Mapping use cases to Struts

This Tutorial illustrates how to map use cases from application logic to Struts. We will use the Struts config designer of MyEclipse to define the Struts configurations very fast. As sample application, we use a library.

Generals

Author:
Sascha Wolski
Sebastian Hennebrueder
http://www.laliluna.de/tutorials.html – Tutorials for Struts, EJB, xdoclet and eclipse.

Date:
January, 25 2005

Software:
Eclipse 3.0
MyEclipse Enterprise Workbench 3.8
Struts Framework 1.1, 1.2

Requirements

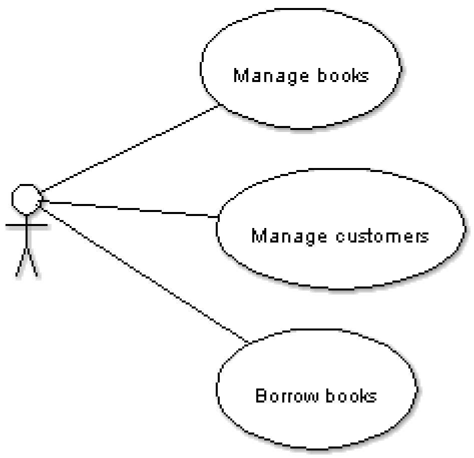
The tutorial http://www.laliluna.de/myeclipse-visual-struts-development-tutorial.html is required.

PDF Version

http://www.laliluna.de/assets/tutorials/mapping-use-cases-struts-tutorial-en.pdf

Business processes

Use cases for a library application can be add, edit and removal of books, borrow books and managing customer data.



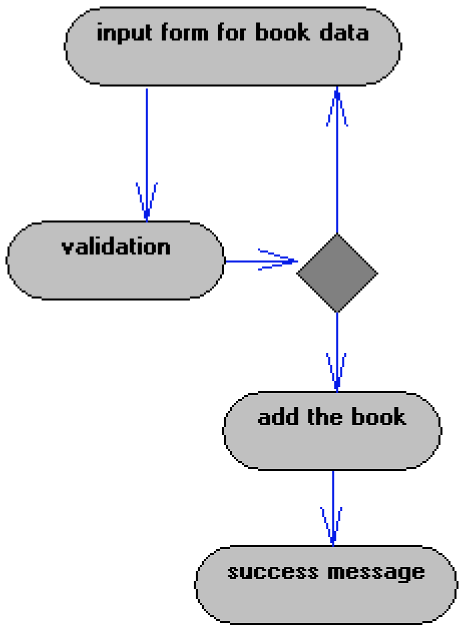
Books administration

We have three detailed use cases: add, edit and remove.

Add books

A form to input the book data is shown to the user. If the user submits the form validation of the data is accomplished. If the validation is not successful error messages will displayed on the input page, otherwise the book will be added and a success page will be displayed.

The following activity diagram shows the course of the process.



Struts-config

In order to build the business processes in struts you need a form bean, a action and two JSP files. The structure in the design mode of the struts config file looks like the following.

Actions

- *bookAdd*

FormBeans

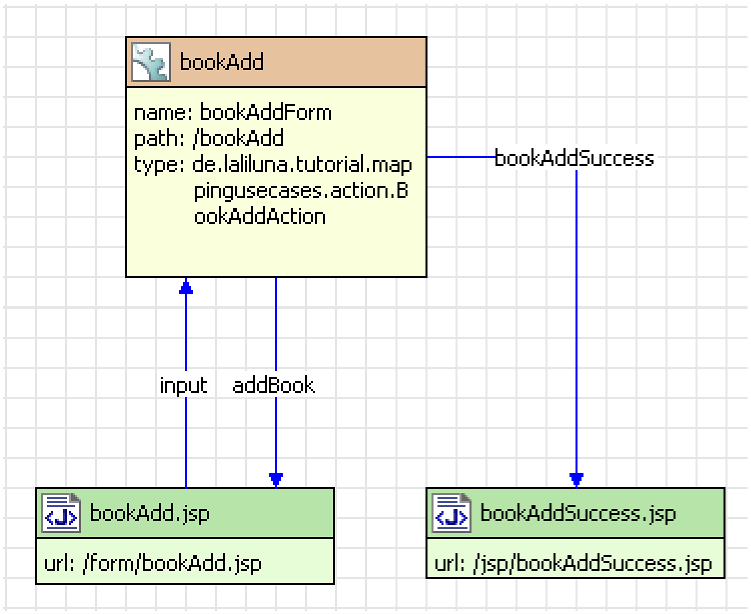
- *bookAddForm*

JSPs

- *bookAdd.jsp*
- *bookAddSuccess.jsp*

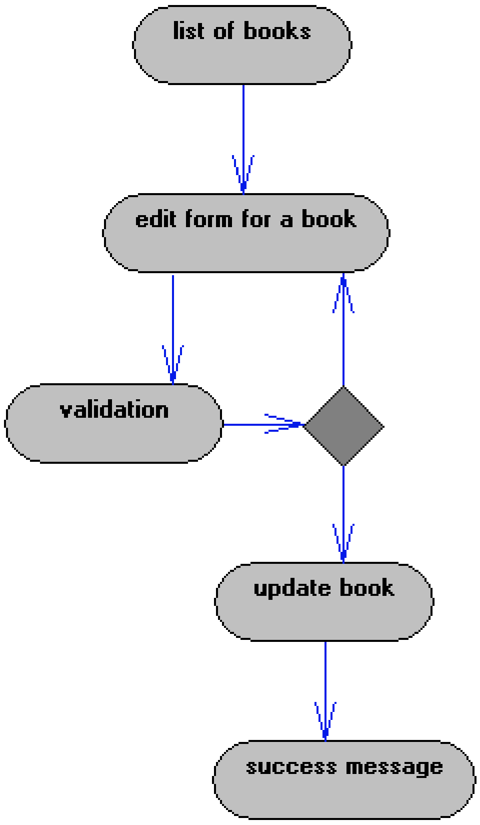
Forwards of the action bookAdd

- *bookAddSuccess* to *bookAdd.jsp*
- *addbook* to *bookAddSuccess.jsp*



Edit books

The user chooses a book to edit from a list of all books. He is forwarded to a form which contains the data of the book. At this point there is a option to abort this process and go back to the list of all books. If the user continues the process he can modify the data and submit the form. After submitting the form validation of the data is accomplished. If the validation is not correct the user will be forwarded to the edit page where the error messages are displayed, otherwise the user will be forwarded to a success page.



Struts-config

You need a form bean, an action and a JSP file for the list of all books and an other form bean, an action and a JSP file to create the edit page where you can modify the book data. The structure looks like the following.

Actions

- *bookEdit*
- *bookList*

FormBeans

- *bookEditForm*
- *bookListForm*

JSPs

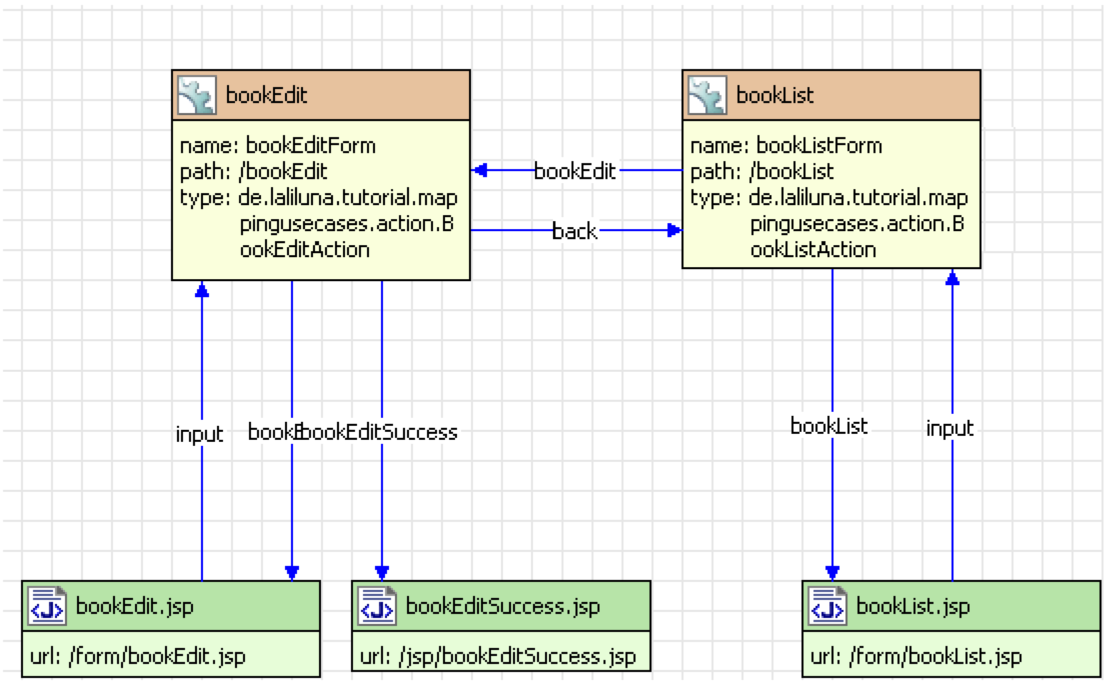
- *bookEdit.jsp*
- *bookEditSuccess.jsp*
- *bookList.jsp*

Forwards of the action bookList

- *bookList* to *bookList.jsp*
- *bookEdit* to action *bookEdit*

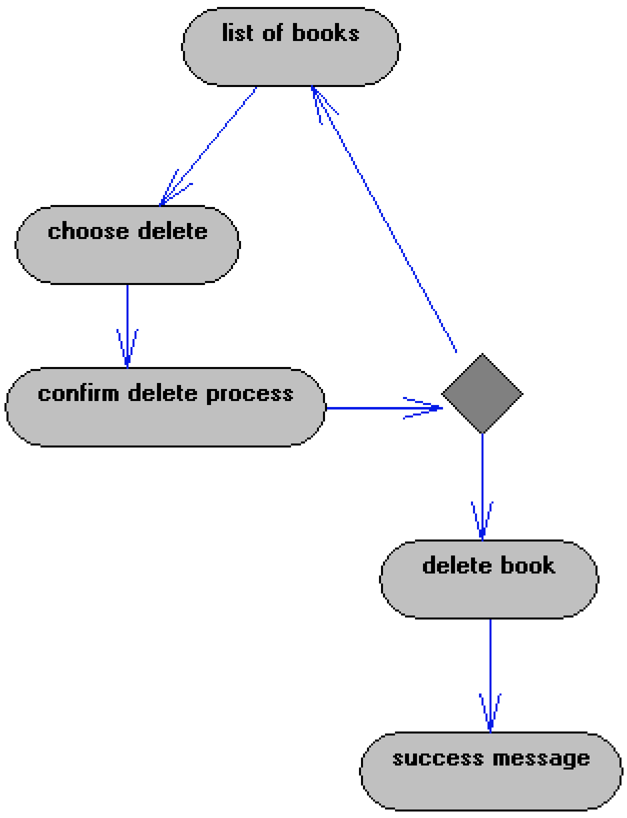
Forwards of the action bookEdit

- *bookEdit* to *bookEdit.jsp*
- *bookEditSuccess* to *bookEditSuccess.jsp*
- *back* to action *bookList*



Delete books

The user can choose a book to delete from the list of all existing books and will be forwarded to a confirm page, where he confirms the delete process. If the user confirms the delete process the book will be deleted, otherwise he will be forwarded back to the list of books.



Struts-config

To build the list of all books you need a form bean, an action and a JSP file. You can use the action which lists all books from the edit process, add only a new forward. You also need two JSP files, a form bean and a action to build the delete process.

Actions

- *bookList*
- *deleteBook*

FormBeans

- *bookListForm*
- *deleteBookForm*

JSPs

- *bookList.jsp*
- *deleteBookSuccess.jsp*
- *deleteBook.jsp*

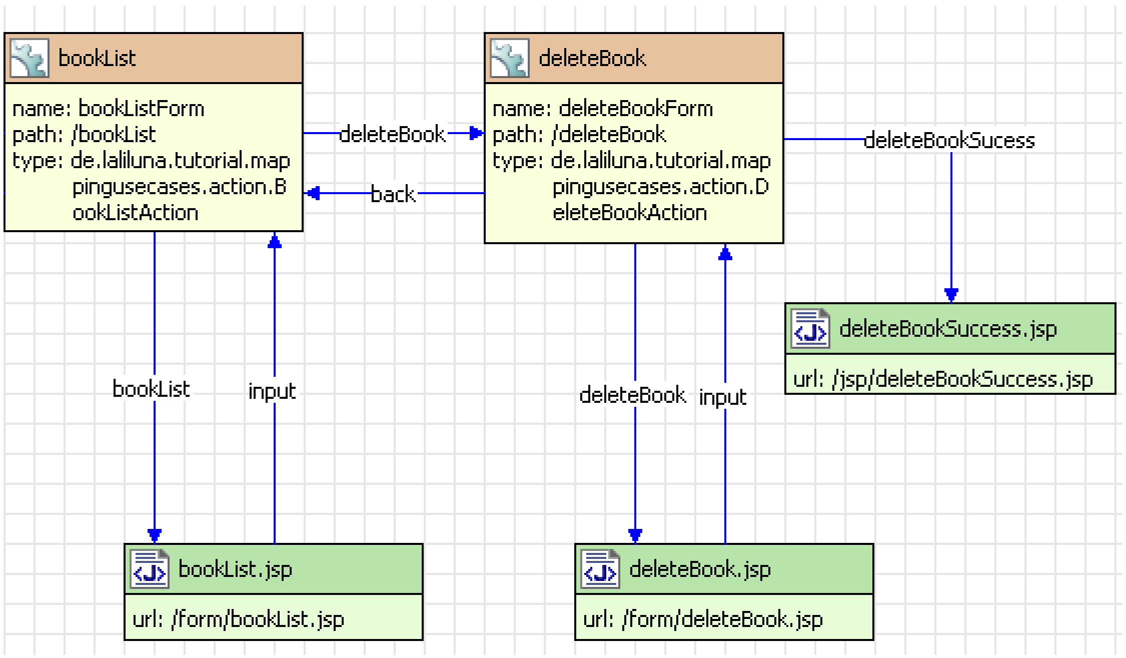
Forwards of the action bookList

- *deleteBook* to action *deleteBook*
- *bookList* to *bookList.jsp*

Forwards of the action deleteBook

- *back* to action *bookList*

- *deleteBook* to *deleteBook.jsp*
- *deleteBookSuccess* to *deleteBookSuccess.jsp*

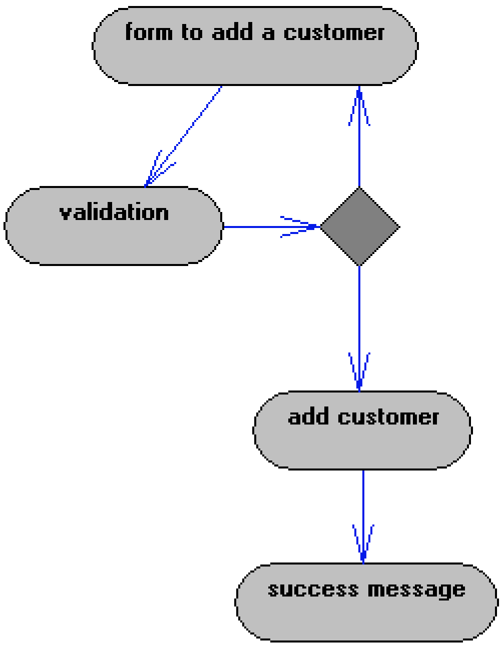


Customers administration

On the customer administration we have three detail processes, add, edit and remove of a customer.

Add customers

A form page where you can at the customer data will be shown to the user. After he added the data and submitted the form a validation is accomplished. If the validation is not successfully the form page with the error messages will be displayed, otherwise a success page with information that the user is successfully added will be shown to the user.



Struts-config

In the config design mode the description of the process above looks like the following. You need a form bean, an action and two JSP files. One jsp file displays the form and the other the success information.

Actions

- *customerAdd*

FormBean

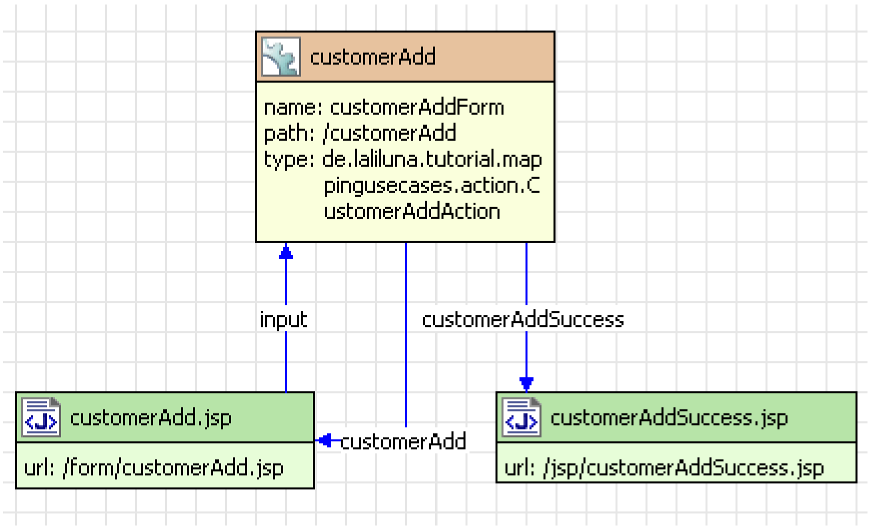
- *customerAddForm*

JSPs

- *customerAdd.jsp*
- *customerAddSuccess.jsp*

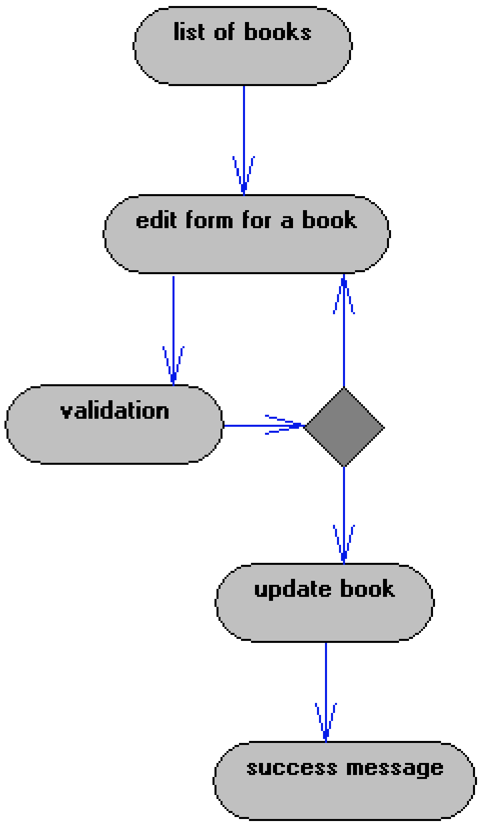
Forwards of the action customerAdd

- *customerAdd* to *customerAdd.jsp*
- *customerAddSuccess* to *customerAddSuccess.jsp*


customerAdd
name: customerAddForm
path: /customerAdd
type: de.laliluna.tutorial.mappingusecases.action.CustomerAddAction
input
customerAddSuccess
customerAdd
customerAdd.jsp
url: /form/customerAdd.jsp
customerAddSuccess.jsp
url: /jsp/customerAddSuccess.jsp

Edit customers

First the user chooses a customer which he wants to edit from a list of all customers. You may recognize that we use the same way as before with the book edit process. After the user selected a customer, a form page will be shown to him, which contains the data of the selected customer. The process can be aborted and the user goes back to the list of customers. After the data has be modified and the form has been submitted, a validation of the data is accomplished. If an error occurs during the validation, the user will be forwarded back to the edit page where the error messages will be displayed. If the validation is successful a success page will be shown to the user, that the customer was added successfully.



Struts-config

You need an action, a form bean and a JSP file for the list of books. The edit process needs also an action, a form bean, but two JSP files.

Actions

- *customerList*
- *customerEdit*

FormBeans

- *customerListForm*
- *customerEditForm*

JSPs

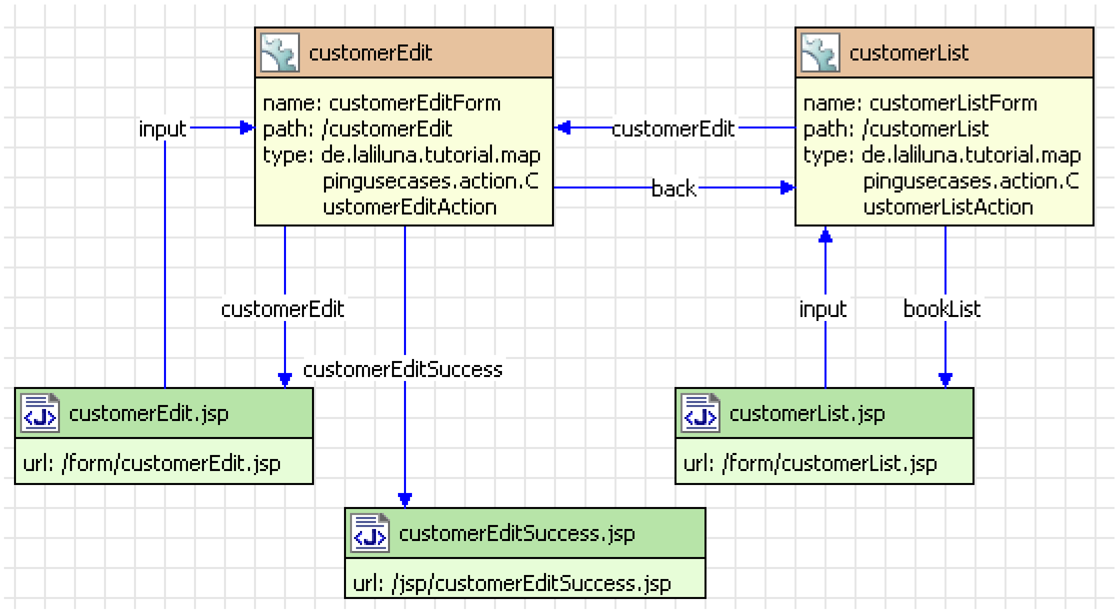
- *customerList.jsp*
- *customerEdit.jsp*
- *customerEditSuccess.jsp*

Forwards of the action customerList

- *customerList* to *customerList.jsp*
- *customerEdit to action customerEdit*

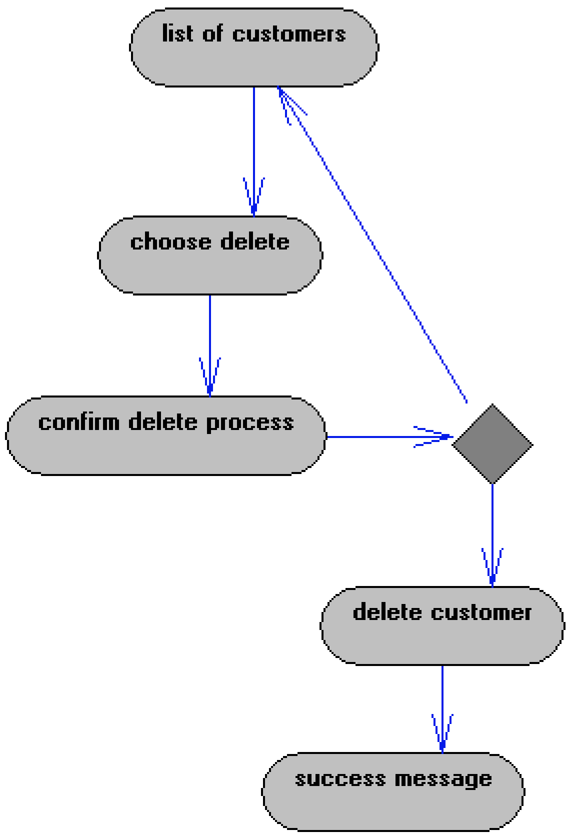
Forwards der Action customerEdit

- *customerEdit to customerEdit.jsp*
- *customerEditSuccess* to *customerEditSuccess.jsp*
- *back* to action *bookList*



Delete customers

The process is similar to the delete process of a book. On the list of customers the user can choose which customer he wants to delete. After this he is forwarded to a confirm page where the user must confirm with yes or no the delete process. If he chooses yes the customer will be deleted, otherwise he goes back to the list of customers.



Stuts-config

You can use the action, the form bean and the jsp for the list of customers of the edit process. Further you need a new action, form bean and jsp files to build the delete cofirm dialog. In the struts config design mode the structure looks like the following.

Actions

- *customerList*
- *customeDelete*

FormBean

- *customerListForm*
- *customerDeleteFrom*

JSPs

- *customerList.jsp*
- *customerDelete.jsp*
- *customerDeleteSuccess.jsp*

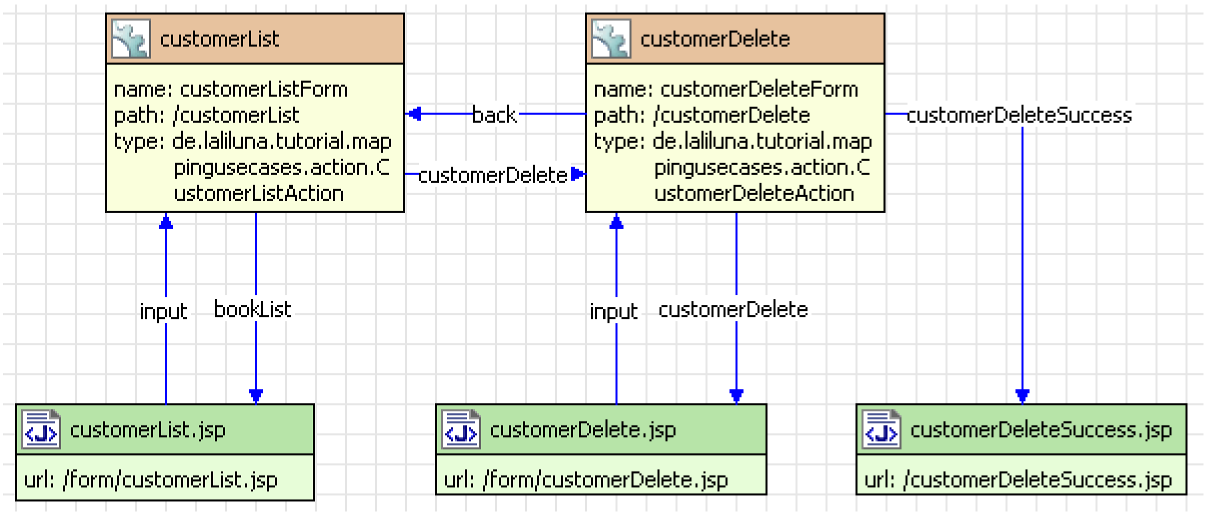
Forwards of the action customerList

- *customerList* to *customerList.jsp*
- *customerDelete* to action *customerDelete*

Forwards of the action customerDelete

- *customerDelete* to *customerDelete.jsp*

- *customerDeleteSuccess* to *customerDeleteSuccess.jsp*
- *back* to action *customerList*


customerList
name: customerListForm
path: /customerList
type: de.laliluna.tutorial.mappingusecases.action.CustomerListAction
back
customerDelete
customerDelete
name: customerDeleteForm
path: /customerDelete
type: de.laliluna.tutorial.mappingusecases.action.CustomerDeleteAction
customerDeleteSuccess
input
bookList
input
customerDelete
customerList.jsp
url: /form/customerList.jsp
customerDelete.jsp
url: /form/customerDelete.jsp
customerDeleteSuccess.jsp
url: /customerDeleteSuccess.jsp

Borrow books

In the first dialogue the user will ask to input the customer number in a form. After he submitted the form a validation of the customer number is accomplished. In the case of invalid validation the user will be forwarded back to the input page and the errors will be displayed.

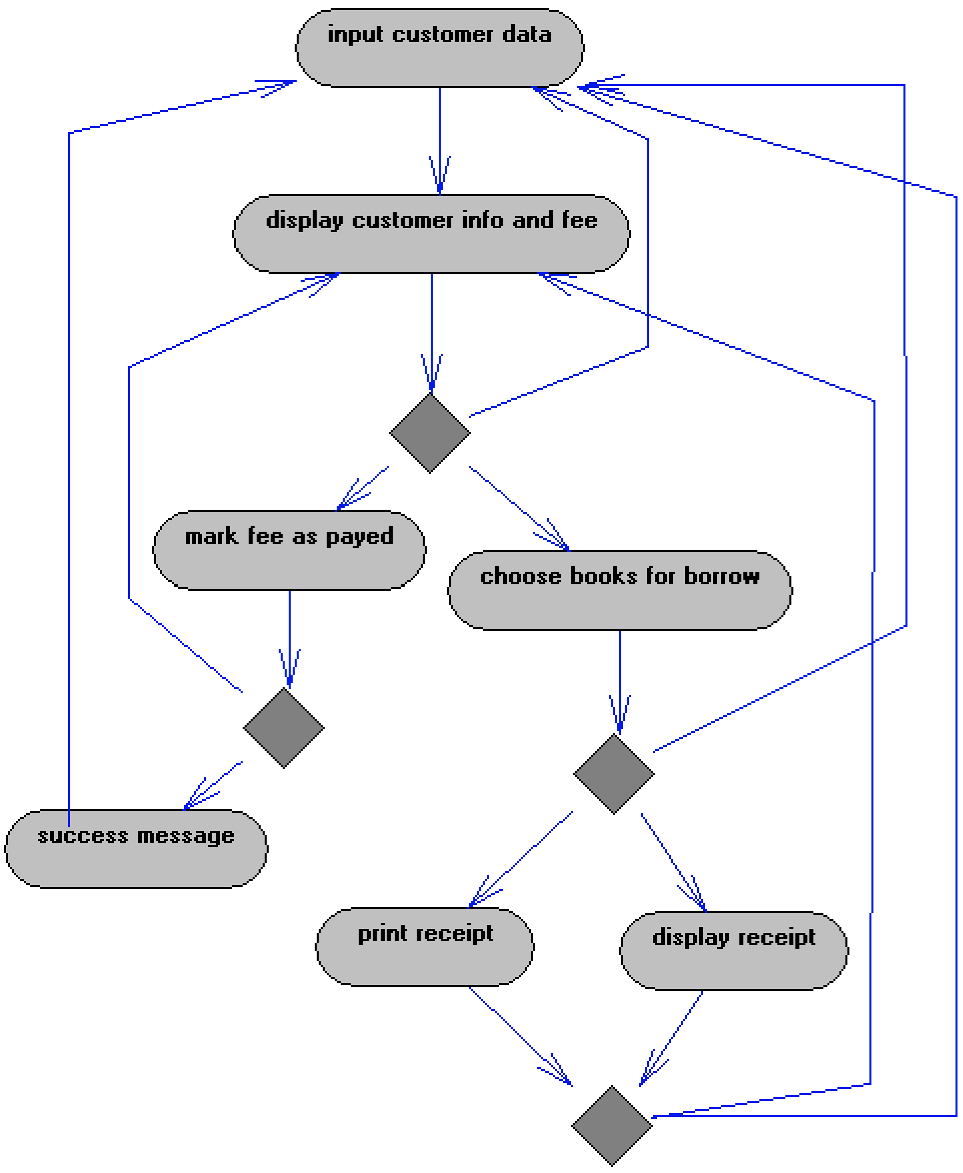
If there are no errors during the validation the user is forwarded to the customer page, where the customer data and the borrow fees will be displayed. If the customer can pay the fees, the user can go to a fees dialogue, otherwise he is forwarded to the borrow dialogue. The process can be aborted by the user and he goes back to the input dialogue for the customer number.

Fee dialogue

On the fee dialogue the existing fees will displayed. The user can abort this process and go back to the customer info page. The existing fees can be marked as paid and after the user confirmed the dialogue a success message will be shown on a new page. The fees are marked as paid, will be displayed. After this step the user will be forwarded to the customer info page.

Borrow Dialogue

On the borrow dialogue the user can search and choose the books, which are borrowed by the customer. If the user confirms the dialogue he is forwarded to the receipt page. He can print the receipt. In the next step the user can choose if he wants to return to the input page of the customer number or back to the customer info page.



Struts-config

You need two actions, two form beans, two JSP files to build the input page for the customer number and the customer info page.

Fee dialogue

You need one action, one form bean and two JSP files for the fee dialogue.

Borrow dialogue

You need two form beans, two actions and three JSP files.

Actions

- *selectCustomer*
- *loadCustomerData*
- *borrowBooks*
- *borrowBills*
- *borrowReceipt*

FormBeans

- *customerNumberForm*
- *customerInfoForm*
- *borrowBooksForm*
- *borrowBillsForm*
- *borrowReceiptsForm*

JSPs

- *customerNumber.jsp*
- *customerInfo.jsp*
- *borrowBooks.jsp*
- *borrowBills.jsp*
- *borrowBillsSuccess.jsp*
- *borrowReceipt.jsp*
- *printReceipt.jsp*

Forwards der Action selectCustomer

- *selectCustomer* auf selectCustomr*.jsp*
- *loadCustomerData* auf Action *loadCustomerData*

Forwards der Action *loadCustomerData*

- *loadCustomerData* auf loadCustomerData.jsp
- *borrowBooks* auf Action *borrowBooks*
- *borrowBills* auf Action *borrowBills*

Forwards der Action borrowBooks

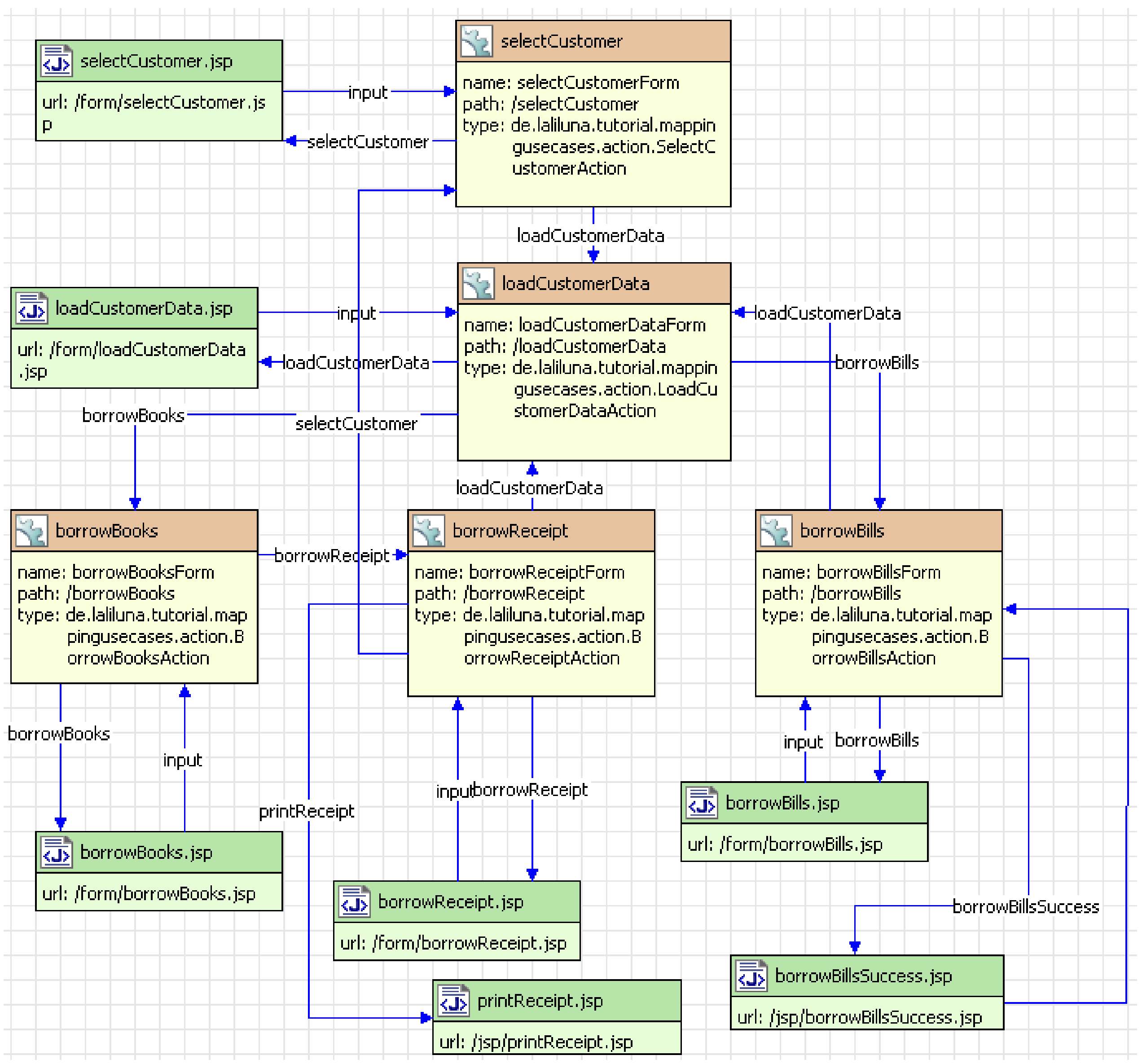
- *borrowBooks* auf *bookBooks.jsp*
- *borrowReceipt* auf Action *borrowReceipt*

Forwards der Action borrowReceipt

- *borrowReceipt* auf *borrowReceipt.jsp*
- *printReceipt* auf *printReceipt.jsp*
- *selectCustomer* auf Action *selectCustomer*
- *loadCustomerData* auf Action *loadCustomerData*

Forwards der Action borrowBills

- *borrowBills* auf *borrowBills.jsp*
- *borrowBillsSuccess* auf *borrowBillsSuccess.jsp*
- *loadCustomerData* auf Action *loadCustomerData*


selectCustomer.jsp
url: /form/selectCustomer.jsp
input
selectCustomer
selectCustomer
name: selectCustomerForm
path: /selectCustomer
type: de.laliluna.tutorial.mappingusecases.action.SelectCustomerAction
loadCustomerData
loadCustomerData.jsp
url: /form/loadCustomerData.jsp
input
loadCustomerData
loadCustomerData
name: loadCustomerDataForm
path: /loadCustomerData
type: de.laliluna.tutorial.mappingusecases.action.LoadCustomerDataAction
loadCustomerData
borrowBills
borrowBooks
selectCustomer
loadCustomerData
borrowBooks
name: borrowBooksForm
path: /borrowBooks
type: de.laliluna.tutorial.mappingusecases.action.BorrowBooksAction
borrowReceipt
borrowReceipt
name: borrowReceiptForm
path: /borrowReceipt
type: de.laliluna.tutorial.mappingusecases.action.BorrowReceiptAction
borrowBills
name: borrowBillsForm
path: /borrowBills
type: de.laliluna.tutorial.mappingusecases.action.BorrowBillsAction
borrowBooks
input
printReceipt
input
borrowReceipt
input
borrowBills
borrowBooks.jsp
url: /form/borrowBooks.jsp
borrowReceipt.jsp
url: /form/borrowReceipt.jsp
borrowBills.jsp
url: /form/borrowBills.jsp
borrowBillsSuccess
borrowBillsSuccess.jsp
url: /jsp/borrowBillsSuccess.jsp
printReceipt.jsp
url: /jsp/printReceipt.jsp