12_06 The Users administration

You probably have a pretty clear idea of what's to come in this document. All four CRUD operations will be applied to the users of your site, the very same way it has been applied to the other sections of the administration site. There are, however, some extra ColdFusion features to discover.
Remember that any site visitor can fill out a form on the public site to mark his interest in joining the band. When a user fills and submit this form, he actually performs the 'Create' operation of CRUD. The administrator must then approve or reject the candidate musician. If the candidate gets approved, an e-mail is automatically sent to the user announcing the good news and providing his password.

- Sending e-mail with ColdFusion will be covered in this section.
- The error handling system of ColdFusion will also be introduced at the end of the document.

But for now, let's start the process by creating the required pages.

Create the necessary pages in the 'admin' folder.

Refer to the following table for details about the pages to create. Remember that each of these three pages has the following basic code.

```
<cf_admin title="title Goes Here">
  <div id="pageBody">
      <h1>main page title goes here</h1>
  </div>
</cf_admin>
```

Page Name	Title attribute	<h1> content
users.cfm	HD street band - Users administration	Users Administration
userEdit.cfm	HD street band - Users administration	Update a user
userAdd.cfm	HD street band - Users administration	Create a new user

Once done, test each of those three pages one by one. If you named your files correctly, the '`Users`' link on the main navigation bar of the admin site should now be working.

Add the necessary methods in the 'usersService.cfc' component

Open the '`usersService.cfc`' component. It currently contains 5 methods used to support the forms on the '`comePalyWithUs.cfm`' and on the '`profile.cfm`' pages.

Before adding the necessary methods to this component, there are two little changes to do on the existing methods.

1. Open the '`userService.cfc`' component in ColdFusion Builder.
2. Locate the '`getUserByID()`' method.
3. In the '`SELECT`' clause of the `<cfquery>` add the '`FLD_USERROLE`' field to the list of fields retrieved from the database.

That's it for the first change!

4. Locate the '`updateUser()`' method.
5. In the '`SET`' clause of the query, add three lines to update the '`FLD_USERROLE`', the '`FLD_USERAPPROVED`' and the '`FLD_USERISACTIVE`' fields of the database.

The final code of the '`UPDATE`' query is the following

```
<cfquery>
  UPDATE TBL_USERS
  SET
  FLD_USERFIRSTNAME = <cfqueryparam value="#arguments.userFirstName#"
    cfsqltype="cf_sql_varchar" />,
  FLD_USERLASTNAME = <cfqueryparam value="#arguments.userLastName#"
    cfsqltype="cf_sql_varchar" />,
  FLD_USEREMAIL = <cfqueryparam value="#arguments.userEmail#"
    cfsqltype="cf_sql_varchar" />,
  FLD_USERPASSWORD = <cfqueryparam value="#arguments.userPassword#"
    cfsqltype="cf_sql_varchar" />,
  FLD_USERINSTRUMENT = <cfqueryparam value="#arguments.userInstrument#"
    cfsqltype="cf_sql_integer" />,
  FLD_USERCOMMENT = <cfqueryparam value="#arguments.userComment#"
    cfsqltype="cf_sql_longvarchar" />,
  FLD_USERROLE = <cfqueryparam value="#arguments.userRole#"
    cfsqltype="cf_sql_integer" />,
  FLD_USERISACTIVE = <cfqueryparam value="#arguments.userIsActive#"
    cfsqltype="cf_sql_integer" />,
  FLD_USERAPPROVED = <cfqueryparam value="#arguments.userIsApproved#"
    cfsqltype="cf_sql_integer" />
  WHERE FLD_USERID = <cfqueryparam value="#arguments.userID#"
    cfsqltype="cf_sql_integer" />
</cfquery>
```

Six more methods are needed to implement the features of the Users Administration.

6. Open the '`helpers/usersComponent.txt`' file.
7. Copy/paste all the code in the '`usersService.cfc`' component after the existing five methods.
8. Save and run the component to access its automatic documentation.

The methods that have been added are the following:

- The '`getUsersToApprove()`' method does not take any argument and returns a query. It is used to display the list of users that submitted the form on the '`comePlayWithUs.cfm`' page, but that have not yet been approved.
- The '`approveUser()`' method takes the ID of the user to approve as its only argument and does not return any value. Notice that this method is not entirely finished.
- The '`getAllActiveUsers()`' method does not take any argument and also returns a query. It is used to display the list of approved and active users.
- The '`deleteUser()`' method takes the '`userID`' as its only argument and does not return any data. It is used to delete a user from the database.
- The '`deactivateUser()`' method takes one argument and does not return anything. In an upcoming video, you will learn that some users cannot be deleted from the database. In such case, this method is used as a plan B to deactivate the user instead of deleting it.
- The '`sendApproveMail()`' method is a private method. It takes the '`userID`' as its only argument and is used to send the confirmation e-mail to the user that has just been approved. As you can see in the code, this method has not been developed yet.

Carefully review the rest of the code of the component before moving on to the next step. You will finalize the '`approveUser()`' and the '`sendApproveMail()`' methods later in this document.

Refresh the 'usersService.cfc' component in the application scope.

To do so, browse any page of the site, add the '`?restartApp`' url parameter and reload the page. When the page reloads, confirm that the '`Execution Time`' section of the debug output has an entry for the '`onApplicationStart()`' method of the '`Application.cfc`' file.

Display the list of users to approve on the 'users.cfm' page.

In ColdFusion Builder, return to the '`users.cfm`' page.

1. At the very top of the page, use `<cfset />` to create the '`usersToApprove`' variable.
2. Make it equal to the query returned by the '`getUsersToApprove()`' method of the '`usersService.cfc`' component.

```
<!--- Get the list of users to approve --->
<cfset usersToApprove =
    application.userService.getUsersToApprove() />
```

3. On the line following the `<h1>` title, create a `<cfif> <cfelse>` block that checks if the '`usersToApprove`' query contains data.
4. If it does not, create a meaningful message in a `<p>` tag block.
5. In the `<cfelse>` part, create a meaningful `<h2>` title.
6. Just below the `<h2>`, create an HTML table of 2 rows by 6 columns.
7. Wrap the second row in a `<cfoutput>` tag block that loops over the '`usersToUpdate`' query.
8. In the first row, create the column headings. In the second row, output the data contained in the '`usersToApprove`' query. The headings are '`User Name`', '`eMail address'`, '`Comment`' and three '`&nbsp;`'.
9. Write '`View`', '`Approve`' and '`Delete`' in the last three cells of the table.
10. Save and run the page to test it out. Remember that you must be logged-in as an administrator to execute the methods and view the page.

The code is

```
<cfif usersToApprove.recordCount EQ 0>
    <p>There are no users to approve at this time.</p>
<cfelse>
    <h2>Users to approve</h2>
    <table>
      <tr>
          <th>User Name</th>
          <th>eMail Address</th>
          <th>Comment</th>
          <th>&nbsp;</th>
          <th>&nbsp;</th>
          <th>&nbsp;</th>
      </tr>
      <cfoutput query="usersToApprove">
          <tr>
```

```
                <td>#fld_userFirstName# #fld_userLastName#</td>
                <td>#fld_userEmail#</td>
                <td>#fld_userComment#</td>
                <td>View</td>
                <td>Approve</td>
                <td>Delete</td>
            </tr>
        </cfoutput>
    </table>
</cfif>
```

TIP : If there are no users to approve in your database, go to the public site and fill the form on the '`comePlayWithUs.cfm`' page.

Code the 'Approve' link.

1. Return to ColdFusion Builder and locate the '`Approve`' link in the fifth cell of the table.
2. Make it a link to the current page and pass the '`approve`' URL parameter.
3. Make that parameter equal to the ID number of the current user.

```
<td><a href="users.cfm?approve=#fld_userID#">Approve</a></td>
```

4. At the very top of the '`users.cfm`' page, use `<cfif>` and '`structKeyExists()`' to detect the '`approve`' parameter in the URL.
5. If the parameter is detected, use it to call the '`approveUser()`' method.

```
<!---Approve Users--->
<cfif structKeyExists(url,'approve')>
    <cfset application.userService.approveUser(url.approve) />
</cfif>
```

6. Scroll down the code and place your cursor after the `<h1>` title.
7. At that location, use `<cfif>` and '`structKeyExists()`' to check for the existence of the '`approve`' parameter in the URL.
8. If the parameter exists, create a meaningful feedback message to the user.

```
<cfif structKeyExists(url, 'approve')>
    <p class="feedback">The user has approved.</p>
</cfif>
```

If you open the '`userService.cfc`' component and take look at the '`approveUser()`' method, you shall see that this method uses an '`UPDATE`' query to change the value of the '`FLD_UserApproved`' and of the '`FLD_userIsActive`' fields of the '`TBL_USERS`'. This is an operation that has no more secrets for you!

But the method should also send an eMail to the new approved user. This is what we will cover in the next few videos.

Watch Video : Configuring ColdFusion to send eMails

Code the ‘view’ link.

Locate the ‘`View`’ link in the fourth cell of the table

1. Make it a link to the ‘`userEdit.cfm`’ page.
2. Pass the ‘`userID`’ variable in the URL.

```
<td><a href="userEdit.cfm?userID=#fld_userID#">View</a></td>
```

Save and run the page. Click on any ‘`View`’ link to check if the ‘`userEdit.cfm`’ page opens as expected and if the ‘`userID`’ parameter is correctly passed to the target page.

Prepare the ‘userEdit.cfm’ form.

Return to ColdFusion Builder and open the ‘`userEdit.cfm`’ page.

1. Open the ‘`helpers/usersForm.txt`’ file.
2. Copy/paste all the content of that file in the ‘`userEdit.cfm`’ page, right below the `<h1>` tag block.
3. At the top of the code, use `<cfset />` to create the ‘`userInstruments`’ variable.
4. Make it equal to the data returned by the ‘`getInstruments()`’ method of the ‘`userService.cfc`’ component.

```
<!---Get the instruments to generate the drop-down menu of the
    form--->
<cfset userInstruments =
    application.userService.getInstruments() />
```

5. Save and run the page to have a first look at the form in the browser.
6. Return the ColdFusion Builder and inspect the HTML code of the form. Pay particular attention to the names of the fields.

Pre-populate the form with the current data of the user.

In ColdFusion Builder, return to the 'userEdit.cfm' page.

1. At the very top of the code, use <cfif> and 'structKeyExists()' to check if the 'userID' url parameter is passed.
2. If not, redirect the administrator to the 'users.cfm' page

```
<!---Check if userID parameter is correctly passed--->
<cfif NOT structKeyExists(URL, 'userID')>
    <cflocation url="users.cfm" />
</cfif>
```

When done, save and run the page. Because no 'userID' parameter is present in the URL, you should be redirected to the 'users.cfm' page. Click on any 'View' link. The 'userEdit.cfm' page should now display as expected, but the form is still empty.

3. Return to the 'userEdit.cfm' page in ColdFusion Builder.
4. Below the <cfif> block you just wrote, use <cfset /> to create the 'userToEdit' variable.
5. Make it equal to the query returned by the 'getUserByID()' method of the 'userService.cfc' component.

```
<!---get the current data of the user to edit.--->
<cfset userToEdit =
    application.userService.getUserByID(url.userID) />
```

Locate the <!---Display form fields---> comment. Below that comment, various <cfinput /> tags are used to create the fields of the form. The first one is of type 'text' and creates the 'fld_userFirstName' field.

1. Use the 'value' attribute of the <cfinput /> tag to populate the starting value of the field and make it equal to '#userToEdit.fld_userFirstName#'.
2. Repeat the same operation for the remaining text fields of the form.
   - For 'fld_userLastName' use '#userToEdit.fld_userLastName#'.
   - For 'fld_userEmail' use '#userToEdit.fld_userEmail#'.
   - For 'fld_userPassword' use '#userToEdit.fld_userPassword#'.
3. Next, locate the <cfselect> tag that creates the 'fld_userInstrument' field.

4. Use the '`selected`' attribute to set the starting value of this field. Make it equal to '`#userToEdit.fld_userInstrument#.`
5. Now, locate the `<cftextarea>` tag block that creates the '`fld_userComment`' field.
6. Use `<cfoutput>` to output the user comment in the body of the `<cftextarea>` tag block.
7. Save and tun the page. Choose a user to view on the '`users.cfm`' page and confirm that its current data appear in the form.

The last field to pre-populate is the '`fld_userRole`' field. This field is made of two radio buttons generated by two `<cfinput />` tags of type '`radio`'. To select the right radio button, you need to use the '`iif()`' function of ColdFusion. Since this is a new function not yet covered in this course, it will be covered in the next video.

Watch video : The iif() function

Add the ‘fld_userID’ hidden field.

After watching the video, there is one last thing to add to the form.

1. Still in the ‘`userEdit.cfm`’ file, locate the `<input>` tag that generates the submit button at the end of the form.
2. Right above this button, create a new `<cfinput />` tag.
3. Make that a hidden field.
4. Set its name to ‘`fld_userID`’ and the value to the corresponding data of the ‘`userToEdit`’ query.

```
<cfinput type="hidden" name="fld_userID"
    value="#userToEdit.fld_userID#" />
```

Create the edit Form processing code.

In ColdFusion Builder, return to the very top of the ‘`userEdit.cfm`’ page and write the form processing code. It follows the same steps as usual, so it won’t be discussed any further here.

The final code should be :

```
<!---Form processing begins here--->
<cfif structKeyExists(form,'fld_editUserSubmit')>
    <cfset aErrorMessages =
        application.userService.validateUser(form.fld_userFirstName
        ,form.fld_userLastName,form.fld_userEmail,form.fld_userPass
        word,form.fld_userPassword) />
    <cfif ArrayIsEmpty(aErrorMessages)>
        <cfset
            application.userService.updateUser(form.fld_userFirstNa
            me,form.fld_userLastName,form.fld_userEmail,form.fld_us
            erPassword,form.fld_userRole,form.fld_userInstrument,fo
            rm.fld_userComment,1,1, form.fld_userID) />
        <cflocation url="users.cfm?updated=true" />
    </cfif>
</cfif>
<!---Form processing ends here--->
```

Save the page when done.

Create a feedback message on the 'users.cfm' page

On the '`user.cfm`' page, right below the `<h1>` tag block of the main title, output a meaningful feedback message if the '`updated`' parameter is detected in the URL.

```
<cfif structKeyExists(url, 'updated')>
  <p class="feedback">The user has been successfully updated.</p>
</cfif>
```

Make sure every file is saved and run the '`users.cfm`' page. Once the page has finished loading in the browser, click on any '`View`' link. This should load the '`editUser.cfm`' page. Submit the form. You should be redirected to the '`users.cfm`' page with the feedback message displayed. If everything works as expected, you can safely move on to the next section.

Create the 'confirmDelete()' JavaScript function.

Again, this function has been discussed in detail previously, so, we won't discuss it any further now. Just make sure you write it immediately below the opening `<cf_admin>` tag in the '`users.cfm`' page.

```
<script>
function confirmDelete(userID)
{
  if(window.confirm('Are you sure you want to delete this user?'))
  {
    window.location.href = 'users.cfm?delete='+userID;
  }
  else
  {
    null;
  }
}
</script>
```

Now that the JavaScript function exists, you should trigger it when the user clicks on the `Delete` link in the sixth cell of the table.

```
<td><a href="javascript:confirmDelete(#fld_userID#);">Delete</a></td>
```

Save and run the page to test your new function.

1. Return to the top of the code and use `<cfif>` and '`structKeyExists()`' to check if the '`delete`' URL parameter exists in the URL.
2. If it does exist, use `<cfset />` to call the '`deleteUser()`' method of the '`userServices.cfc`' component.

```
<!---Delete User--->
<cfif structKeyExists(url,'delete') >
    <cfset application.userService.deleteUser(url.delete) />
</cfif>
```

Save and run the page. Try to delete one of the users to approve to make sure everything works as expected.

Add the [add User] link on the 'users.cfm' page

1. Return to the '`users.cfm`' page and place your cursor after the `<cfif>` blocks that handle the feedback messages to the user.
2. At that location, create a new paragraph.
3. In the paragraph, add a link to the '`userAdd.cfm`' page.

```
<p><a href="userAdd.cfm">[Add a user]</a></p>
```

Save and run the page to test the new link.

Create the 'add user' form on the 'userAdd.cfm' page

1. Open the '`helpers/usersForm.txt`'.
2. Copy / paste the entire content of this file right below the `<h1>` tag block in the '`userAdd.cfm`' page.
3. Update the '`id`' attribute of the opening `<cfform>` tag to '`frm_addUser`'.
4. Also update the '`name`', the '`id`' and the '`value`' of the submit button at the end of the form.
5. Finally, at the very top of the page, use `<cfset />` to create the '`userInstruments`' variable.
6. Make it equal to the query returned by the '`getInstruments()`' method.

Save and run the page. Confirm that the form displays as expected before moving on.

Write the 'add user' form processing code

This form processing code follows the same steps as usual, so it won't be discussed any further here. The only thing to keep in mind is that a user created directly by the site administrator is automatically approved and activated.

```
<!---Form processing begins here--->
<cfif structKeyExists(form,'fld_addUserSubmit')>
  <cfset aErrorMessages =
      application.userService.validateUser(form.fld_userFirstName,f
      orm.fld_userLastName,form.fld_userEmail,form.fld_userPassword
      ,form.fld_userPassword) />
  <cfif ArrayIsEmpty(aErrorMessages)>
      <cfset
          application.userService.addUser(form.fld_userFirstName,fo
          rm.fld_userLastName,form.fld_userEmail,form.fld_userPassw
          ord,form.fld_userRole,form.fld_userInstrument,form.fld_us
          erComment,1,1) />
      <cflocation url="users.cfm?add=true" />
  </cfif>
</cfif>
<!---Form processing ends here--->
```

Save the file when done.

Provide a feedback message on the 'users.cfm' page

Write the following code after the `<cfif>` block of the '`update`' feedback message, but before the '`[Add a User]`' link

```
<cfif structKeyExists(url, 'add')>
    <p class="feedback">A new user has been created.!/p>
</cfif>
```

Save and run the '`users.cfm`' page. Once the page has finished loading in the browser,

1. Click on the '`[Add a user]`' link.
2. Fill and submit the form.
3. The '`users.cfm`' page should reload with the feedback message displayed.
4. Return to the RDS panel of Coldfusion Builder and take a look at the content of the '`TBL_Users`' database table. Confirm that the new user has been correctly inserted in the database.

If everything works as expected, you can safely move on to the next step.

Display the list of approved users on the 'users.cfm' page

1. On the line just above the opening `<cf_admin>` tag, Use `<cfset>` to create the '`activeUsers`' variable.
2. Make it equal to the query returned by the '`getAllActiveUsers()`' method.

```
<!---Get the active users--->
<cfset activeUsers = application.userService.getAllActiveUsers() />
```

Below the table that displays the list of users to approve and outside of its enclosing `<cfif>` block, create a meaningful `<h2>` title.
Just below the `<h2>` title, output the data contained in the '`activeUsers`' query in an HTML table.

```
<h2>Active users</h2>
<table>
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Role</th>
    <th>Instrument</th>
    <th>eMail Address</th>
    <th>&nbsp;</th>
    <th>&nbsp;</th>
  </tr>
  <cfoutput query="activeUsers">
    <tr>
      <td>#fld_userFirstName#</td>
      <td>#fld_userLastName#</td>
      <td>#fld_roleName#</td>
      <td>#fld_instrumentName#</td>
      <td>#fld_userEmail#</td>
      <td><a href="userEdit.cfm?userID=#fld_userID#">Edit</a></td>
      <td><a
        href="javascript:confirmDelete(#fld_userID#);">Delete</a></
        td>
    </tr>
  </cfoutput>
</table>
```

Testing the ‘users.cfm’ Page

Save and run the ‘`users.cfm`’ page. Once the page finishes loading in the browser, perform the following tests.

1. Approve one of the unapproved users. You should see that user being transferred from the first table to the second.
2. Edit the user that you jus approved. When back to the ‘`users.cfm`’ page, you should see a green feedback message displayed at the top of the page.
3. Delete the user that you just approved. You should see the user disappear from the bottom table.
4. Try to edit one of the active users (but do not change their Role in the edit form). When back to the ‘`users.cfm`’ page, you should see a green feedback message displayed at the top of the page.
5. Try to delete the user *Céline Frère*. You should see… an error.

This error requires a bit more explanations. This will be covered in the next video.

Watch video : Basic error handling