12_03 The News Administration

The news administration section of the admin site is the first to perform all four CRUD operations. As a reminder, CRUD stands for Create Read Update Delete.
Even though we will not discuss those operations in that order, this document will guide you through the process of Creating new news, Reading existing news and Updating existing news. Deleting existing news will be covered in a video.

Create the necessary pages in the 'admin' folder.

The news administration requires three '.cfm' pages. Each page has the same basic code.

- A call to the `<cf_admin>` custom tag with a meaningful '`title`' attribute.
- A `<div id="pageBody">` tag block.
- A `<h1>` tag block in the `<div>` to define the main title of the page.

This leads to the following basic code. Of course it needs to be customized for each page.

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

Create those three pages and apply the above basic code to all three pages and customize them according to the following table.

Page Name	Title attribute	<h1> content
news.cfm	HD street band - News administration	News Administration
newsAdd.cfm	HD street band - News administration	Add a news
newsEdit.cfm	HD street band - News administration	Update a news

Once done, test each page one by one. If you named your files correctly, the '`News`' link on the main navigation bar of the Admin site should now be working.

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

To support the additional operations, some methods should be added to the 'newsService.cfc' component.

1. Open the 'newsService.cfc' component. It currently contains four methods.
    - getNewsByID()
    - getLatestNews()
    - getNewsForYears()
    - getNewsYears()

There is one little change to do in the 'getNewsByID()' method.

2. Locate the SELECT query of the 'getNewsByID()' method.
3. Add the 'TBL_NEWS.FLD_NEWSID' field to the list of fields retrieved by the query.
4. Save the component when done.

The final code of the 'getNewsById()' query is

```
<cfquery  name="rsSingleNews">
  SELECT TBL_NEWS.FLD_NEWSCONTENT, TBL_NEWS.FLD_NEWSTITLE,
        TBL_NEWS.FLD_NEWSID, TBL_NEWS.FLD_NEWSCREATIONDATE,
        TBL_USERS.FLD_USERFIRSTNAME, TBL_USERS.FLD_USERLASTNAME
  FROM TBL_NEWS INNER JOIN TBL_USERS ON TBL_NEWS.FLD_NEWSAUTHOR =
        TBL_USERS.FLD_USERID
  WHERE FLD_NEWSID = <cfqueryparam value="#arguments.newsID#"
        cfsqltype="cf_sql_integer" >
</cfquery>
```

Now, you will add a bunch of methods to the component

5. Open the 'helpers/newsComponent.txt' file
6. Copy/paste all the code in the body of the <cfcomponent> tag block of the 'newsService.cfc' component after the existing four methods.
7. When done, save and run the component to access its automatic documentation.

You should see, four additional methods in the component.

- The 'addNews()' method takes three arguments. It creates a new news in the 'tbl_news' table of the database and does not return any value.

- The '`updateNews()`' method takes four arguments. It is used to save the changes made to an existing news and does not return any value.
- The '`deleteNews()`' method takes the '`newsID`' as its only argument. It is used to permanently delete the news whose ID is passed as an argument from the database. It does not return any value.
- The '`validateNewsForm()`' method takes three arguments. It handles the server side data validation and returns an array of error messages.

Return to ColdFusion Builder and carefully review the code of the '`newsService.cfc`' component. Make sure you understand the rest of the component before moving on to the next step.

Cache the new version of the 'newsService.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.

On the news.cfm page, display all the news in a table.

This is the 'Read' operation of CRUD. It is used to display a table containing all the news present in the database.

1. Return to ColdFusion Builder
2. Open the '`admin/news.cfm`' page.
3. At the very top of the code, before the opening `<cf_admin>` tag, write a meaningful comment announcing that you will retrieve all the news from the database.
4. On the next line, use `<cfset>` to create the '`allNews`' local variable.
5. Make that variable equal to the query returned by the '`getLatestNews()`' method of the '`newsService.cfc`' component. Don't pass any value to the '`numNews`' argument.

The code is

```
<!--- retrieve all news from the database --->
<cfset allNews = application.newsService.getLatestNews() />
```

6. Right below the `<h1>` tag block, add a meaningful comment to announce that you will output all the news in a table.
7. Right below the comment, create an HTML table of two rows by five columns.
8. Use the `<th>` tag of html for the cells of the first row and the `<td>` tag for the cells of the second row.
9. Wrap the second row in a `<cfoutput>` tag block that loops over the '`allNews`' query.
10. In the first row of the table, write the column headings in the body of the `<th>` tags (the headings are '`Title`', '`Published date`', `Author`', '`&nbsb`' and '` `' a second time).
11. In the second row, output the corresponding pieces of data contained in the '`allNews`' query.
12. In the last two columns, write '`Edit this news`' and '`Delete this news`'.
13. Make the '`Edit this news`' text a link to the '`newsEdit.cfm`' page. Pass the ID of the current news to the target page as the '`newsID`' URL parameter.

The final code is

```
<!---Display all the news in a table--->
<table>
  <tr>
    <th>Title</th>
    <th>Published date</th>
    <th>Author</th>
    <th> </th>
    <th> </th>
  </tr>
  <cfoutput query="allNews">
    <tr>
      <td>#fld_newsTitle#</td>
      <td>#dateFormat(fld_newsCreationDate, 'mmm-dd-yyyy')#</td>
      <td>#fld_userFirstName# #fld_userLastName#</td>
      <td><a href="newsEdit.cfm?newsID=#fld_newsID#">Edit this
        news</a></td>
      <td>Delete this news</td>
    </tr>
  </cfoutput>
</table>
```

Save and run the page. It should display the list of news sorted by date in descending order. If no error shows up, the 'Read' part of CRUD is finished!

Prepare the 'add news' form on the 'newsAdd.cfm' page

You will now focus on the 'Create' operation. The general process is quite simple to understand :

- The user browses the '`admin/news.cfm`' page and clicks on the '`[Add a news]`' link that you will create in a few moments.
- The '`admin/newsAdd.cfm`' page loads. It displays an empty form to the user.
- The user fills and submits the form.
- Client side and server side data validation take place.
- If the data is validated, the new news is added to the database. If data validation fails, a feedback message is shown to the user.
- When the process successfully completes, the user is redirected to the '`news.cfm`' page where a feedback message acknowledges the creation of the new news.

1. Still on the '`news.cfm`' page, place your cursor between the `<h1>` title and the `<table>` that displays all the news.
2. At that location, create a paragraph that reads, '`[Add a news]`'.
3. Make it a link to the '`newsAdd.cfm`' page

The code is

```
<p><a href="newsAdd.cfm">[Add a news]</a></p>
```

4. Open the '`newsAdd.cfm`'.
5. Open the '`helpers/newsForm.txt`' file
6. Copy/paste the content of that file in the body of the `<div id="pageBody">` tag right below the `<h1>` tag block of the '`newsAdd.cfm`' page.
7. In the form, locate the `<!---Create a date field--->` comment.

On the following line, in the body of the `<dd>` tag you will create a new field. This new field will use one of the many goodies of the ColdFusion forms. As discussed in chapter 6, ColdFusion forms is a superset of the standard HTML form. It means that ColdFusion extends the capabilities of the standard HTML form tags. The date field that you are about to create is a perfect example of this.

8. Create a `<cfinput type="datefield" />` in the body of the `<dd>` tag that follows the `<!– Create a date field--->` comment.
9. As every form fields, this one requires a '`name`' and an '`id`'. Make those two attributes equal to '`fld_newsCreationDate`'.
10. Make sure the '`datefield`' field sends the date in the correct format by adding the '`mask`' attribute. Make it equal to 'MM/DD/YYYY'

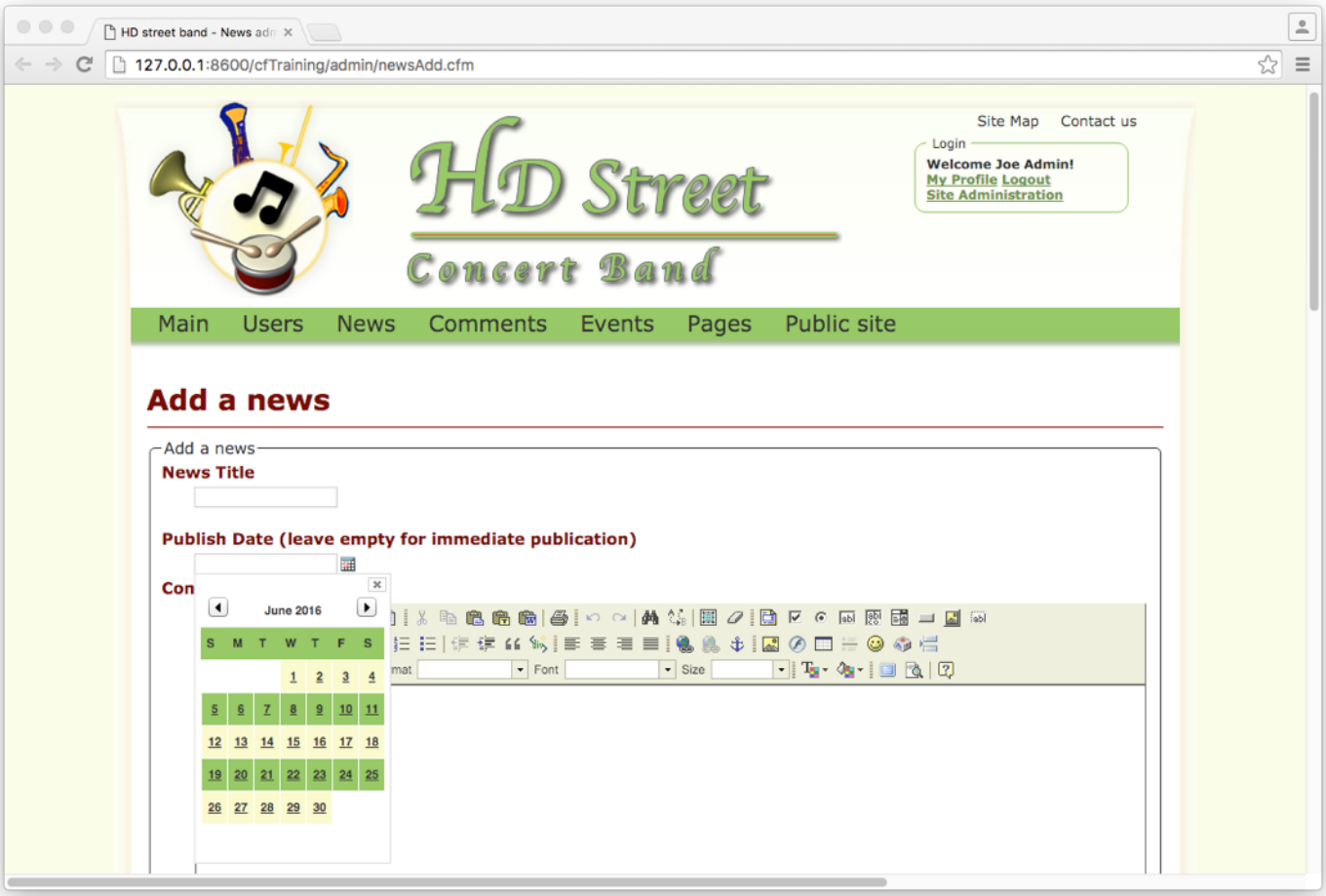
The code is

```
<!---Create a date field--->
<dd>
  <cfinput type="datefield" name="fld_newsCreationDate"
    id="fld_newsCreationDate" mask="MM/DD/YYYY" />
</dd>
```

Save and browse the '`newsAdd.cfm`' page. It should display an empty form with four fields.

- The '`News Title`' field
- The '`Publish Date`' field
- The '`Content`' textarea with the Rich Text editor
- The '`Add News`' submit button

Click on the icon next to the '`Publish Date`' field and let yourself be amazed by the easiness and awesomeness of ColdFusion forms!



Complete the 'Create' operation by writing the 'add news' form processing script.

1. Return to the 'newsAdd.cfm' page in ColdFusion Builder.
2. At the very top of the page, just before the opening `<cf_admin>` tag, create two meaningful comments to mark the beginning and the end of the form processing script.
3. Between the two comments, add a `<cfif>`tag block.
4. Use '`structKeyExists()`' to detect the presence of the '`fld_addNewsSubmit`' button in the form scope.
5. In the `<cfif>` block, use another `<cfif>` to check if the '`form.fld_newsCreationDate`' is equal to an empty string.
6. If it is, use `<cfset />` to set the value of the '`form.fldNewsCreationsDate`' variable to the current date using the '`now()`' function of ColdFusion.
7. Next, use `<cfset>` to create the '`aErrorMessages`' variable.
8. Make it equal to the value returned by the '`validateNewsForm()`' method of the '`newsService.cfc`' component.

9. Use the values gathered by the form as the arguments of the method.
10. On the next line, use `<cfif>` to check if the '`aErrorMessages`' array is empty.
11. If the '`aErrorMessages`' array is empty, call the '`addNews()`' method of the '`newsService.cfc`' component.
12. Use the values gathered by the form as the arguments of the '`addNews()`' method.
13. Finally, use `<cflocation />` to redirect the user to the '`news.cfm?add=true`' page.
14. Save the '`newsAdd.cfm`' page when done.

The final code is

```
<!---Form processing begins here--->
<cfif structKeyExists(form, 'fld_addNewsSubmit')>
  <cfif form.fld_newsCreationDate EQ ''>
      <cfset form.fld_newsCreationDate = now() />
  </cfif>
  <cfset aErrorMessages =
    application.newsService.validateNewsForm(form.fld_newsTitle,fo
    rm.fld_newsCreationDate,form.fld_newsContent) />
  <cfif ArrayIsEmpty(aErrorMessages)>
    <cfset
      application.newsService.addNews(form.fld_newsTitle,form.fld_n
      ewsCreationDate,form.fld_newsContent) />
    <cflocation url="news.cfm?add=true" />
  </cfif>
</cfif>
<!---Form processing ends here--->
```

On the 'admin/news.cfm' page, display a feedback message.

1. In ColdFusion Builder, return to the '`admin/news.cfm`' page.
2. Right below the `<h1>` tag block, create a `<cfif>` block.
3. Use '`structKeyExists()`' to check if the '`add`' variable is present in the `URL` scope.
4. If it is, create a paragraph, with the '`feedback`' css class and write a meaningful feedback message

The code is

```
<cfif structKeyExists(url, 'add')>
  <p class="feedback">The news has been successfully created!</p>
</cfif>
```

Test the CREATE news operation

1. Make sure every file is saved, then, run the '`admin/news.cfm`' page. Remember that you must be logged in as an administrator to access this page.
2. Click on the '`[Add a News]`' link. The '`newsAdd.cfm`' page loads.
3. Try to submit an empty form. The data validation should display a message and the form submission should be canceled.
4. Fill out the form with valid information (could be a news about a news version of the site that is almost there!).
5. When you submit the form, you should be redirected to the '`news.cfm`' page with the feedback message displayed.
6. On the '`admin/news.cfm`' page, the new news should be present in the table. Its position in the table depends on the date you chose for the new news.
7. Browse the public site. The main area of the home page should display the new news, and the side bar should have been updated as well.
8. The '`news`' section of the public site, should also reflect the change.

If no error shows up, the CREATE operation of the news section is now complete.

On the 'newsEdit.cfm' page, prepare the news editing form.

The form that you have to develop here is very similar to the one you developed for the 'Create' operation. The main difference between this form and the previous one is that this form has to display the current information of a specific news. The news to display is the one whose '`newsID`' is passed in the URL scope from the '`admin/news.cfm`' page. The general steps of your work are :

- Making sure the '`URL.newsID`' is correctly passed. Redirect the user to '`admin/news.cfm`' if not.
- Get the current data of the chosen news and store it in a local variable.
- Use that data to pre-populate the form.

1. In ColdFusion Builder, open the 'admin/newsEdit.cfm' page.
2. At the very to of the page, before the opening <cf_admin> tag, use <cfif> and 'structKeyExists()' to check if a 'newsID' parameter is passed in the URL scope.
3. If not, use <cflocation /> to redirect the user to the 'admin/news.cfm' page.

The code is

```
<!--- Check if URL.newsID exists --->
<cfif NOT structKeyExists(url,'newsID')>
    <cflocation url="news.cfm" />
</cfif>
```

Save and run the page. Because you are running the page directly, no parameter is passed in the URL scope and you should be redirected to the 'admin/news.cfm' page. On that page click on any 'Edit this news' link. Confirm that the 'admin/newsEdit.cfm' page loads normally.

Retrieve the current news data from the database.

Place your cursor just below the code you wrote, in the previous section, but before the opening <cf_admin> tag

1. At that location, use <cfset /> to create the 'newsToUpdate' local variable.
2. Make it equal to the query returned by the 'getNewsByID()' method of the 'newsService.cfc' component.
3. Use the 'url.newsID' variable as the only argument of the method.

The code is

```
<!---Get current news data--->
<cfset newsToUpdate =
    application.newsService.getNewsByID(url.newsID) />
```

Create the news update form

You will concentrate on creating the update form itself. It is very similar to the form used in the previous section.

1. Open the '`helpers/newsForm.txt`' file.
2. Copy all the content of that file and paste it in the body of the `<div id="pageBody">` tag, right below the `<h1>` title of the '`admin/newsEdit.cfm`' page.
3. Locate the `<!---Display news form--->` comment.
4. On the very next line, change the '`id`' attribute of the `<cfform>` tag so it reads '`frm_editNews`'.
5. Locate the `<cfinput />` tag that creates the '`fld_newsTitle`' field.
6. Add the '`value`' attribute to that tag. Make it equal to the '`fld_newsTitle`' field of the '`newsToUpdate`' query.
7. Locate the `<!---Create a date field--->` comment.
8. On the next line, in the body of the `<dd>` tag block, use `<cfinput />` to create the same '`datefield`' field as in the previous section.
9. Make its value equal to the '`fld_newsCreationDate`' field of the '`newsToUpdate`' query. Use of the '`dateFormat()`' function to display the date in the correct format.
10. Locate the `<cftextarea>` tag block.
11. In its body, use `<cfoutput>` to display the current value of the '`fld_newsContent`' field of the '`newsToUpdate`' query.
12. Finally, just above the `<input />` tag that creates the submit button, use `<cfinput type=hidden />` to create the '`fld_newsID`' form field. Make its value equal to the corresponding piece of data in the '`newsToUpdate`' query.
13. To make the form final, change the '`value`' of the submit button to '`Update the news`'.
14. Also update the '`name`' and the '`id`' attribute of this submit button.

The final code of the form is

```
<!---Display news form—>
<cfform id="frm_editNews" preservedata="true">
  <fieldset>
    <legend>Add a news</legend>
    <dl>
      <dt><label for="fld_newsTitle">News Title</label></dt>
      <dd><cfinput name="fld_newsTitle" id="fld_newsTitle"
          value="#newsToUpdate.fld_newsTitle#" required="true"
          message="Please enter a valid news title"
          validateAt="onSubmit" /></dd>
      <dt><label for="fld_newsCreationDate">Publish Date (leave empty
          for immediate publication)</label></dt>
```

```
      <!---Create a date field--->
      <dd><cfinput type="datefield" name="fld_newsCreationDate"
          id="fld_newsCreationDate"
          value="#dateFormat(newsToUpdate.fld_newsCreationDate, 'mm/dd/
          yyyy')#" mask="MM/DD/YYYY"></dd>
      <dt><label for="fld_newsContent">Content</label></dt>
      <dd>
        <cftextarea name="fld_newsContent" id="fld_newsContent"
          required="true" message="Please enter a valid news content"
          validateAt="onSubmit" richtext="true" height="500" >
            <cfoutput>#newsToUpdate.fld_newsContent#</cfoutput>
        </cftextarea>
      </dd>
    </dl>
    <cfinput type="hidden" name="fld_newsID"
        value="#newsToUpdate.fld_newsID#" />
    <input type="submit" name="fld_updateNewsSubmit"
        id="fld_updateNewsSubmit" value="Update News" />
  </fieldset>
</cfform>
```

Save the file and run it. You will be redirected to the 'admin/news.cfm' page where you'll be able to click on any 'Update this news' link. Confirm the 'admin/newsEdit.cfm' page reloads and that the form is pre-populated with the current data of the chosen news.
If everything goes as expected, you will write the 'Update' operation of the CRUD in the next section.

Write the 'edit news' form processing script

In ColdFusion Builder, return to the 'newsEdit.cfm' page. At the very to of the code, write two meaningful comments to announce the beginning and the end of the form processing script

1. Between the comments, use <cfif> and 'structKeyExists()' to check if the form has been submitted.
2. Use another <cfif> block is to check if the 'fld_newsCreationDate' is an empty string.
3. If it is, set the value of this piece of data to the current date/time using the <cfset /> tag and the 'now()' function.
4. After this second <cfif> block, use <cfset /> to create the 'aErrorMessages' variable.
5. Make that new variable equal to the value returned by the 'validateNewsForm()' method of the 'newsService.cfc' component.
6. Use the data gathered by the form as the arguments of the method.
7. On the next line, use <cfif> to check if the 'aErrorMessages' array is empty. If it is, the form processing can continue.

8. In the body of the last <cfif> block, call the 'udpateNews()' method of the 'newsService.cfc' component.
9. Use the values gathered by the form as the arguments of this method.
10. When done, redirect the user to the 'news.cfm' page.
11. Notify the 'news.cfm' page that the process has successfully completed by passing the 'update=true' URL parameter.

The final form processing code is the following

```
<!---Form processing begins here--->
<cfif structKeyExists(form,'fld_updateNewsSubmit')>
  <cfif form.fld_newsCreationDate EQ ''>
      <cfset form.fld_newsCreationDate = now() />
  </cfif>
  <cfset aErrorMessages =
    application.newsService.validateNewsForm(form.fld_newsTitle,fo
    rm.fld_newsCreationDate,form.fld_newsContent) />
  <cfif ArrayIsEmpty(aErrorMessages)>
    <cfset
      application.newsService.updateNews(form.fld_newsTitle,form.fl
      d_newsCreationDate,form.fld_newsContent,form.fld_newsID) />
    <cflocation url="news.cfm?update=true" />
  </cfif>
</cfif>
<!---Form processing ends here--->
```

Save the page when done.

Provide a feedback to the user on the 'admin/news.cfm' page

1. In ColdFusion Builder, return to the 'admin/news.cfm' page. Locate the <cfif> block that handles the feedback of the Create procedure.
2. Right below that block, create a second <cfif> block that tests if the 'update' variable exists in the URL scope.
3. Create a meaningful feedback message in the body of this <cfif> block.
4. Don't forget to add the 'class="feedback"' attribute to the paragraph.

The code is

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

Save the page when done.

The 'Update' operation of the CRUD should now be complete. You will now test your new code to make sure everything works as expected.

Testing the Update operation

Make sure every file is saved, then, run the '`admin/news.cfm`' page.

1. When it loads in the browser, click on any '`Update this news`' link.
2. The '`newsEdit.cfm`' page should load.
3. Modify some data in the form and submit it.
4. If nothing goes wrong, you should be redirected to the '`admin/news.cfm`' page with a feedback message displayed.

It is now time to concentrate on the last CRUD operation.

The 'Delete' operation.

The Read, Create and Update operations being done, it is time to focus on the last CRUD operation: the Delete operation. Basically, it is a 3 steps process.

- The user click a '`Delete this news`' link.
- A Javascript alert asks the user to confirm the deletion of the selected news.
- If the user confirms, the news if physically deleted from the database

Because this is a new operation, it will be covered in a video.

Watch the following video :
Deleting a news from the database