Internet Engineering

Tomasz Babczyński, Zofia Kruczkiewicz
Tomasz Kubik

Information systems modelling—UML and service description languages
Laboratory 2

Choose yourself and new technologies
Example of the architecture of web application based on synchronization data by application
Architecture of web application based on synchronization data by application

- many clients of the **Client Tier** as the www pages
- many JSF pages of the **Presentation Tier** – each client has own pages
- common **Application Service Component** of the **Business Tier** (*ApplicationBean1*) as the remote facade of the **Business Service Subtiers** (Java Application project) based on the POJO objects – each client uses the common *ApplicationBean1*
Installation the Visual Web JSF

• Download the Netbeans 6.7.1. [http://netbeans.org/downloads/6.7.1/index.html](http://netbeans.org/downloads/6.7.1/index.html) - version all
  Note: During the customization step set the installation option of the Apache Tomcat 6.0.18 server

• Install the UML plugin:
  – Chose the Tools from the Menu Bar
  – Click the Plugins item from the pop menu
  – Click the Available Plugins tab
  – Set the check box of the UML item
  – Click the Install button for installing the UML plugin
  – Execute the final tasks of the installing process
  You can see the result of the plugin installation at the Installed tab
  – Click the Installed tab
  – Set the Show details check box - the shown list now includes the UML item

• Install the Visual JSF plugin
  – Chose the Tools from the Menu Bar
  – Click the Plugins item from the pop menu
  – Click the Available Plugins tab
  – Set the check boxes of the Visual JSF and VisualJSF Runtime items
  – Click the Install button for installing the Visual JSF plugins
  – Execute the final tasks of the installing process
  You can see the result of the plugin installation at the Installed tab
  – Click the Installed tab
  – Set the Show details check box - the shown list now includes the Visual JSF items
Steps for building the presentation tier of a multitier web application

1. Creating a new Java Web Application project
2. Creating page fragment components
3. Create new Visual Web JSF pages
4. Adding Fragments to a Second Page
5. Add Binding Attributes – for hyperlinks of the Menu Page Fragment
6. Disabling the Link for the Current Page
7. An implementation of the web page of the Add titles use case
8. An implementation of the web page of the Add books use case
9. An implementation of the web page of the Find Accessible book use case
10. Linking the projects together: BT_LibraryWeb1 and Code-Library1 (see: INEA_102_Laboratory1.pdf)
11. Events handling for Add titles – an implementation of the service of the first use case
12. Events handling for Add books – an implementation of the service of the second use case
13. Events handling for Find the accessible book – an implementation of the service of the third use case
14. Build, deploy and run the project
1. Create a new web application project and call it BT-LibraryWeb1

1.1. Create a new visual web application project

1.2. The location and naming a new project

1.3. Selecting a GlassFish v.2 applications server

1.4. Selecting a Visual Web JavaServer Faces Framework
2. Creating page fragment components

A page fragment is a portion of a page, such as a header, footer, or navigation bar, that can be reused in other pages. For example, you might put a common element such as a graphic or a Search field in a page fragment and then include that fragment as a header in all pages in the application. You might also include your company name and copyright information in a page fragment and use that fragment as your application’s footer. Like a main page, a page fragment is a JSP page with its own associated page bean; however, the file extension of a page fragment is jspf instead of jsp.

2.1. From the Layout section of the Components Palette, drag a Page Fragment Box component onto the upper left corner of the page. The Select Page Fragment dialog box opens.
2.2 Click Create New Page Fragment.

2.3. Type Logo in the Name field and click OK. The page fragment appears on the page. In addition, the page fragment is added to the Projects window and to a <div> block in the Navigator window.

2.4. Click Close to close the Select Page Fragment dialog box.
2.5. The dotted line in the Visual Designer shows the size of the page fragment. The default size is 400 pixels wide by 200 pixels high.
2.6. Now you define the content of the Logo fragment, as shown in the figure. Any changes you make to a fragment must be made in the fragment itself, and not in the page. Open the Logo fragment by double-clicking the component in the Visual Designer.

2.7. The white background shows the size of the page fragment. In the Properties window, set the Width property to 600px and the Height prop property to 100px. (default size: Width property is 400px and the Height prop property is 200px.)

2.8. From the Basic section of the Palette, drag an Image component into the upper left corner page fragment.
2.9. In the Properties window, click the ellipsis button for the Image's url property. Add the company logo to the page fragment as follows:
   2.9.1. In the dialog box, click Add File.
   2.9.2. Navigate to the folder where you stored the company logo, Garden.jpg or another file, and select the image.
       If you haven’t done so already, save the Garden JPEG file to your filesystem (or another file).
   2.9.3. Click Add File. The IDE copies the image to the resources directory of the project and displays the image's relative URL.
   2.9.4. Click OK.
2.10. If necessary, click and drag Garden.jpg to position it within the page fragment’s borders (with resizing).
After resizing the image
2.11. From the Basic section of the Palette, drag a Static Text component onto the Logo page fragment and drop it to the right of Image component. Be sure to drop the component on the page fragment and not on the Page1 page. Change the text of this component to Example of multilayer application.

2.12. In the Properties window, click the ellipsis button for the Static Text's style property. Change the size of font od Static Text as follows:

2.12.1. In the dialog box, click Font.
2.12.2. Select the size of fonts equals 18
2.12.3. Click OK.
After changing the style of Static Text

Example of multilayer application

Did you forget to reset the value after changing the option?

style

Example of multilayer application

Example of multilayer application
2.13. Click the Page1 tab to view the updates to the Logo fragment. Adjust the layout of the components on the page as necessary.
2.14. Place a second Page Fragment Box component on the left side of the page under the Logo page fragment. Name this page fragment Menu. Adjust the layout of the components on the Page1 page as necessary.
2.15. Open the Menu fragment by double-clicking the component in the Visual Designer.
2.16. In the Properties window, set the Width property to 280px and the Height property to 200px.
2.17. From the Basic section of the Palette, drag a Hyperlink component and drop it in the page fragment. Set the text of the Hyperlink to Main Page.
2.18. Repeat three times these steps for inserting a hyperlink into the page Menu.
2.19. Set the text of the first hyperlink component to Main page.
2.20. The results of changing text of other hyperlinks

Main page

Add titles in application

Add books in application

Check an accessible book of a select title
2.21. Results of Creating the Logo and Menu Fragments
3. Create new Visual Web JSF pages
   You will design the three new pages shown in the figure below.
3.1. Click the File of the MenuBar window, click the New File >
3.2. Click the New File > JavaServer Faces node and choose > Visual Web JSF Page.

This template will create a new blank web page. Design the page by dragging components from the Palette.
3.3. Name the new page Titles and click Finish.
3.4. The Titles page opens in the Visual Designer.
3.5. After creating Titles, Books and Accessible VisualWeb JSF pages
4. Adding Fragments to a Second Page

4.1. **Click the Page1 tab.** You will copy the Page Fragment component from this page to the Titles page.

4.2. From the Page 1 - Navigator window, **hold down the Control key and select both div elements.** Be sure you select the div elements themselves and not just their jsp:directive.include child elements.

4.3. **Right-click the selection and choose Copy from the pop-up menu.**
4.4. Click the Titles tab.
4.5. In the Navigator window, expand the Titles > page1 > html1 > body1 node. Right-click form1 and select Paste. The components you copied from Page1.jsp appear in the Visual Designer.
4.6. Repeat these activities for the Books and AccessibleBook pages
4.7. Setting the url Properties of the hyperlink components of Menu Page Fragment Component

4.7.1. Open the Menu fragment by double-clicking the component in the Visual Designer.

4.7.2. In the Properties window of the hyperlink1, set the url property to /faces/Page1.jsp.
4.7.3. In the Properties window of the hyperlink2 set the url property to /faces/Titles.jsp.
4.7.4. In the Properties window of the hyperlink3 set the url property to /faces/Books.jsp.
4.7.5. In the Properties window of the hyperlink4 set the url property to /faces/AccessibleBook.jsp.
4.8. Click the Run Main Project button to run the application. Verify that the Main page, Add titles in application, Add books in application and Check an accessible links work correctly.
5. Add Binding Attributes – for hyperlinks of the Menu Page Fragment

(If we plan to script the component in Java code we must manually add a binding attribute to the component. To do so, right-click the component and choose Add Binding Attribute.)
6. Disabling the Link for the Current Page

6.1. Add code to disable the Main page link on the Page1 page - add the following code to the prerender method (shown in the figure below)

6.2. Right-click in the Java Editor and choose Fix Imports. The IDE adds the following import statement:

```java
import com.sun.webui.jsf.component.Hyperlink;
```
6.3. Add code to disable the Add titles in application link on the Titles page. Add the following code to the prerender method (shown in the figure below):

```java
public void prerender() {
    Menu menuDiv = (Menu) getBean("Menu");
    Hyperlink link2 = menuDiv.getHyperlink2();
    link2.setDisabled(true);
}
```

6.4. Right-click in the Java Editor and choose Fix Imports. The IDE adds the following import statement:

```java
import com.sun.webui.jsf.component.Hyperlink;
```
6.5. Add code to disable the Add books in application link on the Books page - add the following code to the prerender method (shown in the figure below)

6.6. Right-click in the Java Editor and choose Fix Imports. The IDE adds the following import statement:

```java
import com.sun.webui.jsf.component.Hyperlink;
```
6.7. Add code to disable the Check an accessible book of a select title link on the AccessibleBook page - add the following code to the prerender method (shown in the figure below)

6.8. Right-click in the Java Editor and choose Fix Imports. The IDE adds the following import statement:

import com.sun.webui.jsf.component.Hyperlink;
6.9. Click the Run Main Project button to run the application.
6.10. Verify that links work correctly. Note: You must clean and build project after changes in code and therefore you must to restart the GlassFish v2.1 server. In the Services window, right-click the Sun Java System Application Server/GlassFish node and choose Start / Stop Server. Alternatively, click the Start the Server button in the server log tab in the Output window.
7. An implementation of the web page of the Add titles use case - you define the content of the Titles page shown in figure below

7.1. From the Basic section of the Palette, drag Button component and drop to the right side of the Titles page.

7.2. Add a binding attribute to the button component - hold down the Control key and select the Button component. Right-click the selection and choose Add Binding Attribute from the pop-up menu.

7.3. Set the id property of Button component to addtitle

7.4. From the Basic section of the Palette, drag Message Group component and drop to the right on the Titles page.
7.5. Now you create the new FormTitle Page Fragment (see p.2)
7.6. Next you define the content of the FormTitle fragment as shown in figure. Any changes you make to a fragment must be made in the fragment itself, and not in the page. Open the FormTitle fragment by double-clicking the component in the Visual Designer.

7.7. The white background shows the size of the page fragment. In the Properties window, set the Width property to 250px and the Height prop property to 150px (default size: Width property is 400px and the Height prop property is 200px).

7.8. From the Basic section of the Palette, drag Grid Pane component. Set the columns property to 2 and the id property to titleapplicationpane.

7.9. From the Basic of the Palette, drag the Label components into the Grid Panel component and set the text properties of Label components as like as in the figure on the right.

7.10. From the Basic of the Palette, drag the Text Field components into the Grid Panel component. In theirs Properties windows, you must set the id property of the Text Field components respectively: title, author, ISBN, publisher, actor.

7.11. Add a binding attribute to the all Text Field components - hold down the Control key and select the Text Field component. Right-click the selection and choose Add Binding Attribute from the pop-up menu.
7.12. In the Properties Window of the title, author, ISBN and publisher TextField components (except the actor component), you must set the check box of the required property to indicate that the input value for this field is mandatory, and failure to provide one will trigger a validation error.
7.13. In the Properties Window of the Title Label component you must set the for property to the title Text Field component.
7.14. This Input component use this attribute to specify the labeled component (the red asterisk).
7.15. You must repeat it for the labels like as Author, ISBN and Publisher.
7.16. Now you create the new TitlesApplication Page Fragment (see p.2)

7.17. Next you define the content of the TitlesApplication fragment as shown in figure. Any changes you make to a fragment must be made in the fragment itself, and not in the page. Open the TitlesApplication fragment by double-clicking the component in the Visual Designer.

7.18. The white background shows the size of the page fragment. In the Properties window, set the Width property to 250px and the Height prop property to 200px. (default size: Width property is 400px and the Height prop property is 200px.)

7.19. From the Basic section of the Palette, drag Drop Down List component into the upper left corner page fragment.

7.20. Set the id property of the Drop Down List component to titles

7.21. Add a binding attribute to the titles component - hold down the Control key and select the titles component. Right-click the selection and choose Add Binding Attribute from the pop-up menu.

7.22. Hold down the Control key and select the titles component. Right-click the selection and choose Auto-Submit on Change from the pop-up menu.
8. An implementation of the web page of the Add books use case - you define the content of the Titles page shown in figure below

8.1. From the Basic section of the Palette, drag Button component to the right side of the Books page.

8.2. Add a binding attribute to the button component - hold down the Control key and select the Button component. Right-click the selection and choose Add Binding Attribute from the pop-up menu.

8.3. Set the id property of Button component to addbook1

8.4. Now you create the new FormBook Page Fragment. Next you define the content of the FormBook fragment as shown in figure. Any changes you make to a fragment must be made in the fragment itself, and not in the page. Open the FormBook fragment by double-clicking the component in the Visual Designer.
8.5. The white background shows the size of the page fragment. In the Properties window, set the Width property to 200px and the Height prop property to 100px (default size: Width property is 400px and the Height prop property is 200px)

8.6. From the Basic section of the Palette, drag Grid Pane component. Set the columns property to 2 and the id property to booksApplicationPane.

8.7. From the Basic of the Palette, drag the Label components into the Grid Panel component and set the text properties of Label components as like as in the figure on the right.

8.8. From the Basic of the Palette, drag the Text Field components into the Grid Panel component. In theirs Properties windows, you must set the id property of the Text Field components respectively: number, period.

8.9. Add a binding attribute to the all Text Field components - hold down the Control key and select the Text Field components. Right-click the selection and choose Add Binding Attribute from the pop-up menu.
8.10. In the Properties Window of the number TextField component, you must set the validatorExpression with the LongRangeValidator selected from the pop menu.
8.11. In the Navigator Window you must select the longRangeValidator1. In its Properties Window you must set the maximum and minimum values as shown below.
8.12. In the Properties Window of the Title Label component you must set the for property to the title Text Field component.
8.13. This Input component use this attribute to specify the labeled component (the red asterisk).
8.15. Now you create the new BooksApplication Page Fragment (see p.2).
8.16. Next you define the content of the BooksApplication fragment as shown in figure. Any changes you make to a fragment must be made in the fragment itself, and not in the page. Open the BooksApplication fragment by double-clicking the component in the Visual Designer.
8.17. From the Basic section of the Palette, drag Drop Down List component into the upper left corner page fragment.
8.18. Add a binding attribute to the Drop Down List component - hold down the Control key and select the Drop Down List component. Right-click the selection and choose Add Binding Attribute from the pop-up menu.
You define the content of the Titles page shown in figure below

9.1. From the Basic section of the Palette, drag Button component to the right side of the Books page.

9.2. Add a binding attribute to the Button component - hold down the Control key and select the Button component. Right-click the selection and choose Add Binding Attribute from the pop-up menu.
9.3. Now you create the new FormPeriod Page Fragment. Next you define the content of the FormPeriod fragment as shown in figure. Any changes you make to a fragment must be made in the fragment itself, and not in the page. Open the FormPeriod fragment by double-clicking the component in the Visual Designer.

9.4. The white background shows the size of the page fragment. In the Properties window, set the Width property to 200px and the Height prop property to 60px (default size: Width property is 400px and the Height prop property is 200px).

9.5. From the Basic section of the Palette, drag Grid Pane component. Set the columns property to 2. 
and the id property to booksApplicationPane

9.6. From the Basic of the Palette, drag the Label component into the Grid Panel component and set the text properties of Label component as like as in the figure on the right.

9.7. From the Basic of the Palette, drag the Text Field components into the Grid Panel component. In its Properties window, you must set the id property respectively: period.

9.8. Add a binding attribute to the all Text Field component - hold down the Control key and select the Text Field component. Right-click the selection and choose Add Binding Attribute from the pop-up menu.
10. Linking the projects together: BT_LibraryWeb1 and Code-Library1 (see: INEA_102_Laboratory1.pdf)

10.1. In the Projects window, right Click the BT_LibraryWeb1 project node and select Properties from its context menu.
10.2. In the Project Properties dialog, click the Libraries node in the Categories section on the left. Then, click Add Project.
10.3. In the Add Project window, browse to the location with the Code-Library1 project and select it.

10.4. Then click Add Project JAR Files.
10.5. This step adds the Code-Library1 jar file to the compile-time libraries for the BT_LibraryWeb1 project, and Code-Library1 appears in the Project Properties screen. Click OK in the Project Properties screen to complete the process.
11. Events handling for Add titles – an implementation of the service of the first use case

11.1. In Design view:
Right-click the button and choose Edit action Event Handler. It generates the empty method as the button method:

```java
public String addtitle_action() {
    // TODO: Replace with your code
    return null;
}
```

Code the action event handler, the button method that is called when the user clicks the button.
11.2. The addtitle_action method is used to the event handle of the click of the addtitle button. It must get data of the title (the form_title method of the FormTitle fragment) and deliver it to the prepare_titles method of the ApplicatioBean1 object. Add the code shown below to the Titles.java.

11.3. The prerender method is used for last activities before the Response phase: disable the link to itself and clear the form fields (refresh_form method of the FormTitle fragment) - add the code shown below to the Titles.java.
11.4. Add the code shown below to the refresh method to clear the form fields to FormTitle.java – this method is called from the prerender method of the Titles class.

```java
public void refresh_form()
{
    titleapplicationpane.setRendered(true);
    title.setText("");
    author.setText("");
    ISBN.setText("");
    publisher.setText("");
    actor.setText("");
}
```
11.5. Add the code shown below to the form_title method of the FormTitle class for getting the content of form fields.
11.6. These methods, shown in the figure on the right, are for changing the space char into the "_" char because of the fact that the space is used to split the string, as the return value from the toString method of the TTitle_book class.
11.7. The ApplicationBean1 object as the Application Service design pattern – it enables remote calls the methods of the TFacade class as the Facade design pattern. Add the code shown below to the ApplicationBean1.java.

```java
public ApplicationBean1() {...}

/**...*/
@override
public void init() {...}

private TFacade facade = new TFacade();

public TFacade getTFacade() {
    return facade;
}

public void setTFacade(TFacade facade) {
    this.facade = facade;
}

public void add_title(String data[]) {
    facade.add_title_book(data);
}
```
11.8. The prepare_titles method of ApplicationBean1 calls the add_titles method of the TFacade class and the own preparetitles method for changing the content of the titles array which is binded with the titles component of the TitlesAppliaction fragment. Add the following code to the ApplicationBean1.java.
11.9. Click the TitlesApplication tab to open it in the Visual Designer.
11.10. Hold down the Control key and select the titles component. Right-click the selection and choose Bind to Data from the pop-up menu.
11.11. It opens the Bind to Data dialog. From the Bind to an Object tab, select ApplicationBean1 >titles_. Click OK.
12. Events handling for Add books – an implementation of the service of the second use case
Add the code shown below to the Books.java (the button addbook1_action method as the action event handler is generated in the same way as addtitle_action.).

```java
@Override
public void prerender() {
    Menu menudiv = (Menu) _jspx_page_context.getJspApplicationContext().createJspComponent(menudiv);
    Hyperlink link3 = menudiv.getHyperlink3();
    link3.setDisabled(true);
    FormBook bookdiv = (FormBook) _jspx_page_context.getJspApplicationContext().createJspComponent(bookdiv);
    bookdiv.refresh_form();
}

public String addbook1_action() {
    // TODO: Process the action. Return value is a navigation
    // case name where null will return to the same page.
    TitlesApplication titlesdiv = (TitlesApplication) _jspx_page_context.getJspApplicationContext().createJspComponent(titlesdiv);
    String data1[] = titlesdiv.select_title();
    FormBook bookdiv = (FormBook) _jspx_page_context.getJspApplicationContext().createJspComponent(bookdiv);
    String data2[] = bookdiv.form_book();
    if (data1 != null && data2 != null)
        getApplicationBean1().add_book(data1, data2);
    return null;
}
```
12.1. Add the code shown below to the TitlesApplication.java – the select_title method of the TitlesApplications class selects the data of the title of the book by the click on the item of the titles list as the titles component of the TitlesApplication fragment.

```java
public String[] select_title() {
    String what;
    String select = (String) titles.getSelected();
    if (select == null)
        return null;
    int nr = Integer.parseInt(select);
    if (nr == 0) {
        return null;
    }
    Option help[] = getApplicationBean1().getTitles_();
    String help1 = help[nr - 1].getLabel();
    String help2[] = help1.split(" ");
    if (help2.length == 8) // what type of title of book
    {
        help1 = "";
        what = "0";
    } else {
        help1 = help2[9];
        what = "2";
    }
    String data1[] = (what, (String) help2[5], help1);
    return data1;
}```
12.2. Add the code shown below to the FormBook.java.

```java
public void refresh_form()
{
    booksApplicationPane.setRendered(true);
    number.setText("");
    period.setText("");
}

public String[] form_book()
{
    // TODO: Process the action. Return value is a navigation case name where null will return to the same page.
    String what_book_type;
    if (number.getText().equals(""))
        return null;
    if (period.getText().equals(""))
        what_book_type = "0";
    else
        what_book_type = "1";
    String data[] = {what_book_type, (String)number.getText(), (String)period.getText()};
    return data;
}
```
12.3. Add the code shown below to the TitlesApplication.java – the titles_processValueChange method is called when happens event as the clicking the item of the titles component (the titles_processValueChange method as the action event handler is generated in the same way as addtitle_action). This method calls own select_titles method. Next the returned array with data of chosen title is delivered to the prepare_books method of ApplicationBean1 object which updates the books_array. This array is binded with the Drop Down List component of the BooksAppliaction fragment and provides the view of the book list of the chosen title.

```java
public void titles_processValueChange(ValueChangeEvent event) {
    String data[] = select_title();
    if (data != null) {
        getApplicationBean1().prepare_books(data);
    }
}
```
12.4. Click the BooksApplication tab to open it in the Visual Designer.

12.5. Hold down the Control key and select the Drop Down List component. Right-click the selection and choose Bind to Data from the pop-up menu.
12.6. Add the code shown below to the ApplicationBean1.java – this method changes the content of the books_array, binded with the Drop Down List component of BooksApplication fragment.
12.7. It opens the Bind to Data dialog. From the Bind to an Object tab, select ApplicationBean1 >books_. Click OK.
12.8. Add the code shown below to the ApplicationBean1.java

```java
public void preparebooks(ArrayList<String> books) {
    if (books == null) {
        books = null;
        return;
    }
    int amount = books.size();
    if (amount > 0) {
        Option help[] = new Option[amount];
        Iterator<String> iterator = books.iterator();
        int i = 0;
        while (iterator.hasNext()) {
            help[i++] =
                new Option(Integer.toString(i), iterator.next());
        }
        books = help;
    } else
        books = new Option[0];
}
```
12.9. Add the code shown below to the ApplicationBean1.java

```java
public void add_book(String data1[], String data2[]) {
    preparebooks(facade.add_book(data1, data2).getbooks());
}
```
13. Events handling for finding the accessible book – an implementation of the third use case
13.1. Add the code shown below to the ApplicationBean1.java

```java
private Option books_[] = new Option[0];

public Option[] getBooks() {
    return books_
}

public void setBooks_(Option[] books) {
    books_ = books;
}

public void prepare_books(int number) {
    if (!facade.getTitle_books().isEmpty()) {
        preparebooks(facade.getTitle_books().get(number).getbooks());
    }
}

public void prepare_books(String data[]) {
    ...
}

public void preparebooks(ArrayList<String> books) {
    ...
}
```
13.2. Add the code shown below to the AccessibleBook.java

```java
@Override
public void prerender() {
    Menu menudiv = (Menu) getBean("Menu");
    Hyperlink link4 = menudiv.getHyperlink4();
    link4.setDisabled(true);
    FormPeriod bookdiv = (FormPeriod) getBean("FormPeriod");
    bookdiv.refresh_form();
}
```
13.3. Add the code shown below to the FormPeriod.java

```java
public void refresh_form() {
    booksApplicationPane.setRendered(true);
    period.setText("");
}

public String form_book() {
    // TODO: Process the action. Return value is case name where null will return to the sql
    String what_period;
    if (period.getText().equals("")) {
        what_period = "0";
    } else {
        what_period = (String) period.getText();
    }
    String data = what_period;
    return data;
}
```
13.4. Add the code shown below to the AccessibleBook.java

```java
public String button1_action() {
    // TODO: Process the action. Return value is a navigation
    // case name where null will return to the same page.
    TitlesApplication titlesdiv = (TitlesApplication) getBean("TitlesApplication");
    String data1[] = titlesdiv.select_title();
    FormPeriod bookdiv = (FormPeriod) getBean("FormPeriod");
    String data2 = bookdiv.form_book();
    if (data1 != null && data2 != null) {
        getApplicationBean1().search_accessible_book(data1, data2);
    }
    return null;
}
```
13.5. Add the code shown below to the ApplicationBean1.java

```java
public void search_accessible_book(String[] data1, String data2) {
    Object search_book = facade.Search_accessible_book(data1, data2);
    if (search_book != null) {
        Option help[] = new Option[1];
        help[0] = new Option("1", search_book.toString());
        books = help;
    } else {
        books = new Option[0];
    }
}
```
14. Build, deploy and run the project
Example of multilayer application
Example of multilayer application

Main page
Add titles in application
Add books in application
Check an accessible book of a select title

Title *: 1 1
Author *: 2 2
ISBN *: 3 3
Publisher *: 4 4
Actor: 5 5

Title: 1 Author: 1 ISBN: 1 Publisher: 1
Example of multilayer application

Main page
Add titles in application
Add books in application
Check an accessible book of a select title

<table>
<thead>
<tr>
<th>Title *</th>
<th>Author</th>
<th>ISBN *</th>
<th>Publisher *</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add title
Example of multilayer application

Main page
Add titles in application
Add books in application
Check an accessible book of a select title

Title *
Author *
ISBN *
Publisher *
Actor

System Messages
- form:FormTitle:title: Validation Error: Value is required.
- form:FormTitle:author: Validation Error: Value is required.
- form:FormTitle:ISBN: Validation Error: Value is required.
- form:FormTitle:publisher: Validation Error: Value is required.

Title: 1 Author: 1 ISBN: 1 Publisher: 1
Example of multilayer application

Main page
Add titles in application
Add books in application
Check an accessible book of a select title

Number: 3
Period:

Title: 1_1 Author: 2_2 ISBN: 3_3 Publisher: 4_4 Actor: 5_5 Number: 1 Period: Wed Feb 23 13:32:40 CET 2011
Title: 1_1 Author: 2_2 ISBN: 3_3 Publisher: 4_4 Actor: 5_5 Number: 1 Period: Wed Feb 23 13:32:40 CET 2011
Title: 1_1 Author: 2_2 ISBN: 3_3 Publisher: 4_4 Actor: 5_5 Number: 2
It is the view after providing the incorrect format of the number. It is the result of applying the LongRangeValid component to the number component.
Example of multilayer application

Main page
Add titles in application
Add books in application
Check an accessible book of a select title

Period
Title: 1 Author: 1 ISBN: 1 Publisher: 1
Title: 1 Author: 1 ISBN: 1 Publisher: 1
Title: 1 Author: 1 ISBN: 1 Publisher: 1 Number: 1 Period: Sat Feb 19 03:09:45 CET 2011
Title: 1 Author: 1 ISBN: 1 Publisher: 1 Number: 1 Period: Sat Feb 19 03:09:45 CET 2011
Title: 1 Author: 1 ISBN: 1 Publisher: 1 Number: 2
We can set the period for searching the first accessible book of the chosen title, that will be available after a specified period.
Selected the first book of the chosen title, that will be available after a specified period.