our user should only have to upload the full size image and then the thumbnail should be created from that.

```
<cfcomponent extends="StarterCart">

<cffunction name="customxml">
<tables>
   <table name="prodProducts">
      <field
         name="ProductImage"
         label="Image"
         type="image"
         folder="images"
         MaxWidth="400"
         MaxHeight="400"
      />
      <field
         name="ProductThumb"
         label="Thumbnail"
         type="thumb"
         folder="thumbs"
         original="ProductImage"
         MaxWidth="80"
         MaxHeight="80"
      />
   </table>
</tables>
</cffunction>

</cfcomponent>
```

Make sure to **refresh StarterCart** so that this change will take effect. The edit form will automatically reflect the change with no need to change the product-edit.cfm code at all (though you certainly can if you want).

In case you are curious what code makes the product-edit.cfm page so malleable, here it is:

```
<cf_PageController>

<cf_layout>
<cf_layout showTitle="true">

<cf_sebForm />

<cf_layout>
```

The **cf_sebForm** tag can take a **cf_sebField** tag for each field in the form. If it doesn't get any, however, it is able to figure out what **fields should be on the form**. It also figures out **what methods to call** on the component (the **Neptune** framework tells the form which component to use based on the file name - though you can **specify this** if you want).

Now the admin form will have a new field with a label of "Image" (the thumbnail field will not display by default). It will have client and server side validation to ensure that the uploaded file is an image.

It will re-size the image to fit in 400X400 and create a thumbnail from it that will fit in 80X80.

Using the fields

Let's say you are using the default product output for your products page (which you probably won't be, but it is an easy example):

```
<ul>
<cfoutput query="qProducts">
   <li><a href="product.cfm?id=#ProductID#">#ProductName#</a></li>
</cfoutput>
</ul>
```

Then you could change your output code to this to show the thumbnail on this page. Similar code could be used on the product detail page to show the full size image as well.

```
<ul>
<cfoutput query="qProducts">
   <li>
      <cfif Len(ProductThumb)>
         <a href="product.cfm?id=#ProductID#"><img src="#ProductThumbURL#" alt="#ProductName#"></a>
      <cfelse>
         <a href="product.cfm?id=#ProductID#">#ProductName#</a>
      </cfif>
   </li>
</cfoutput>
</ul>
```

Note that the "ProductThumbURL" field is used for the src. Any query returned with fields fields will have a "URL" and a "Path" version returned for the URL to the file and the server path to the file respectively.

Summing Up

This example used image fields, but you could just as easily as text fields or fields or select boxes. If there is interest in tutorials on any specific kind of field, I would be happy to write those up.

StarterCart is open source and free for any use. It can be downloaded from **RIAForge**.