09_05 Using the Application scope

In this exercise, you will re-write the code of your application so that the pages use the instances of the components that are cached in the Application scope instead of creating new instances of those components each time.

Update the remaining pages of the site

There are three more pages to update on the model that has been demonstrated in the previous video.

1. Open the '`siteMap.cfm`' page.
2. At the top of the code, delete the `<cfset />` statement that creates the '`pageService.cfc`' component instance.
3. Update the second `<cfset />` statement so that the call to the '`getPageByID()`' method uses the instance of the component stored in the '`application`' scope instead of the local instance.
4. Save and test the page. Confirm it works as expected before moving on to the next page.
5. Repeat that same procedure on the '`wePlayForYou.cfm`' page and on the '`contactUs.cfm`' page

The code of the '`siteMap.cfm`' page should be the following.

```
<!---Get page content for fld_pageID = 5--->
<cfset rsPage = application.pageService.getPageByID(5) />
```

Update the 'agenda.cfm' page.

This page is a bit more complex to update. Remember that it has 2 'states'. It can display either a list of future events or a single event.

1. Open the '`agenda.cfm`' page.
2. At the top of the code, locate the `<cfset />` tag that creates an instance of the `eventsService` component and delete it.
3. Locate the `<!---Output a single agenda if url.eventID is defined--->` comment.
4. Update the `<cfset />` that follows so it uses the `eventsService` instance that is in the `application` scope.

The code should be :

```
<!---Output a single agenda if url.eventID is defined--->
<cfset rsSingleEvent =
    application.eventsService.getEventByID(url.eventID) />
```

5. Locate the `<!---Output the upcoming event table if url.eventID not defined—>` comment.
6. Apply the same update to the `<cfset />` statement that follows.

The code should be :

```
<!---Output the upcoming event table if url.eventID not defined--->
<cfset rsCurrentEvents = application.eventsService.getCurrentEvents() />
```

7. Save and test the page.
8. Confirm that both 'states' work, by clicking on some '`Read More`' links next to the events.

If everything works as expected, close all the files in ColdFusion Builder before moving on to the next step.

Update the 'news.cfm' page.

This page also is a bit more complex. It has three 'states' : the *All news* state, the *Single news* state and the *News for year* state.

1. Open the '`news.cfm`' page in ColdFusion Builder.
2. At the top of the page, delete the `<cfset />` that instantiate the `newsService` component. Also delete the associated comment.
3. Update the `<cfset />` statement that follows the `<!---Get news years--->` comment so it uses the instance of the newsService component that is in the application scope.

The code should be :

```
<!---Get news years--->
<cfset rsNewsYears = application.newsService.getNewsYears() />
```

4. Apply the same update at the following places
   - just after the `<!---Output a single news—>` comment.
   - Just after the `<!---Get all news—>` comment.
   - Just after the `<cfelseif isDefined('url.year') >` line of code.

Save and test all three states of the page. Confirm it works as expected before moving on to the next step.

Update the ‘comePlayWithUs.cfm’ page.

This page contains a form as well as the form processing code. Even though it may sound a bit more complex, the story is the same as for the previous pages.

1. Open the ‘`comePlayWithUs.cfm`’ page in ColdFusion Builder.
2. First, delete every piece of code that instantiate a component.

Beware of the fact that ‘`comePlayWithUs.cfm`’ instantiates two components : the ‘`userService.cfc`’ component and the ‘`pageService.cfc`’ component. Make sure you delete the instantiation of both components.

- The ‘`userService.cfc`’ component instantiation is at the top of the page.
- For the ‘`pageService.cfc`’ component, you need to scroll down the page a little bit and locate the `<!---Get page content … --->` comment.

3. Just below the `<!---Get page content for fld_pageID = 4 --->` comment, update the `<cfset />` so it uses the instance of the `pageService` component that is the application scope.

The code should be :

```
<!---Get page content for fld_pageID = 4--->
<cfset rsPage = application.pageService.getPageByID(4) />
```

In the form processing code, there are two `<cfset />` statements to modify.

4. The first one is the one that calls the ‘`validateUser()`’ method.
5. The second one is the one that calls the ‘`addUser()`’ method to insert the user in the database.
6. Finally, update the `<cfset />` that calls the ‘`getInstruments()`’ method, just below the `<!---Get the instruments list to fill the drop down menu of the form—>` comment.

When all updates are done, save and test the page. Submit a new user to make sure the form processing code still works. If the page works as it should, you can safely move on to the next step.

Update the ‘profile.cfm’ page.

The ‘`profile.cfm`’ page is the last one to update. If you feel confident enough, feel free to update it on your own without going through the steps explained below. If you still need some guidance, the next few paragraphs have been written just for you!

1. First, open the ‘`profile.cfm`’ page in ColdFusion Builder.
2. At the top of the code, delete the `<cfset />` tag that instantiate the ‘`userService.cfc`’ component as well as its associated comment.
3. In the form processing code, update the `<cfset />` tag that calls the ‘`validateUser()`’ method. Add the ‘`application.`’ prefix so it uses the instance of the ‘`userService`’ component that is in the application scope.
4. Apply the same update to the `<cfset />` that calls the ‘`updateUser`’ method.

The form processing script should be the following.

```
<!---Form processing begins here--->
<cfif structKeyExists(form,'fld_editUserSubmit')>
  <!---Server side form validation--->
  <cfset aErrorMessages =
      application.userService.validateUser(form.fld_userFirstName,f
      orm.fld_userLastName,form.fld_userEmail,form.fld_userPassword
      ,form.fld_userPasswordConfirm) />
  <!---Continue form processing if the aErrorMessages array is
      empty--->
  <cfif arrayIsEmpty(aErrorMessages)>
      <cfset
          application.userService.updateUser(form.fld_userFirstName
          ,form.fld_userLastName,form.fld_userEmail,form.fld_userPa
          ssword,form.fld_userRole,form.fld_userInstrument,form.fld
          _userComment,form.fld_userIsApproved,form.fld_userIsActiv
          e,form.fld_userID) />
      <cfset variables.formSubmitComplete = true />
  </cfif>
</cfif>
```

To display the form, there are two more `<cfset />` tags to update.

5. The first one calls the ‘`getUserByID()`’ method and is located just after the `<!---Get user to update --->` comment
6. And the second one calls the ‘`getInstruments()`’ method.

When all the updates are done, save and run the page. Modify some data in the form and submit it to make sure everything works fine!