1st Exercise on "Enterprise Application Integration"

in summer term 2013
Prof. Dr. Herbert Kuchen

Due: 13.05.2013, $12^{\underline{59}}$ 26.04.2013

Java EE (50 points)

Scenario

"Jackie Vaughn" (short: JaVa) runs a local pharmacy. For the drugs she sells, she keeps varying inventory levels to satisfy most demands immediately. On a regular schedule she replenishes her inventory from her wholesaler, who commissions and delivers orders multiple times per day. Because JaVa's pharmacy is in a contractual relationship with the wholesaler, individual orders and deliveries do not incur additional costs.

JaVa keeps track of her drug inventory levels and collects replenishment orders as to communicate them to her wholesaler in batch. To better control varying inventory levels, she inquires about being informed whenever drugs fall short of individual minimum inventory levels and suggesting predefined replenishment order quantities. Nonetheless, irregular customer demands also require manual replenishing.

Replenishment orders consist of various positions; one position per replenished drug. Initially, a replenishment order is an *open* state. Before it is communicated to the wholesaler, it should change into a *posting* state, thus prohibiting further changes to the replenishment order. When *posting* is done, the expected delivery date and time should be entered and the order should usually change into an *ordered* state.

Once the replenishment order reaches the pharmacy, it advances into a *finished* state and inventories of all replenished drugs are restocked accordingly. Since the wholesaler is absolutely reliable, replenishment orders need not be checked for inconsistencies when they arrive at the pharmacy. Additionally, changing into the *finished* state should require entering of the actual delivery date and time.

Yet, *posting* – and only *posting* – may be cancelled. In these situations, a replenishment should advance into a *cancelled* state. Furthermore, all positions should either be added to an *open* replenishment order that might have been created in the mean time. In case of any open replenishment order, a new replenishment order should be created and all positions added to it.

In Germany every drug in each package size is identified by a unique eight digit number that is known as *Pharmazentralnummer* (PZN). However, customers usually do not ask for drugs by their PZN but the drug's name. Therefore, JaVa also tracks a drug's label (name) as well as a short description aiding her pharmacists to better understand the drug at hand.

Use Cases

JAVA needs a business application to support the aforementioned processes. For now, the application's user interface should be completely web-based. Your applications needs to cover the following use cases:

- Drug
  - Create with unique PZN, name, and short description
  - Display details, inventory level, and replenishing orders
  - Change name and short description
  - Change optional configuration for replenishment suggestions
  - Withdrawal incl. suggestion for replenishment and restocking (keep track of amount and date)
  - Search by PZN and by approximate name
- Replenishment order
  - List of all replenishment orders
  - Details (to aid communication to wholesaler)
  - Update: proceed to next state; when proceeding to *ordered* enter expected delivery date/time; when proceeding to *finished* enter actual delivery date/time and adjust inventory levels accordingly
- Add new position to replenishment order
  - New positions can only be added while an order is in state *open*, else a new replenishment order should be opened

The system should not take into account legal notions, such as prescriptions or the narcotics law.

Organisational Issues

Implement the business application as described above using Java EE technology. The application should run on a standard JBoss Application Server 7 with the default configuration. Please use JPA Entities (`@Entity`) for persistence, EJB Session Beans for the business logic, and JavaServer Faces technology for the web layer. Have a look at the first two tutorials (JBoss setup and Java EE projects) and at the example application.

Please adhere to the following guidelines in order to isolate your solution from other applications:

- Include your two-digit group number in the name of all projects. This will, among other things, ensure that the name of your persistence unit and the path to your web application contain your group number.

- Please do **NOT** name your projects specific to this assignment, as your solution will also be the basis for the upcoming assignments.
- Supply your own data source with your persistence project:
  a) Create your own datasource PharmacyXX (replace XX with your group number). See Installing JBoss AS 7 in the JBoss Setup Tutorial for details. Use the following information (replace every occurrence of XX with your group number):
     - Name: *PharmacyXX*
     - JNDI Name: *java:/PharmacyXX*
     - Choose *hsqldb.jar* as driver
     - Connection URL: *jdbc:hsqldb:${jboss.server.data.dir}${/}hypersonic${/}PharmacyXX*
     - username: *sa*
  b) In `persistence.xml`, use `java:/PharmacyXX` as the content of the `jta-data-source` tag (replacing XX as before).

Please do not hesitate to use the discussion forum in Learnweb for any questions or problems. Other groups probably have had similar problems and thus can provide advice that fits the requirements at hand.

Please try to keep your solution simple and implement only what is required. The web-based user interface does not have to be pretty, but functional and tolerant of any user input. Your solution should feature a reasonably structured design and a clean coding style. Use Javadoc and inline comments to document and explain your code. Your solution should be self-explanatory – however, if you deem it necessary, you may include a short PDF document with your submission to provide further explanations.

Git Checksum

Use the Git repository that has been provided to your group. You may use the Eclipse plugin EGit as a git client. Create a subfolder `JavaEE` and save your Java EE project folders there. To submit a solution you can choose a checksum as your submission and just send the corresponding checksum to jan.ernsting@wi.uni-muenster.de.

Make sure that you pick the correct checksum, for example from the history view in Eclipse, and that your repository contains all files necessary to import the projects into Eclipse (everything except all "build" subfolders).