Web applications/DAT076 HT2012
Group 02

Documentation

TareaWebShop is a project created during course DAT076-HT12 and has been in development for about 3-4 weeks. The project has now come to an end and it is time to show off a prototype.

The group name for the project is TAREA where the first letter means Team and the following letters stands for the group members.

(T)eam
(A)dam Book - pnumb: 871116 - mail: adambook1234@gmail.com
(R)ickard Edfast - pnumb: 911112 - mail: rickard.edfast@gmail.com
(E)rik Sjögren - pnumb: 910309 - mail: eriksjogren91@gmail.com
(A)lexander Kurkimäki - pnumb: 910912 - mail: lolip0op@hotmail.se

General overview

TareaWebShop is a basic web application for the purpose of online shopping. The application lets a user buy products that will be added to a shopping cart where a user later can checkout. The webshop is build with Java Enterprise Edition 6 (JEE6) as platform. The system is based on Maven and may present only running on a single computer using Glassfish.

There are also some roles and permissions for the system. The users may only buy existing products stored at a database and add products to their shopping cart. The admin on the other hand has permission to update the database with new products, edit products or remove products.

Fully functional use cases:

- Login-page for admin
- Working database (db)
- Show products stored at db
- Availability to update db with products
- Add products to shopping cart

Technical design

Tiers

The project is using the most common three-tiered solution consisting of presentation, application and storage. The presentation match a web browser showing HTML/CSS/ JavaScript, the application is dynamic web content technology using logic from JSF, and finally the storage that represents by a database using JPA.

Modules

The modules (packages) used in the project is the following:

Source Packages, contains backing beans that together with the xhtml-pages represents the view for the user. Each xhtml-page has its own bean containing information about what to be shown. These beans also acts as controller and perform all logic that is needed, like getting, setting and processing information.

Web-Pages, folder WEB-INF contains the configuration of the application. It defines loginconfiguration, roles and permissions. adminPages contains the xhtml-files only an admin has access to, files that can create/edit/remove products from the system. userPages contains xhtml-files only users has access to, like home, cart, products and checkout. The resource folder contains the necessary Cascading Style Sheets (CSS) used for the view experience.

UML and class description

To be found at the end of this document.

Used technologies

The application uses the following web based technologies: HTML, CSS, JavaScript, JSF and JPA.
The structure of the application is an OO-model based on a design pattern where we aim for high coupling and low cohesion between classes and methods. The GUI is basic but fully functional, just like a regular webshop.

Final reflection

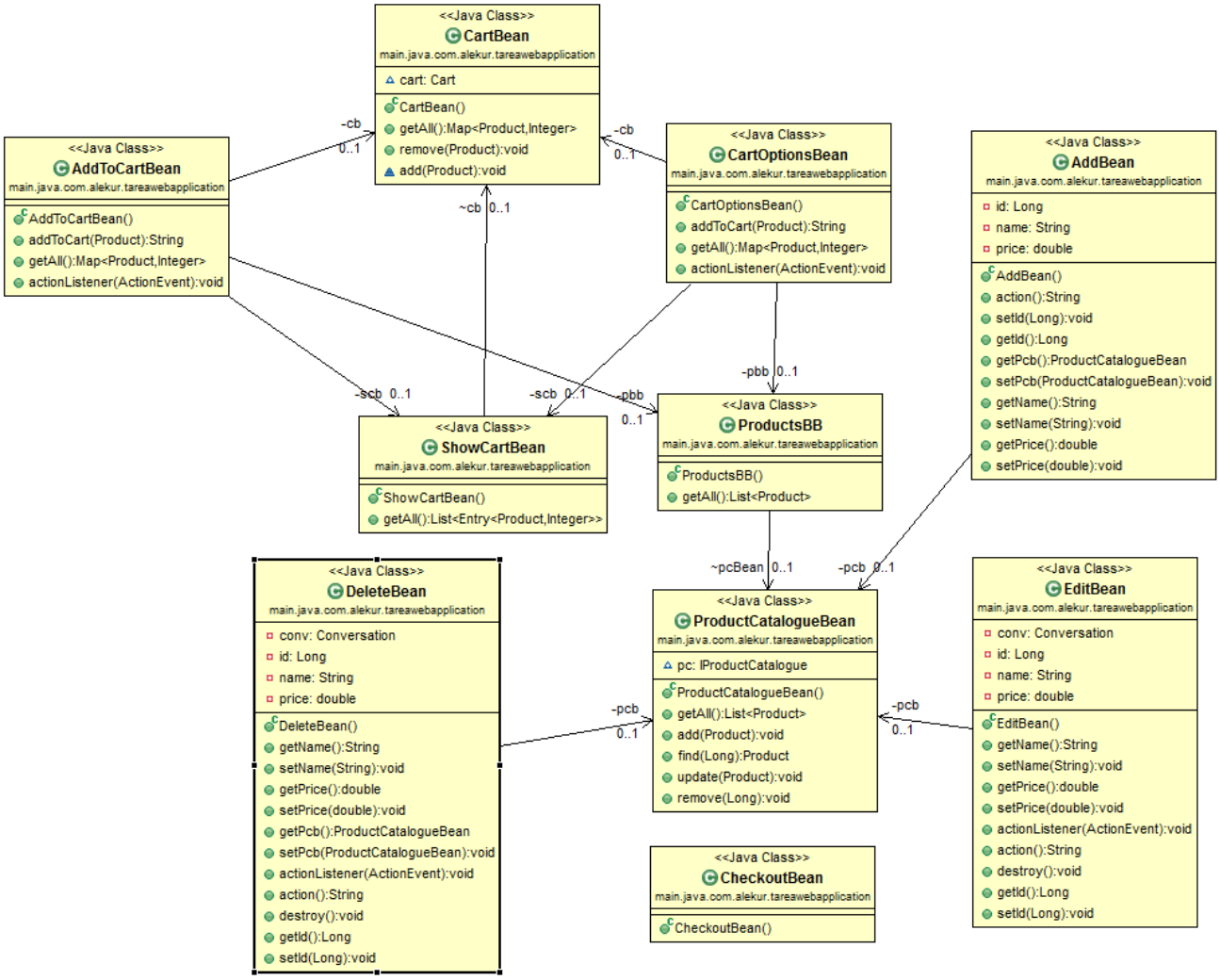
Strong parts about the webshop:
Overall we are satisfied about our webshop, it is very basic but it works. Given the short time we had at our disposal the webshop has working functionality with a number of use cases and a correct OO-model. The GUI is basic but usable with a nice login feature.

The weakness of the application is the small problem where a user cannot remove an added product from its shopping cart. Since we had a lot of problems with NetBeans and Git at the

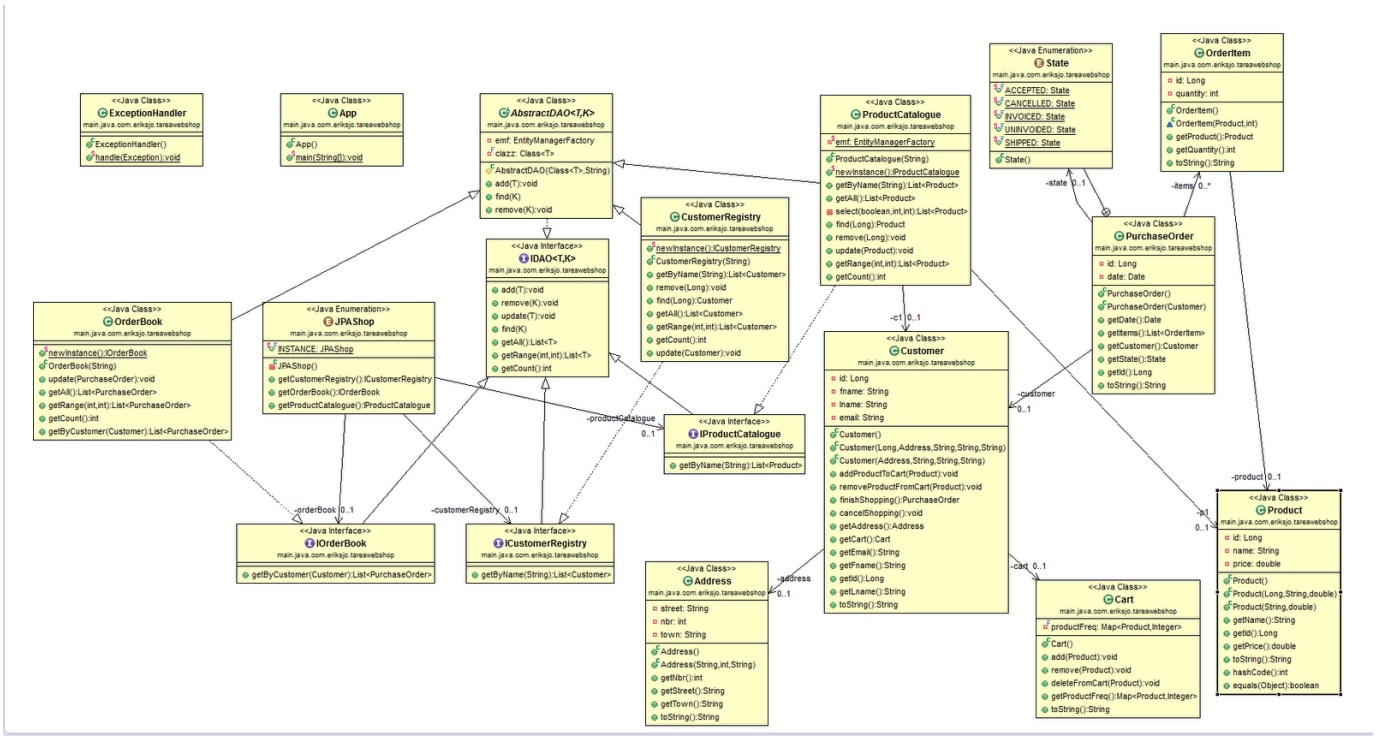
end, and mention that Erik's hard drive broke, we just did not have the time to fix it. The option remove in shopping cart has worked and should do but it does not at the moment.

UML and class description

The structure of the backing beans



The structure of the shop


<<Java Class>>
ExceptionHandler
main.java.com.erikajo.tareawebshop
ExceptionHandler()
handle(Exception):void
<<Java Class>>
App
main.java.com.erikajo.tareawebshop
App()
main(String[]):void
<<Java Class>>
AbstractDAO<T,K>
main.java.com.erikajo.tareawebshop
emf: EntityManagerFactory
clazz: Class<T>
AbstractDAO(Class<T>,String)
add(T):void
find(K)
remove(K):void
<<Java Class>>
ProductCatalogue
main.java.com.erikajo.tareawebshop
emf: EntityManagerFactory
ProductCatalogue(String)
newInstance():IProductCatalogue
getByName(String):List<Product>
getAll():List<Product>
select(boolean,int,int):List<Product>
find(Long):Product
remove(Long):void
update(Product):void
getRange(int,int):List<Product>
getCount():int
<<Java Enumeration>>
State
main.java.com.erikajo.tareawebshop
ACCEPTED: State
CANCELLED: State
INVOICED: State
UNINVOIDED: State
SHIPPED: State
State()
<<Java Class>>
OrderItem
main.java.com.erikajo.tareawebshop
id: Long
quantity: int
OrderItem()
OrderItem(Product,int)
getProduct():Product
getQuantity():int
toString():String
<<Java Class>>
CustomerRegistry
main.java.com.erikajo.tareawebshop
newInstance():ICustomerRegistry
CustomerRegistry(String)
getByName(String):List<Customer>
remove(Long):void
find(Long):Customer
getAll():List<Customer>
getRange(int,int):List<Customer>
getCount():int
update(Customer):void
<<Java Interface>>
IDAO<T,K>
main.java.com.erikajo.tareawebshop
add(T):void
remove(K):void
update(T):void
find(K)
getAll():List<T>
getRange(int,int):List<T>
getCount():int
<<Java Class>>
PurchaseOrder
main.java.com.erikajo.tareawebshop
id: Long
date: Date
PurchaseOrder()
PurchaseOrder(Customer)
getDate():Date
getItems():List<OrderItem>
getCustomer():Customer
getState():State
getId():Long
toString():String
<<Java Class>>
OrderBook
main.java.com.erikajo.tareawebshop
newInstance():IOrderBook
OrderBook(String)
update(PurchaseOrder):void
getAll():List<PurchaseOrder>
getRange(int,int):List<PurchaseOrder>
getCount():int
getByCustomer(Customer):List<PurchaseOrder>
<<Java Enumeration>>
JPAShop
main.java.com.erikajo.tareawebshop
INSTANCE: JPAShop
JPAShop()
getCustomerRegistry():ICustomerRegistry
getOrderBook():IOrderBook
getProductCatalogue():IProductCatalogue
<<Java Interface>>
IProductCatalogue
main.java.com.erikajo.tareawebshop
getByName(String):List<Product>
<<Java Class>>
Customer
main.java.com.erikajo.tareawebshop
id: Long
fname: String
lname: String
email: String
Customer()
Customer(Long,Address,String,String,String)
Customer(Address,String,String,String)
addProductToCart(Product):void
removeProductFromCart(Product):void
finishShopping():PurchaseOrder
cancelShopping():void
getAddress():Address
getCart():Cart
getEmail():String
getFname():String
getId():Long
getLname():String
toString():String
<<Java Class>>
Product
main.java.com.erikajo.tareawebshop
id: Long
name: String
price: double
Product()
Product(Long,String,double)
Product(String,double)
getName():String
getId():Long
getPrice():double
toString():String
hashCode():int
equals(Object):boolean
<<Java Interface>>
IOrderBook
main.java.com.erikajo.tareawebshop
getByCustomer(Customer):List<PurchaseOrder>
<<Java Interface>>
ICustomerRegistry
main.java.com.erikajo.tareawebshop
getByName(String):List<Customer>
<<Java Class>>
Address
main.java.com.erikajo.tareawebshop
street: String
nbr: int
town: String
Address()
Address(String,int,String)
getNbr():int
getStreet():String
getTown():String
toString():String
<<Java Class>>
Cart
main.java.com.erikajo.tareawebshop
productFreq: Map<Product,Integer>
Cart()
add(Product):void
remove(Product):void
deleteFromCart(Product):void
getProductFreq():Map<Product,Integer>
toString():String
-state 0..1
-items 0..*
-ct 0..1
-customer 0..1
-product 0..1
-pt 0..1
-cart 0..1
-address 0..1
-productCatalogue 0..1
-orderBook 0..1
-customerRegistry 0..1