

Chapter

26

## SECTION IV: PROCESS FLOW

### Manage Authors [manageAuthors.jsp]

Manage Authors is a link available in adminHome.jsp. This link when clicked invokes the action class named manageAuthorsAction.java.

To locate the action class, the struts configuration file [struts-manageAuthors.xml] holds the following code spec:

```
10 <action name="showManageAuthors" method="view" class="admin.manageAuthorsAction">
11     <result>/admin/manageAuthors.jsp</result>
12 </action>
```

When the user clicks the hyper link, the manageAuthors.jsp is served as shown in diagram 26.1.

## User Interface [showManageAuthors.action → manageAuthors.jsp]

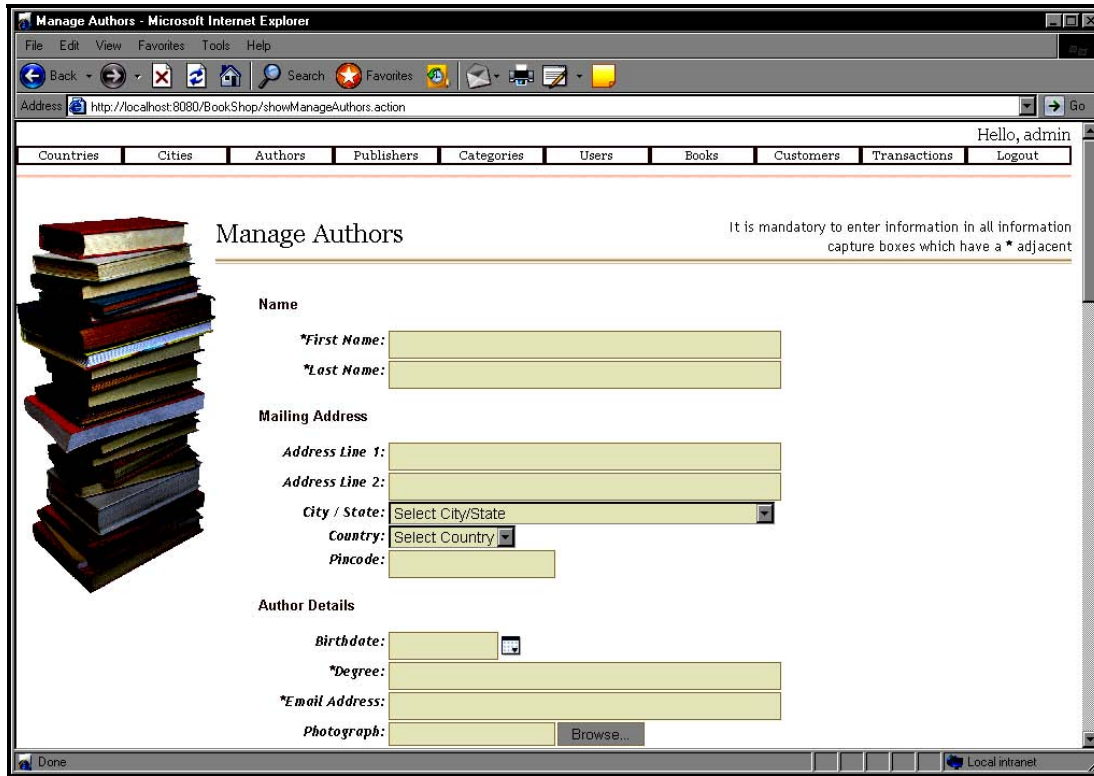


Diagram 26.1

### Process Flow

Prior serving this page, a method called `view()` of the action class, named `manageAuthorsAction.java`, is invoked.

#### `manageAuthorsAction.java [view()]`

*Code Spec [manageAuthorsAction.java]*

```
84  @SkipValidation
85  public String view() throws Exception{
```

```

86     pageNo = 1;
87     getPage();
88     return SUCCESS;
89 }

```

The **view()** method is invoked every time the **Manage Authors** form is invoked.

The **view()** method:

- ❑ Sets the first page as 1
- ❑ Invokes the **getPage()** method to return the first page to **manageAuthors.jsp**

### **manageAuthorsAction.java [getPage()]**

*Code Spec [manageAuthorsAction.java]*

```

110     @SkipValidation
111     public String getPage() throws Exception{
112         pageCount = pg.getTotalPages("Authors");
113         authors = pg.getPage(pageNo-1, "Authors").getList();
114         return SUCCESS;
115     }

```

The **getPage()** method when invoked retrieves the required page's records using the **DAO** object's **[pg]** method and populates the same in the **authors** List object of the **Authors bean class** with the appropriate data. This object is made available to **manageAuthors.jsp** using the **Getter/Setter** methods.

The **DAO** object's method uses **Hibernate** to retrieve such data from the underlying **tables** and **return** the same as an **object**.

## **Add New Authors**

After the **manageAuthors.jsp** loads, the user can key in the required data and click **Save**.

On clicking **Save**, the **FORM** is submitted to **doInsertAuthor.action** and **manageAuthors.jsp** is re-served.

## 310 Struts 2 With Hibernate 3 Project For Beginners

To locate the action class, the struts configuration file [struts-manageAuthors.xml] holds the following code spec:

```
18 <action name="doInsertAuthor" method="insertupdate" class="admin.manageAuthorsAction">
19   <interceptor-ref name="defaultStack"/>
20   <interceptor-ref name="token"/>
21   <result name="success" type="redirect-action">doViewAuthors</result>
22   <result name="input"> admin/manageAuthors.jsp</result>
23 </action>
```

### Process Flow

Prior serving this page, a method called **insertupdate()** of the action class, named **manageAuthorsAction.java**, is invoked.

#### **manageAuthorsAction.java [insertupdate()]**

*Code Spec [manageAuthorsAction.java]*

```
65   public String insertupdate() throws Exception{
66       File photographImage = null;
67       InputStream iStreamPhotograph = null;
68
69       if(Photograph != null) {
70           photographImage = new File(Photograph.toString());
71           iStreamPhotograph = new FileInputStream(Photograph);
72           author.setPhotograph(org.hibernate.Hibernate.createBlob(iStreamPhotograph));
73       }
74
75       if(author.getAuthorNo(>0) {
76           authorsService.update(author);
77       }
78       else {
79           authorsService.insert(author);
80       }
81       return SUCCESS;
82   }
```

The **insertupdate()** method is invoked every time the user clicks **Save**.

The insertupdate() method:

- ❑ Determines the mode i.e. whether the data has to be inserted or updated based on the AuthorNo
- ❑ If **AuthorNo** exists then the data is updated using the DAO object's update() method
- ❑ If **AuthorNo** does not exist then the data is inserted using the DAO object's insert() method

## Edit Author

Every record that is listed in manageAuthors.jsp hyper links to an action called showEditAuthor.action.

```
126 <td width="15%" valign="top" align="left" style="cursor:pointer;" onclick="javascript:location.href='<s:url
    action='showEditAuthor'><s:param name='author.AuthorNo' value='AuthorNo' /></s:url>'">
127   <s:property value="FirstName"/> <s:property value="LastName" />
128 </td>
```

To locate the action class, the struts configuration file [struts-manageAuthors.xml] holds the following code spec:

```
35 <action name="showEditAuthor" method="edit" class="admin.manageAuthorsAction">
36   <result name="success">admin/manageAuthors.jsp</result>
37 </action>
```

When the user clicks the hyper link, manageAuthors.jsp [pre-populated with the selected author's data] is served as shown in diagram 26.2.

## User Interface [showEditAuthor.action → manageAuthors.jsp]

Manage Authors

It is mandatory to enter information in all information capture boxes which have a \* adjacent

Name

\*First Name: Sharanam

\*Last Name: Shah

Mailing Address

Address Line 1: Some Building

Address Line 2: Some Road

City / State: Mumbai - Maharashtra

Country: India

Pincode: 400016

Author Details

Birthdate: 2009-01-07

\*Degree: CDAC

\*Email Address: sharanams@gmail.com

Photograph: Browse...

Diagram 26.2

## Process Flow

Prior serving this page, a method called `edit()` of the action class, named `manageAuthorsAction.java`, is invoked.

`manageAuthorsAction.java [edit()]`

Code Spec [`manageAuthorsAction.java`]

```

103  @SkipValidation
104  public String edit() throws Exception{
105      author = authorsService.edit(author.getAuthorNo());

```

```

106     view();
107     return SUCCESS;
108 }

```

The **edit()** method is invoked when a user clicks a record from the data grid in `manageAuthors.jsp`.

The **edit()** method:

- ❑ Retrieves the select record's data based on the `AuthorNo` received as parameter from `manageAuthors.jsp`. This is done with the help of the DAO object's **edit()** method. The **edit()** method returns an object of the Authors bean class which is made available to `manageAuthors.jsp` [for pre-populating the form data] using the Getter/Setter methods
- ❑ Invokes the **view()** method to take care of the data grid population

After the user makes the desired changes and clicks Save, the **FORM** is submitted to `doInsertAuthor.action` and `manageAuthor.jsp` is re-served.

To locate the action class, the struts configuration file [`struts-manageAuthors.xml`] holds the following code spec:

```

18 <action name="doInsertAuthor" method="insertupdate" class="admin.manageAuthorsAction">
19   <interceptor-ref name="defaultStack"/>
20   <interceptor-ref name="token"/>
21   <result name="success" type="redirect-action">doViewAuthors</result>
22   <result name="input"> admin/manageAuthors.jsp</result>
23 </action>

```

## Process Flow

Prior serving this page, a method called **insertupdate()** of the action class, named `manageAuthorsAction.java`, is invoked.

### `manageAuthorsAction.java` [insertupdate()]

*Code Spec* [`manageAuthorsAction.java`]

```

65 public String insertupdate() throws Exception{
66     File photographImage = null;
67     InputStream iStreamPhotograph = null;
68
69     if(Photograph != null) {

```

## 314 Struts 2 With Hibernate 3 Project For Beginners


```
70     photographImage = new File(Photograph.toString());
71     iStreamPhotograph = new FileInputStream(Photograph);
72     author.setPhotograph(org.hibernate.Hibernate.createBlob(iStreamPhotograph));
73 }
74
75 if(author.getAuthorNo(>0) {
76     authorsService.update(author);
77 }
78 else {
79     authorsService.insert(author);
80 }
81 return SUCCESS;
82 }
```

The `insertupdate()` method is invoked every time the user clicks **Save**.

The `insertupdate()` method:

- ❑ Determines the mode i.e. whether the data has to be inserted or updated based on the AuthorNo
- ❑ If **AuthorNo** exists then the data is updated using the DAO object's `update()` method
- ❑ If **AuthorNo** does not exist then the data is inserted using the DAO object's `insert()` method


## Delete Author

Every record that is listed in `manageAuthors.jsp` holds  which is hyper linked to an action called `doDeleteAuthor.action`.

```
122 <a href="<s:url action="doDeleteAuthor"><s:param name="author.AuthorNo" value="AuthorNo"
    /></s:url">
123 
124 </a>
```

To locate the action class, the struts configuration file [`struts-manageAuthors.xml`] holds the following code spec:

```
30 <action name="doDeleteAuthor" method="delete" class="admin.manageAuthorsAction">
31 <result name="success" type="redirect-action">doViewAuthors</result>
32 <result name="error">admin/manageAuthors.jsp</result>
33 </action>
```

When the user clicks the  hyper link, the selected record is deleted and `manageAuthors.jsp` is re-served as shown in diagram 26.3.



## User Interface [doDeleteAuthor.action → manageAuthors.jsp]

Manage Authors - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites

Address http://localhost:8080/BookShop/showManageAuthors.action Go

Hello, admin

Countries Cities Authors Publishers Categories Users Books Customers Transactions Logout

Manage Authors

It is mandatory to enter information in all information capture boxes which have a \* adjacent

Name

\*First Name:

\*Last Name:

Mailing Address

Address Line 1:

Address Line 2:

City / State:

Country:

Pincode:

Author Details

Birthdate:

\*Degree:

\*Email Address:

Photograph:  Browse...

Done Local intranet

Diagram 26.3

## Process Flow

Prior serving this page, a method called `delete()` of the action class, named `manageAuthorsAction.java`, is invoked.

`manageAuthorsAction.java [delete()]`


Code Spec [`manageAuthorsAction.java`]

```

91     @SkipValidation
92     public String delete() throws Exception{
93         try{
94             authorsService.delete(author.getAuthorNo());

```

```
95     }
96     catch(Exception e) {
97         addActionError(e.getCause().toString().split("[:]")[1]);
98         return ERROR;
99     }
100    return SUCCESS;
101 }
```

The **delete()** method is invoked when a user clicks  from the data grid in `manageAuthors.jsp`.

The **delete()** method in turn invokes the DAO object's `delete()` method passing it the parameter received from `manageAuthors.jsp` using the Authors bean object's Getter method.

The DAO object's `delete()` method uses Hibernate and deletes the chosen record from the underlying tables.

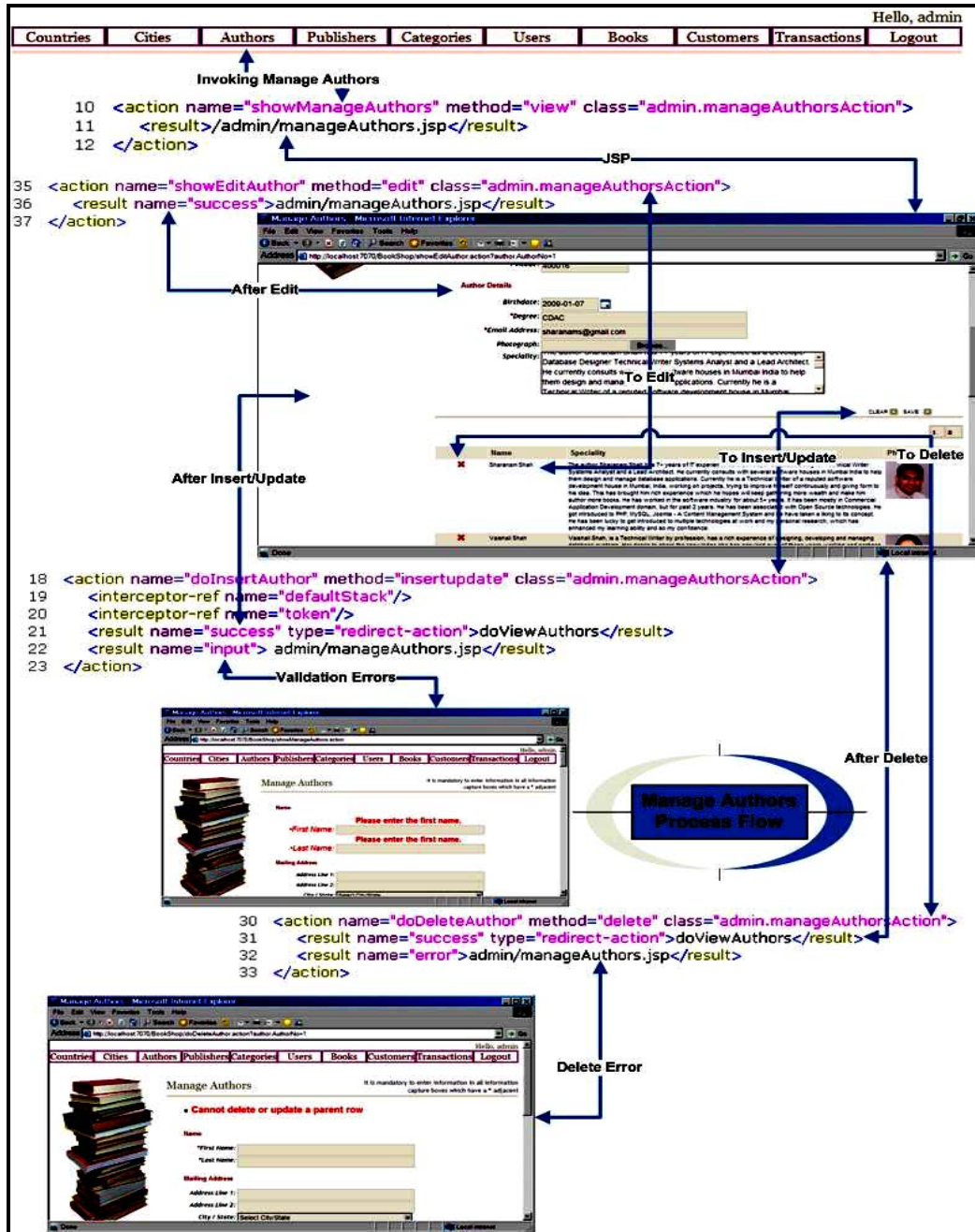


Diagram 26.4: Manage Authors Process Flow

