Workshop 3: A Component Based Approach, Java Server Faces

Objectives

Same as previous workshops (expose the ProductCatalogue). We need;

- GlassFish application server.
- Java Server Faces (JSF) and Facelets (possibly must add as framework in NetBeans).
- Context and Dependency Injection (CDI).
- JSF and Bean validation.
- A higher level component library for JSF (PrimeFaces, ...).

PLEASE: INSPECT CODE SAMPLES FROM THE LECTURES (ON COURSE PAGE)! EVERYTHING YOU NEED SHOULD BE THERE. WILL HOPEFULLY SAVE YOU A LOT OF TIME!

Final date: See course page

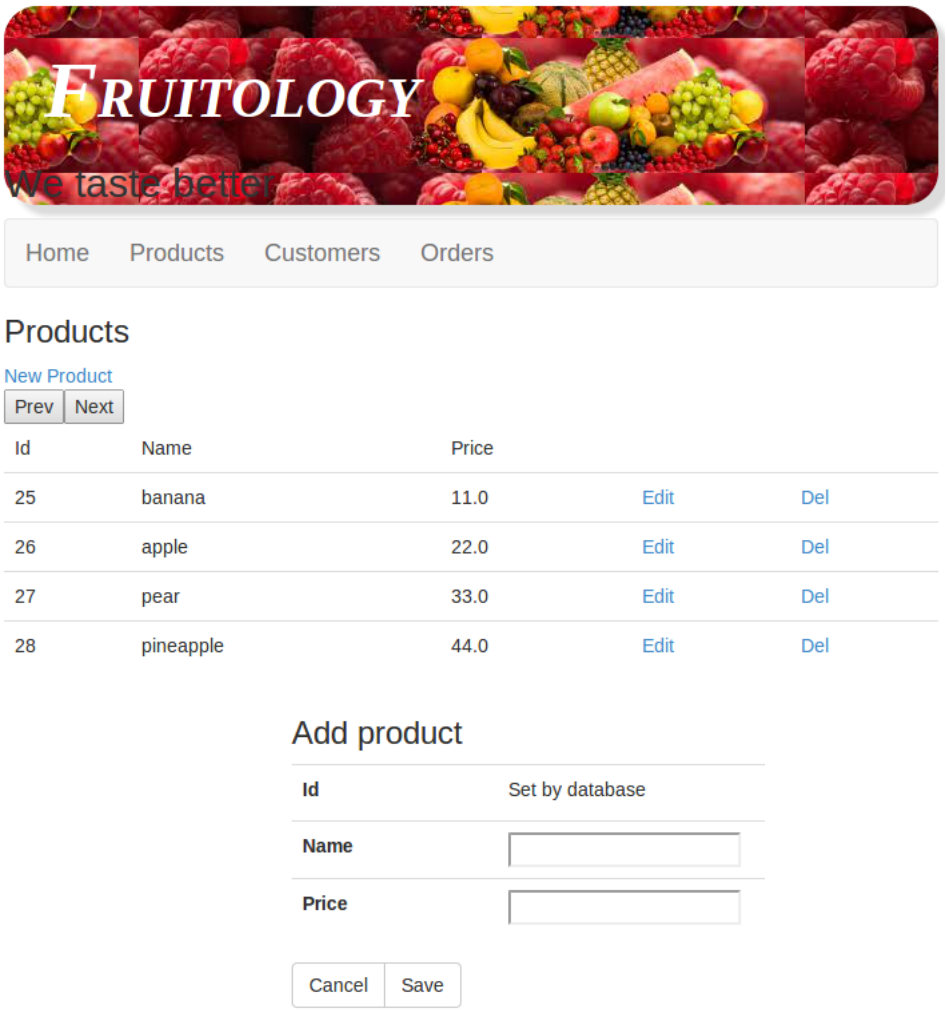
1 The JSF request cycle

Download the JSF request cycle demo from course page. Run and inspect output from GlassFish. What happens? When?

Note The URL's, to trigger the JSF-system, see <servlet-mapping> in web.xml (compare how to trigger JAX-RS).

2 GUI

The look is (should be) very similar to previous workshops (using Bootstrap).

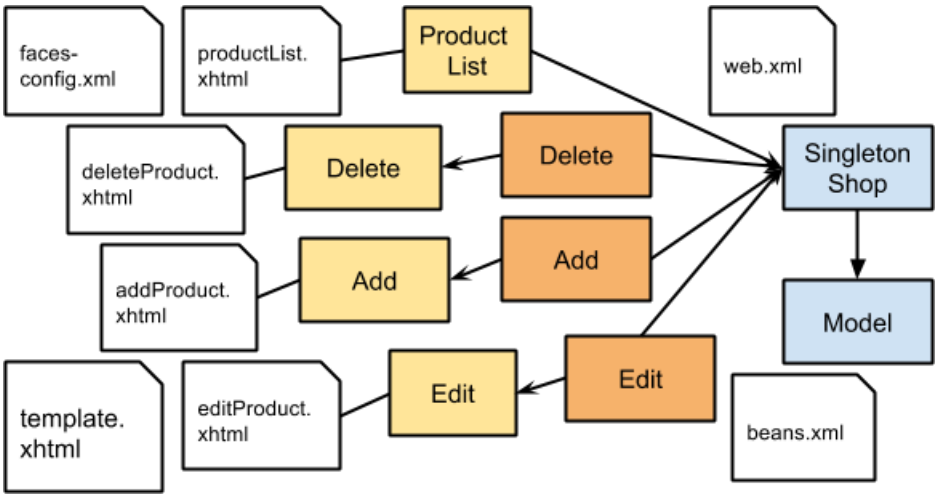


3 Design

Application design is given by the picture below. The design is a bit overkill for this small application (describes the general approach). Some classes will be very small. Arrows are associations.

- Overall page design is given by template.xhtml. Other xhtml is swapped in and out of template.
- We have one JSF page for each CRUD operation.
- Each page has a CDI managed bean (yellow) holding page data (input/output channel).
- The control layer is handled by control beans (orange). Any modification of the model is handled by this layer.

- The SingletonShop is a bean wrapping the shop model.
- faces-config.xml handles navigation (outcome from control beans). There could be other navigation (master detail).
- web.xml contains mappings to trigger JSF (JSF Servlet). Also used if implementing authorization.
- beans.xml in needed by CDI.



4 Implementation

All associations between beans should be realized by CDI injection. Final project structure in appendix.

1. Download the skeleton code from course page. It should be possible to run. There will be some navigation error messages, no problem leave for now.
2. The files to work with are: The partials xhtml-pages, the backing beans and the control beans (incl. possibly modifications of faces.config.xml)
3. Start with productList functionality i.e. productsList.xhtml and it's backing bean. Goal is to list the ProductCatalogue. Add CDI annotations as needed (select correct scope)
4. Add pagination for the list.
5. Master-Detail: Insert links for add, edit and delete pages in list page.
6. Start implementing the add page and it's beans. Add CDI annotations as needed (select appropriate scope)

7. Implement edit and delete functionality (pages very similar to add). Use view-parameters in edit and delete pages to transfer data from list page. Select scope for beans.

8. Add navigation. Prefer rule based navigation i.e. use faces-config.xml.

4.1 Validation

Add validation. Try both JSF validation and Bean validation (prefer). There are some validation messages in src/main/resources. Use!

5 PRG

Check that the PRG-pattern is implemented.

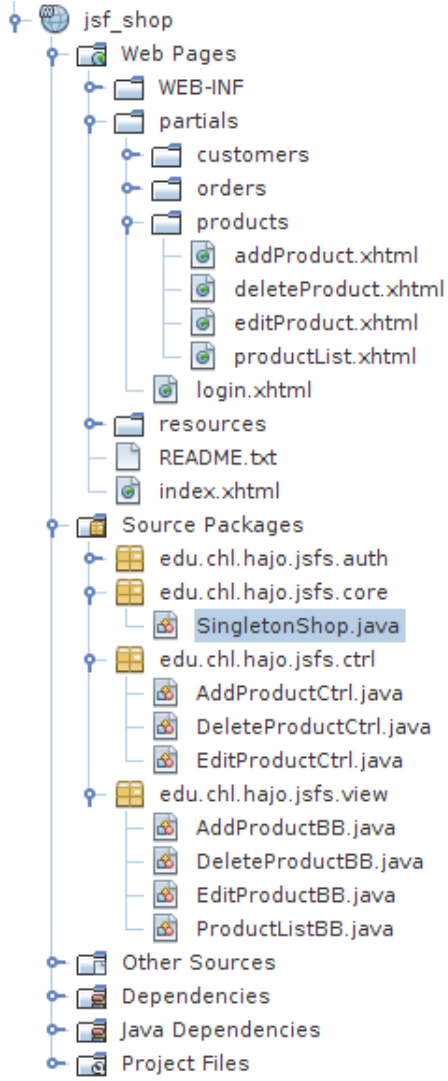
6 Authentication and Authorization

(Optional) Try to add "the official" JEE style of authorization, see code samples. Use the login page and the User class. Add an AuthBean class to handle login/logout.

7 Using a JSF component suite

(Optional) Clone the project and try a higher level component suite (PrimeFaces recommended). This may end in a single page ProductCatalogue (using advanced AJAX table components).

Appendix


jsf_shop
Web Pages
WEB-INF
partials
customers
orders
products
addProduct.xhtml
deleteProduct.xhtml
editProduct.xhtml
productList.xhtml
login.xhtml
resources
README.txt
index.xhtml
Source Packages
edu.chl.hajo.jsfs.auth
edu.chl.hajo.jsfs.core
SingletonShop.java
edu.chl.hajo.jsfs.ctrl
AddProductCtrl.java
DeleteProductCtrl.java
EditProductCtrl.java
edu.chl.hajo.jsfs.view
AddProductBB.java
DeleteProductBB.java
EditProductBB.java
ProductListBB.java
Other Sources
Dependencies
Java Dependencies
Project Files