

12_04 The Events administration

The events administration of the site is very similar to the news administration section. The same four CRUD operations have to be implemented for the events present in the agenda section of the site. These are the general steps you will follow :

1. Create the 'events.cfm', 'eventAdd.cfm' and 'eventEdit.cfm' pages.
2. Add the necessary methods to the 'eventsService.cfc' component.
3. Display a list of events on the 'events.cfm' page (the 'Read' operation).
4. On the same page, implement the 'Delete' operation.
5. Implement the 'Create' operation on the 'eventAdd.cfm' page.
6. Implement the 'Update' operation on the 'eventEdit.cfm' page

Because the events administration is so similar to the news administration you developed in the previous PDF file, there will be fewer details and explanations in this document. Please, refer to the previous PDF or to the final version of the site in the 'final' folder of the download if there is something unclear.

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
events.cfm	HD street band - Events administration	Events Administration
eventAdd.cfm	HD street band - Events administration	Add an event
eventEdit.cfm	HD street band - Events administration	Update an event

Once done, test the new pages one by one. If you named your files correctly, the 'Events' link of the main navigation bar of the admin site should now be working.

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

The 'eventsService.cfc' component currently contains two methods

- The 'getCurrentEvents()' method.
- The 'getEventByID()' method

You will add four additional methods to support the event administration tasks.

1. Open the 'helpers/eventsComponent.txt' file .
2. Copy/paste its entire content in the body of the <cfcomponent> tag, after the existing two methods of the 'eventsService.cfc' component.
3. Save and run the component.

Four methods have been added to the component.

- The 'updateEvents()' method - Takes six arguments and does not return any value. It handles the 'update' operation.
- The 'addEvent()' method - Takes five arguments and does not return anything either. It is used to save a new event in the database.
- The 'deleteEvent()' method - Takes the ID of the event to delete as its only argument.
- The 'validateEventForm()' method - Is used to implement the server side form validation. It returns an array of error messages.

Return to ColdFusion Builder and inspect the code of these four methods. Notice the use of the <cfscript> tag in the 'validateEventForm()' method. This tag allows you to use the cfscript syntax in the middle of a tag-based component or even in the middle of a regular .cfm page.

Update the 'getCurrentEvents()' method of the 'eventsService.cfc' component.

In its current state, the 'getCurrentEvents()' method fits the needs of the public site. The admin site, however, has some slightly different requirements as it needs the first name and the last name of the user that created the event.

1. Locate the 'getCurrentEvents()' method in the eventsService.cfc component.
2. In the SELECT statement of the <cfquery>, add the 'fld_userFirstName' and the 'fld_userLastName' of the 'tbl_users' in the list of fields to retrieve.
3. Also update the 'FROM' clause of the query. Use a 'INNER JOIN' to describe the relationship between the two tables.

The final code of the <cfquery> is

```
<cfquery name="rsCurrentEvents" maxrows="#arguments.numEvents#">
  SELECT TBL_EVENTS.FLD_EVENTID, TBL_EVENTS.FLD_EVENTNAME,
         TBL_EVENTS.FLD_EVENTDATETIME, TBL_EVENTS.FLD_EVENTLOCATION,
         TBL_EVENTS.FLD_EVENTVENUE, TBL_USERS.FLD_USERFIRSTNAME,
         TBL_USERS.FLD_USERLASTNAME
  FROM TBL_EVENTS INNER JOIN TBL_USERS ON
         TBL_EVENTS.FLD_EVENTAUTHOR = TBL_USERS.FLD_USERID
  WHERE FLD_EVENTDATETIME >= #now()#
  ORDER BY FLD_EVENTDATETIME ASC
</cfquery>
```

Load an instance of the updated 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.

Add the [Add new event] link on the 'events.cfm' page.

Create a link to the 'eventAdd.cfm' right below the <h1> tag block

```
<p><a href="eventAdd.cfm">[Add an event]</a></p>
```

Save and run the page to test your link

Display a list of current events on the 'events.cfm' page

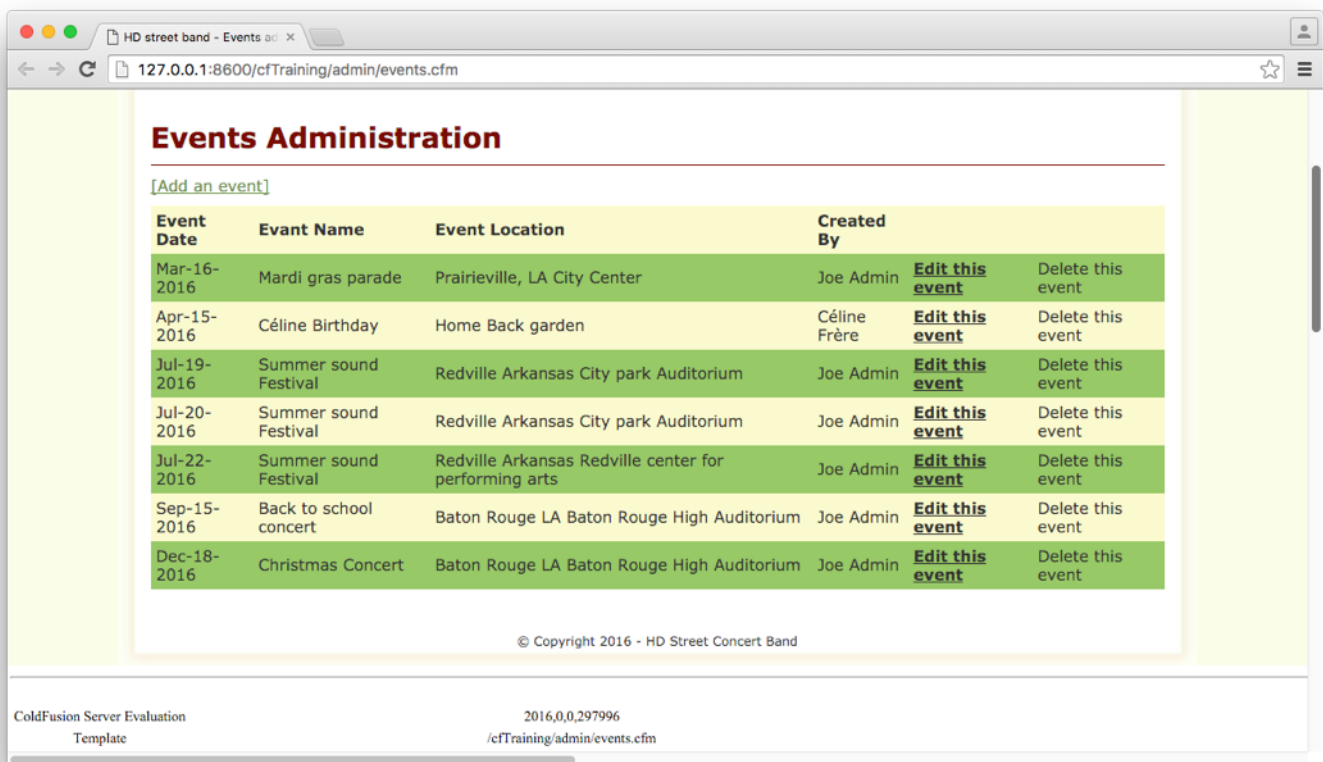
This is a two-step process

- Retrieve the data from the database.
 - Display the data in an HTML table
1. At the very top of the page, use `<cfset />` to create the 'currentEvents' variable.
 2. Make it equal to the value returned by the 'getCurrentEvents()' method of the 'eventsService.cfc'.

```
<!--- Get the list of current events --->  
<cfset currentEvents = application.eventsService.getCurrentEvents() />
```

3. In the body of the `<div id="pageBody">`, right below the `<h1>`, create a `<cfif>` block to test if the 'currentEvents' variable has data to display.
4. If not, write a meaningful message to the user.
5. If yes, create a 6 by 2 table.
6. Wrap the second row of the table in a `<cfoutput>` that loops over the 'currentEvents' query.
7. Create the table headings in the first row (headings are 'Event Date', 'Event Name', 'Event Location', 'Created by', ' ' and ' ' a second time).
8. Output the corresponding data contained in the 'currentEvents' query in the second row of the table.
9. In the last two columns of the table, write 'Edit this event' and 'Delete this event'.
10. Make 'Edit this event' a link to the 'eventEdit.cfm' page and pass the 'eventID' url parameter.

Save and run the page. Make sure it displays with no error. Also test the 'Edit this event' link. It should open the 'eventEdit.cfm' page and pass the 'eventID' url parameter.



Finalize the 'events.cfm' page by implementing the 'Delete' operation

Return to ColdFusion Builder and place your cursor right below the `<cf_admin>` opening tag in the 'events.fm' page.

1. Create an html `<script>` block.
2. Inside the `<script>` block, create the 'confirmDelete()' JavaScript function based on the one created in the 'news' section.

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

3. Execute the function when the 'Delete this event' link in the last cell of the table is clicked.
4. Pass the 'fld_eventID' to the 'confirmDelete()' JavaScript function.
5. At the very top of the page, use <cfif> and 'structKeyExists()' to detect the presence of the 'url.delete' variable.
6. If the variable is detected, use <cfset /> to call the 'deleteEvent()' method of the 'eventsService.cfc' component

```
<!--Delete an event if the url.delete variables exists-->
<cfif structKeyExists(url,'delete')>
    <cfset application.eventsService.deleteEvent(url.delete) />
</cfif>
```

Save and run the 'events.cfm' page. When the page has finished loading in the browser, click on any 'Delete this event' link and confirm your intention to delete an event. When the page reloads, an event should have been removed from the table.

On the 'eventAdd.cfm' page, prepare the 'addEvent' form.

1. Open the 'helpers/eventsForm.txt' file and copy / paste the entire content in the 'eventAdd.cfm' file right below the <h1> tag block.
2. When done, save and run the page to take a look at the form.

When you have a good idea of the form in mind, return to ColdFusion Builder and carefully review the form's code. Pay particular attention to the names of the various form fields.

Write the 'add event' form processing script.

Return to the 'eventAdd.cfm' page in ColdFusion Builder,

Add a <cfif> tag block at the very top of the code to check if the form has been submitted or not. In the <cfif> block

- Call the 'validateEventForm()' method of the 'eventsService.cfc' component to handle the server side data validation.
- Check if the validation succeeded or not.
- Call the 'addEvent()' method if the validation is successful.
- Redirect the user to the 'events.cfm' page. Pass a url parameter to handle user feedback on the target page.

The final code is

```
<!--Form processing begins here-->
<cfif structKeyExists(form,'fld_newEventSubmit')>
    <cfset aErrorMessages =
        application.eventsService.validateEventForm(form.fld_eventName,
        form.fld_eventDate,form.fld_eventLocation,form.fld_eventVenue,
        form.fld_eventDescription) />
    <cfif ArrayIsEmpty(aErrorMessages)>
        <cfset
            application.eventsService.addEvent(form.fld_eventName,form.
            fld_eventDate,form.fld_eventLocation,form.fld_eventVenue,fo
            rm.fld_eventDescription) />
        <cflocation url="events.cfm?add=true" />
    </cfif>
</cfif>
<!--Form processing ends here-->
```

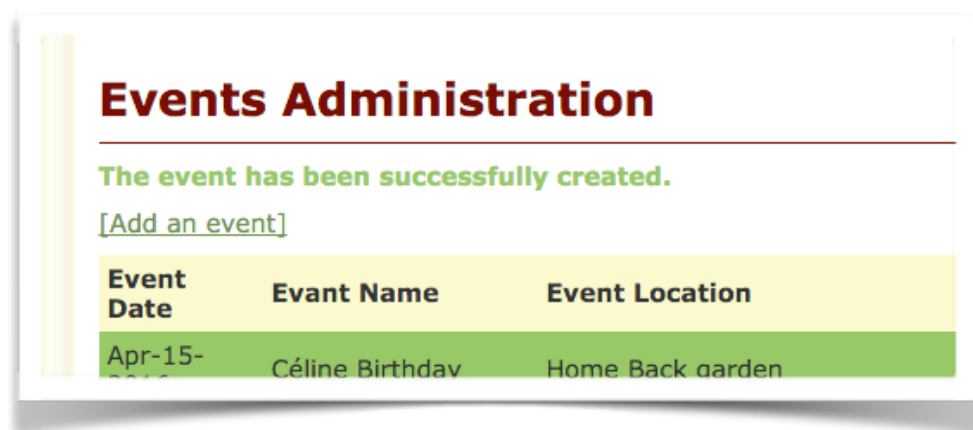
Provide feedback to the user on the 'events.cfm' page

On the 'admin/events.cfm' page, right below the <h1> tag block, use <cfif> to detect the 'add' url parameter. If the parameter is detected, display a meaningful feedback message to the user.

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

Test the 'add Event' operation

Save all the files and run the 'admin/events.cfm' page. When the page has finished loading in the browser, click on the '[Add Event]' link. Fill out and submit the form. You should be redirected to the 'admin/events.cfm' page with a feedback message displayed and with an extra event in the table.



On the 'eventEdit.cfm' page, prepare the 'edit event' form

At the very top of the 'eventEdit.cfm' page, check if the 'eventID' url parameter exists. Redirect the user to the 'admin/events.cfm' page if it does not exist.

```
<!---Check if the eventID parameter exists in the URL scope--->
<cfif NOT structKeyExists(url, 'eventID')>
    <cflocation url="events.cfm" />
</cfif>
```

Right below this piece of code, use `<cfset />` to create the 'eventToUpdate' variable. Make it equal to the value returned by the 'getEventByID()' method and pass the 'url.eventID' variable as the argument.

```
<!---Retrieve current event data--->
<cfset eventToUpdate =
    application.eventsService.getEventByID(url.eventID) />
```

1. Open the 'helpers/eventsForm.txt' file again and copy/paste all of its content just below the `<h1>` tag block in the 'eventEdit.cfm' page.
2. When the form code is pasted, do the following changes to the form
 - Change the 'id' attribute of the `<cfform>` tag so it reads 'frm_updateEvent'.
 - Change the 'name' and the 'id' of the submit button to 'fld_updateEventSubmit'
 - Also change the value of the submit button.
 - Pre-populate the form fields with the values found in the 'eventToUpdate' query/
 - Add the 'fld_eventID' hidden form field in the form to store the ID number of the event to update.
3. Save all files and run the 'admin/events.cfm' page.
4. Click on any 'Update this Event' link.
5. The 'eventEdit.cfm' page should open and display a pre-populated form

Write the 'editEvent' form processing script

This code follows the same basic steps as most of the form processing scripts you wrote so far.

- Detect form submission using `<cfif>`.
- Server side data validation.
- Save the updated event in the database.
- Redirect the user to the 'events.cfm' page with a URL parameter to handle feedback to the user.
- Save the 'eventEdit.cfm' page when done

The form processing script should be written at the very top of the page, before any existing code.

When finished, it should read :

```
<!---Form processing begins here--->
<cfif structKeyExists(form,'fld_updateEventSubmit')>
    <cfset aErrorMessages =
        application.eventsService.validateEventForm(form.fld_eventName,form.fld_eventDate,form.fld_eventLocation,form.fld_eventVenue,form.fld_eventDescription) />
    <cfif ArrayIsEmpty(aErrorMessages)>
        <cfset
            application.eventsService.updateEvent(form.fld_eventName,
            form.fld_eventDate,form.fld_eventLocation,form.fld_eventVenue,form.fld_eventDescription,form.fld_eventID) />
        <cflocation url="events.cfm?update=true" />
    </cfif>
</cfif>
<!---Form processing ends here--->
```

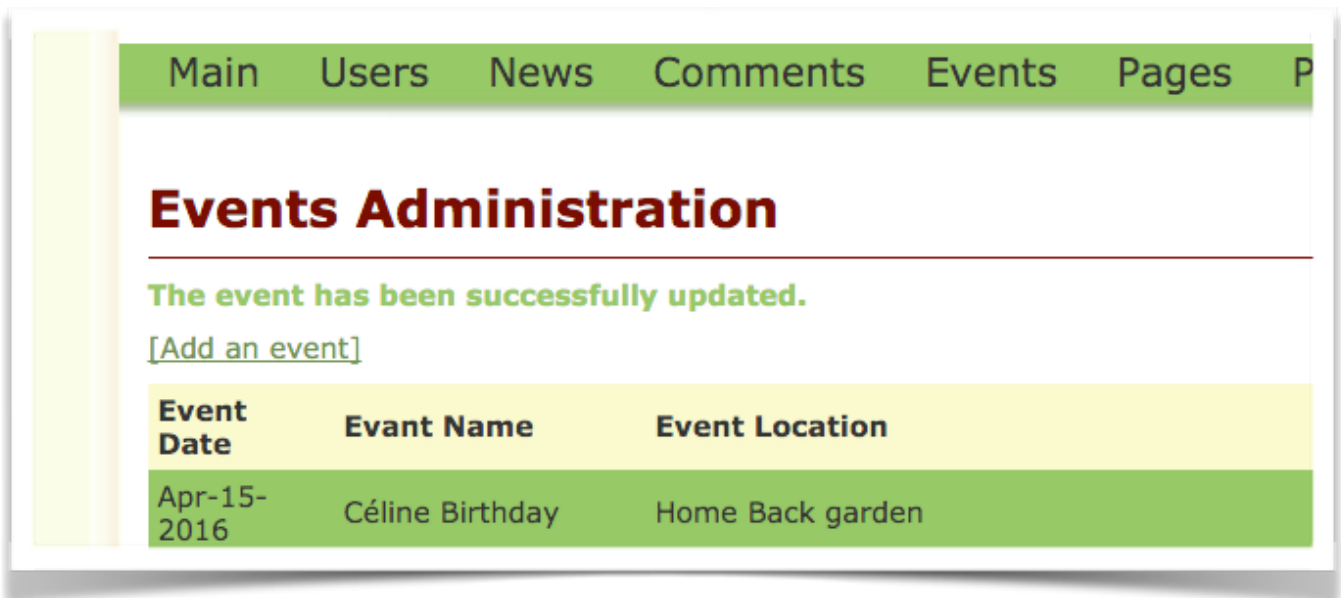
16.Provide feedback to the user on the 'events.cfm' page

Return to the 'admin/events.cfm' page. Locate the `<cfif>` block you wrote earlier to provide user feedback about the successful completion of the add event process. Right below it, create a similar `<cfif>` block to handle user feedback about the successful completion of the update event process.

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

Test the edit event operation

1. Save all the files and run the 'admin/events.cfm' page. When the page has finished loading in the browser, click on any the 'update this event' link.
2. Fill out and submit the form.
3. You should be redirected to the 'admin/events.cfm' page with a feedback message displayed.



The screenshot shows a web application interface for 'Events Administration'. At the top is a green navigation bar with links: Main, Users, News, Comments, Events, Pages, and P. Below the navigation bar is a red heading 'Events Administration'. A green message box states 'The event has been successfully updated.' with a link '[Add an event]'. Below this is a table with three columns: Event Date, Evant Name, and Event Location. The table contains one row of data: Apr-15-2016, Céline Birthday, and Home Back garden.

Event Date	Evant Name	Event Location
Apr-15-2016	Céline Birthday	Home Back garden

All four CRUD operations have now been implemented and tested on the 'events' section of the admin site. In the next PDF file, You will implement the 'comments' section of the admin site.

Next step : 12_05-CommentsAdministration.pdf