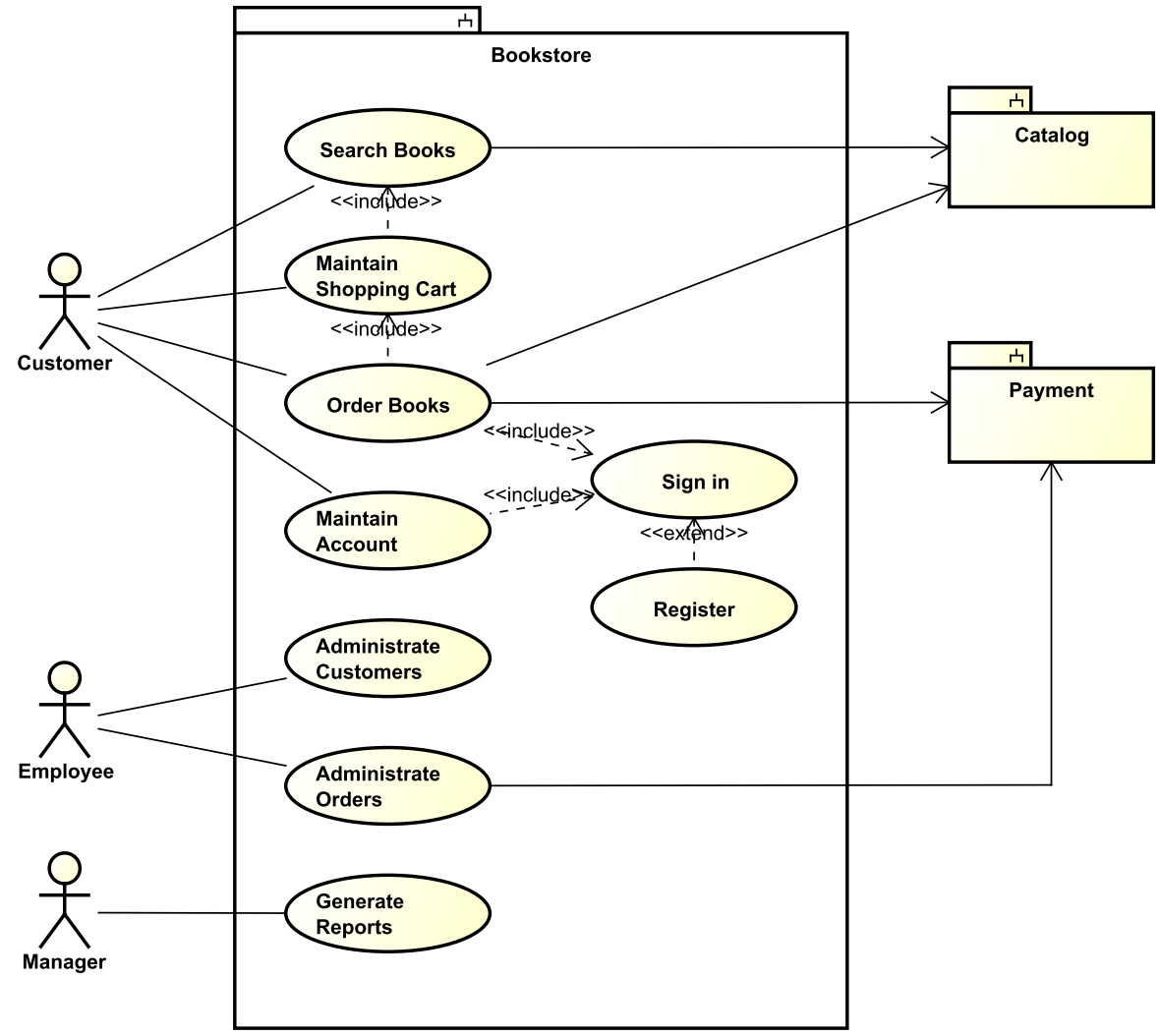
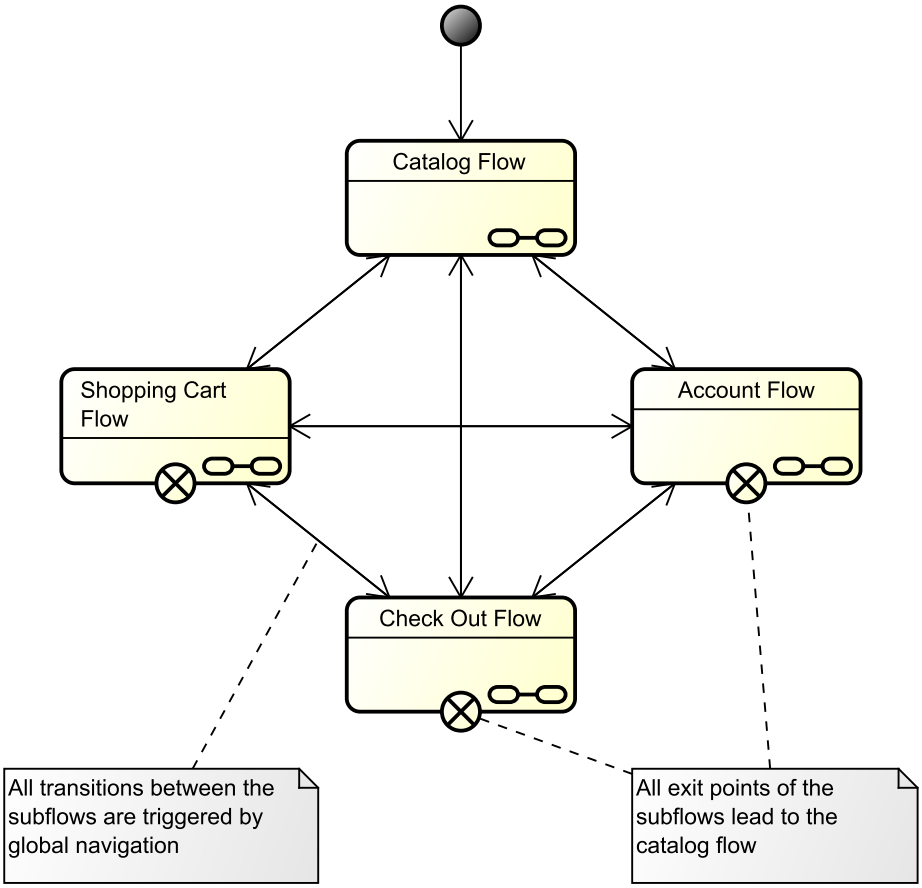
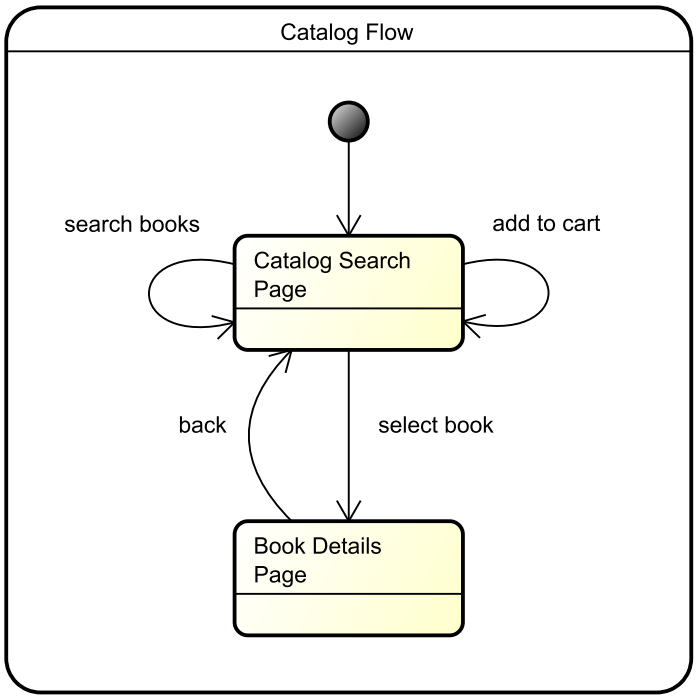
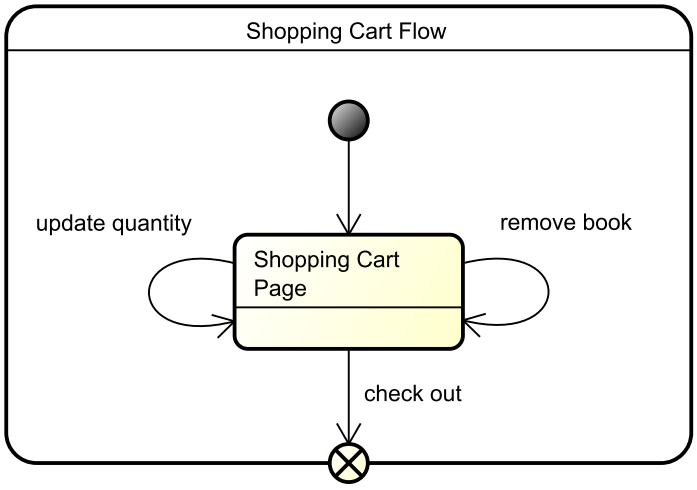
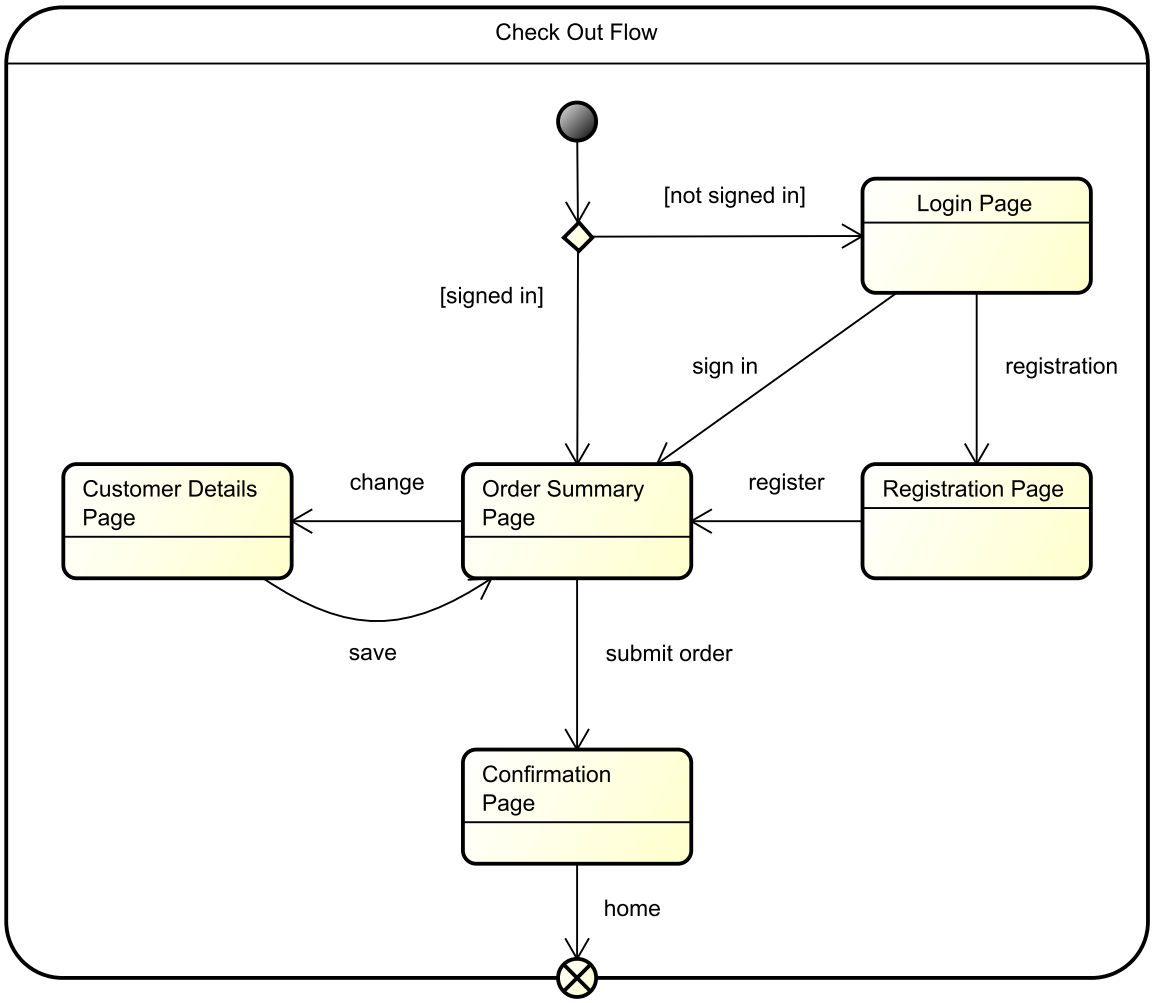
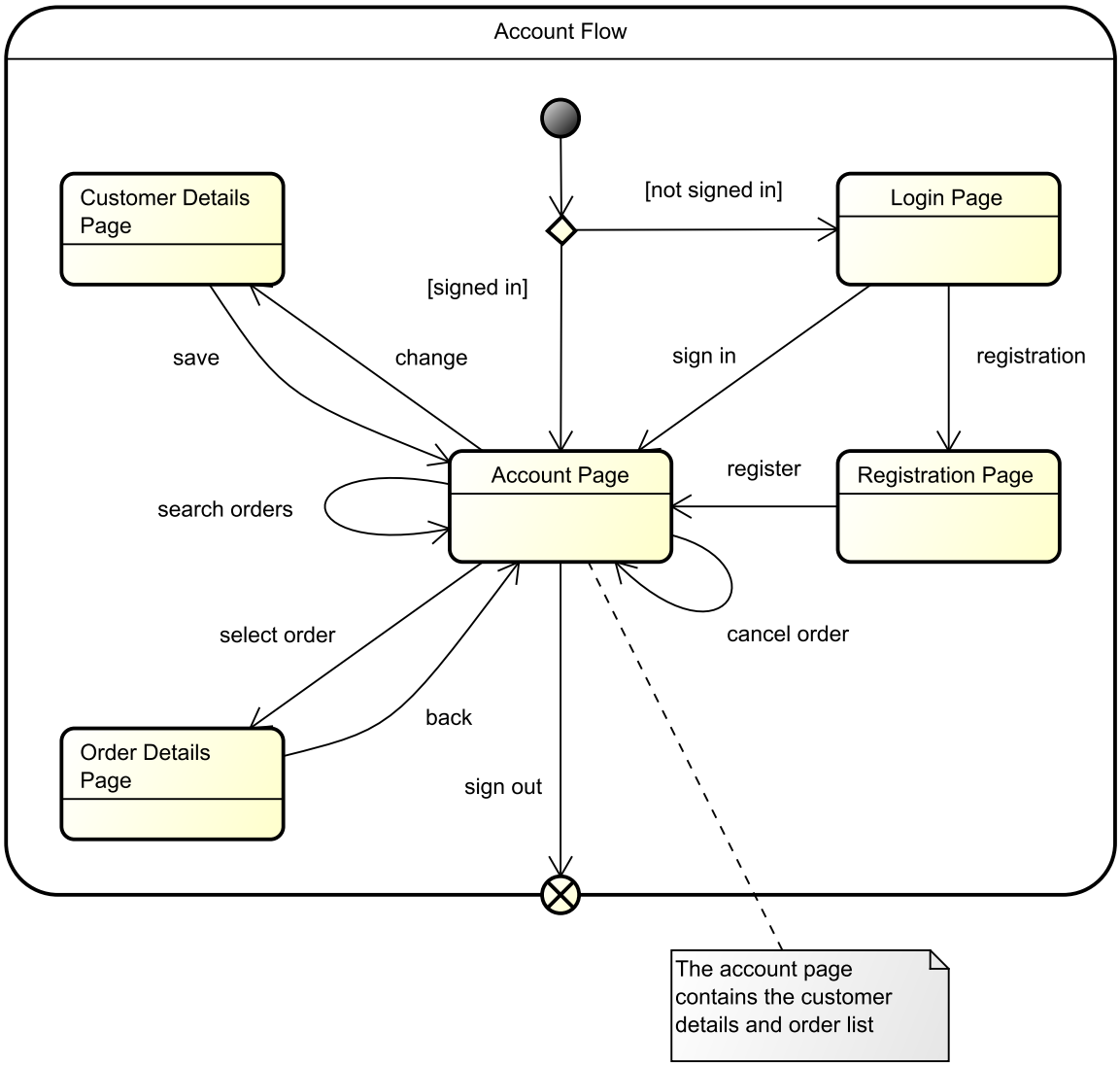

Bookstore
Search Books
Maintain Shopping Cart
Order Books
Maintain Account
Sign in
Register
Administrate Customers
Administrate Orders
Generate Reports
<<include>>
<<include>>
<<include>>
<<include>>
<<extend>>
Catalog
Payment
Customer
Employee
Manager

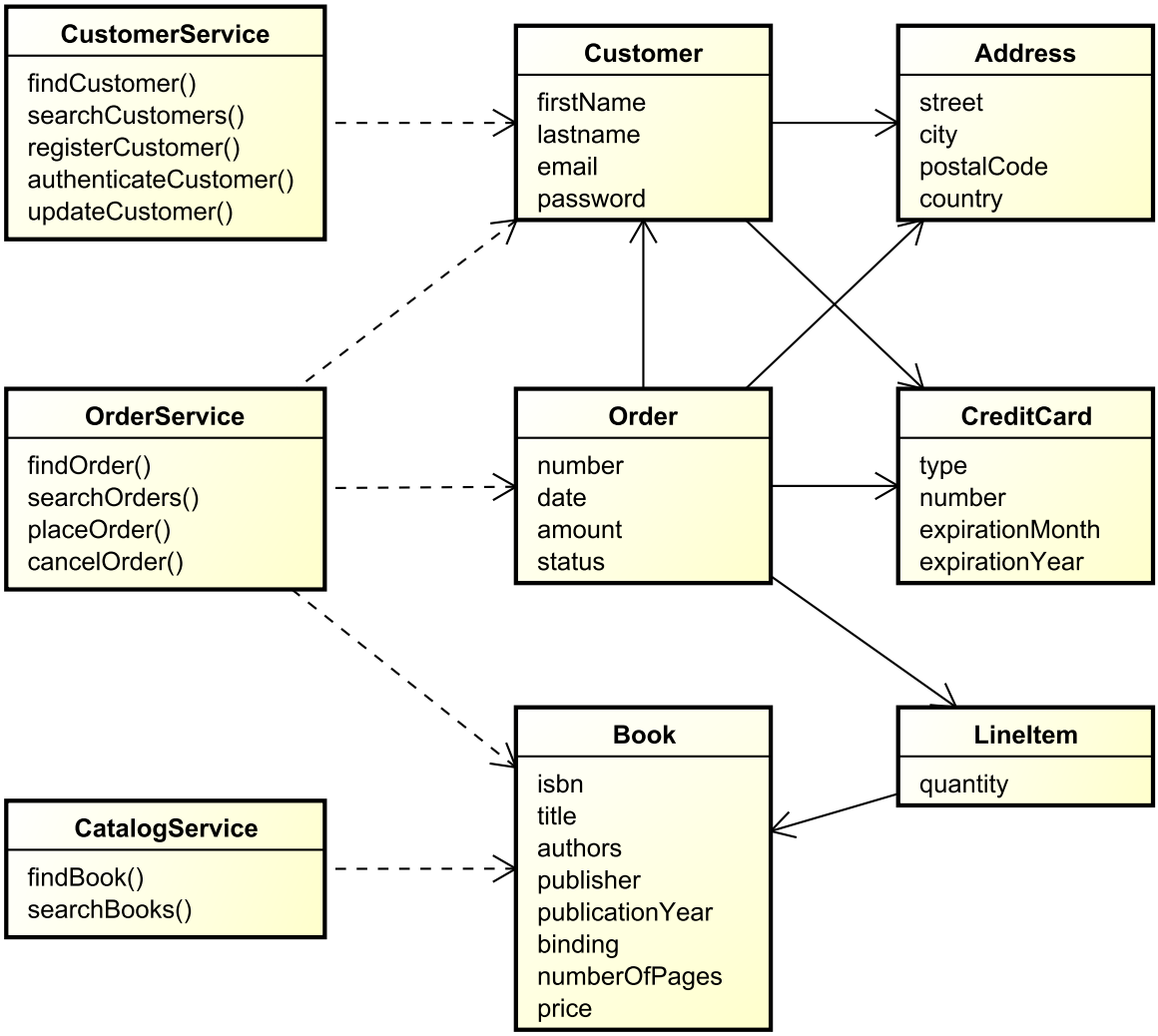

Catalog Flow
Shopping Cart Flow
Account Flow
Check Out Flow
All transitions between the subflows are triggered by global navigation
All exit points of the subflows lead to the catalog flow


Catalog Flow
search books
add to cart
Catalog Search Page
back
select book
Book Details Page


Shopping Cart Flow
update quantity
remove book
Shopping Cart Page
check out


Check Out Flow
[not signed in]
Login Page
[signed in]
sign in
registration
Customer Details Page
change
Order Summary Page
register
Registration Page
save
submit order
Confirmation Page
home


Account Flow
Customer Details Page
[not signed in]
Login Page
[signed in]
save
change
sign in
registration
Account Page
register
Registration Page
search orders
select order
cancel order
back
Order Details Page
sign out
The account page contains the customer details and order list

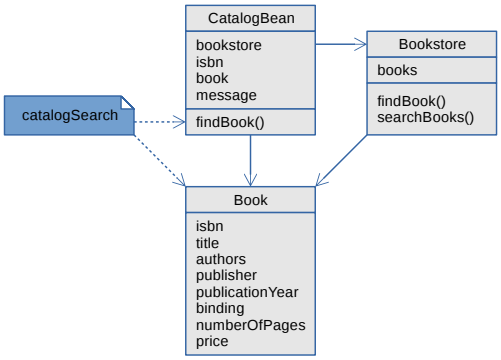

CustomerService
findCustomer()
searchCustomers()
registerCustomer()
authenticateCustomer()
updateCustomer()
Customer
firstName
lastname
email
password
Address
street
city
postalCode
country
OrderService
findOrder()
searchOrders()
placeOrder()
cancelOrder()
Order
number
date
amount
status
CreditCard
type
number
expirationMonth
expirationYear
Book
isbn
title
authors
publisher
publicationYear
binding
numberOfPages
price
LineItem
quantity
CatalogService
findBook()
searchBooks()

JavaServer Faces

Exercise: Managed Beans

Description

The objective of this exercise is to implement a managed bean and a JSF page which allow customers to find a book.



Documentation

Assignment

1. Extract the exercise archive
2. Implement the managed bean
   `src/main/java/org/books/presentation/CatalogBean.java`
   which has an action method to find a book by ISBN number
3. Write the JSF page
   `src/main/webapp/catalogSearch.xhtml`
   which has an input form and displays the details of a found book
4. Deploy the web application and test it

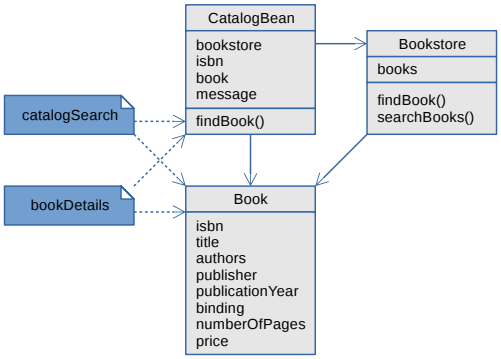
Solution

JavaServer Faces

Exercise: Navigation

Description

The objective of this exercise is to implement two JSF pages and to define a navigation between them.



Documentation

Assignment

1. In the managed bean
   `src/main/java/org/books/presentation/CatalogBean.java`
   define an outcome of the find method such that the book details page is shown next
2. Write the JSF page
   `src/main/webapp/bookDetails.xhtml`
   which displays the details of a found book
3. Deploy the web application and test it

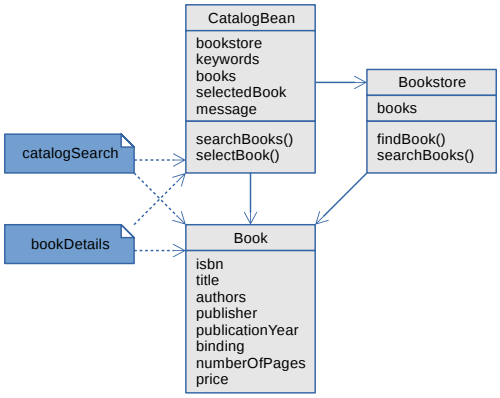
Solution

JavaServer Faces

Exercise: Standard Components

Description

The objective of this exercise is to implement a data table which contains the results of a book search.



Documentation

Assignment

1. In the managed bean
   `src/main/java/org/books/presentation/CatalogBean.java`
   implement an action method to search for books by keywords and a method to select an entry of the search results
2. In the JSF page
   `src/main/webapp/catalogSearch.xhtml`
   add a data table to display the search results and use an action method parameter to select a book
3. Deploy the web application and test it

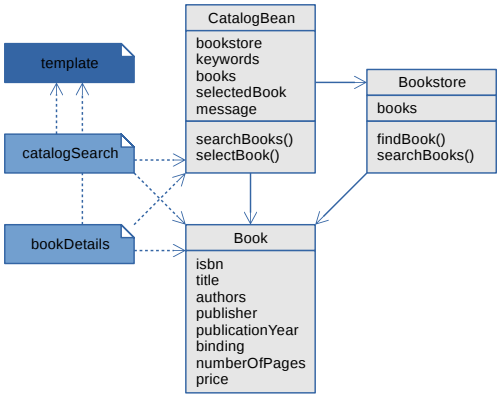
Solution

JavaServer Faces

Exercise: Facelets Templating

Description

The objective of this exercise is to unify the layout of the JSF pages by using a Facelets template.



Documentation

Assignment

1. Write a Facelets template
   `src/main/webapp/template.xhtml`
   which contains a page header and a variable content region
2. Rewrite the JSF pages
   `src/main/webapp/catalogSearch.xhtml`
   `src/main/webapp/bookDetails.xhtml`
   as client pages of the template by defining the content region
3. Deploy the web application and test it

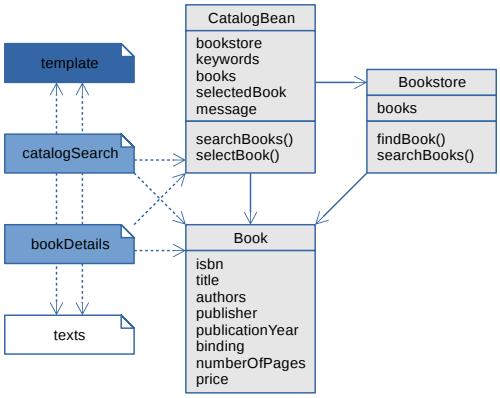
Solution

JavaServer Faces

Exercise: Internationalization

Description

The objective of this exercise is to internationalize the bookstore application.



Documentation

Assignment

1. Write the resource bundle
   `src/main/resources/bundles/texts_en.properties`
   `src/main/resources/bundles/texts_de.properties`
   which contains the English and German texts
2. In the JSF pages
   `src/main/webapp/catalogSearch.xhtml`
   `src/main/webapp/bookDetails.xhtml`
   replace the literal texts by corresponding value expressions
3. Write a JSF configuration file
   `src/main/webapp/WEB-INF/faces-config.xml`
   which declares the supported locales and the resource bundle of localized texts
4. Deploy the web application and test it

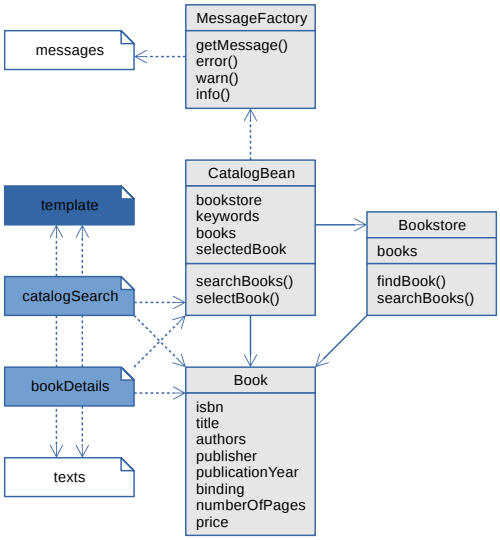
Solution

JavaServer Faces

Exercise: Messaging

Description

The objective of this exercise is to display error messages using a message bundle.



Documentation

Assignment

1. Write the message bundle
   `src/main/resources/bundles/messages_en.properties`
   `src/main/resources/bundles/messages_de.properties`
   which contains the English and German error messages
2. In the managed bean
   `src/main/java/org/books/presentation/CatalogBean.java`
   add an error message to the faces context if the book search fails
3. In the Facelets template
   `src/main/webapp/template.xhtml`
   display the error message at the bottom of the page
4. In the JSF configuration file
   `src/main/webapp/WEB-INF/faces-config.xml`
   declare the application message bundle
5. Deploy the web application and test it

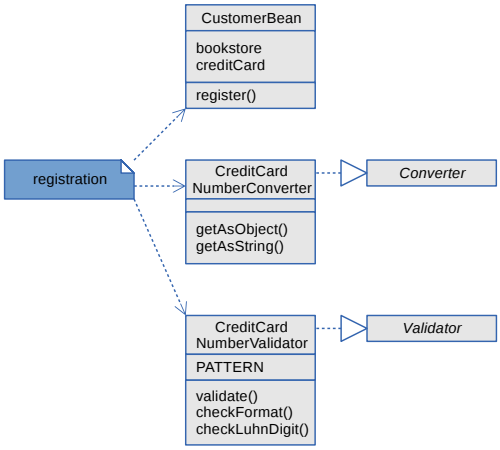
Solution

JavaServer Faces

Exercise: Validators and Converters

Description

The objective of this exercise is to implement a custom converter and validator for credit card numbers.



Documentation

Assignment

1. Extract the exercise archive
2. Implement the converter
   `src/main/java/org/books/presentation/CreditCardNumberConverter.java`
   which removes spaces from credit card numbers
3. Implement the validator
   `src/main/java/org/books/presentation/CreditCardNumberValidator.java`
   which validates the format and the check digit of credit card numbers according to the Luhn alogrithm
4. In the JSF page
   `src/main/webapp/registration.xhtml`
   insert the credit card converter and validator
5. Add appropriate error messages to the application message bundle
   `src/main/resources/bundles/messages_en.properties`
   `src/main/resources/bundles/messages_de.properties`
6. Deploy the web application and test it

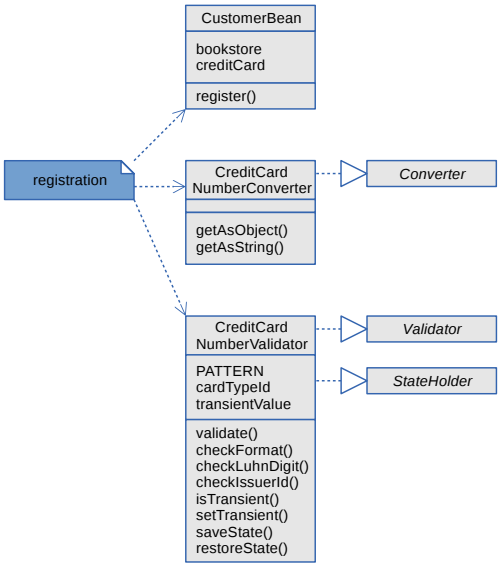
Solution

JavaServer Faces

Exercise: Custom Tags

Description

The objective of this exercise is to implement a validator which allows to validate the numbers of credit cards according to their type.



Documentation

Assignment

1. Add the property cardTypeId to the validator
   `src/main/java/org/books/presentation/CreditCardNumberValidator.java`
   which references the input component of the card type and use the card type to validate the card number
2. Write the tag library descriptor
   `src/main/webapp/WEB-INF/books.taglib.xml`
   which defines the validator tag
3. Register the tag library in the application deployment descriptor
   `src/main/webapp/WEB-INF/web.xml`
4. Use the validator tag with the cardTypeId attribute in the JSF page
   `src/main/webapp/creditCard.xhtml`
5. Deploy the web application and test it

Solution

JavaServer Faces

Exercise: Composite Components

Description

The objective of this exercise is to implement a composite component for the input of credit cards.

Assignment

1. Implement the composite component
   `src/main/webapp/resources/components/inputCreditCard.xhtml`
   which consists of input elements for the credit card type, number and expiration date
2. In the JSF page
   `src/main/webapp/registration.xhtml`
   use the composite component to get the credit card information of a customer
3. Deploy the web application and test it

Solution