Middleware Project

TECHNISCHE UNIVERSITÄT DARMSTADT

Winter Term 2014/2015

Tobias Freudenreich

Implementation assignment 21.10.2014

The implementation assignment combines a variety of middleware technologies. The lecture teaches you the concepts of those technologies. In this assignment, you will learn how to use them and integrate them with each other. During the assignment, you will use the Java EE ecosystem with an application server, database server (via ORM mappers) and a webserver contributing to a large application. The application you build will use asynchronous communication with a queuing and publish/subscribe model. While implementing a small project detailed below, you will learn how these middleware technologies interact and help you as the system developer in implementing the system.

The project will run throughout the semester with three defined milestones. **For each milestone you have to submit a progress report of two to four (not more, not less) pages that**

- **describes your major design discussions (e.g., architectural overview (diagram), rejected implementation alternatives),**
- **encountered problems (e.g., Java EE configuration issues),**
- **and implemented features (e.g., description of functionality).**

Send the progress report as a PDF document to Tobias by email.

We will choose groups to check consistency of their system with their progress report. We will focus on groups with weak progress reports, i.e., the better understanding we gain from the progress report the less need we see for an attestation.

We define a list of six features, each feature being worth 10 percent points. At each milestone at least one additional feature has to be implemented. If you miss a milestone, 10 points are taken off your final score. For example: If you complete 5 features at the third milestone, but you have only one feature at the first and second milestone, you are awarded 40 points.

Important Dates:

- 11.01.2015: Group details (names, IDs, contact information) – **required to qualify for bonus**
- 11.01.2015: Milestone 1 (at least 1 feature)
- 15.02.2015: Milestone 2 (at least 2 features)
- 15.03.2015: Milestone 3 = Project due (at least 3 features) – **this must not be missed!**

We encourage working in teams of two or three people.

Scenario Description

Omazan Inc. provides high quality logistics services to customers. The CTO of Omazan is convinced that real-time shipment tracking will greatly enhance customer experience. He hires you as experts to implement the desired functionality. Unfortunately, you discover that the existing Enterprise Resource Planning (ERP) system is implemented in COBOL; so you take the opportunity and suggest the CTO to implement a brand new ERP solution using Java EE components (Enterprise Beans, Java Server Faces/Pages, Java Persistence API). After long negotiations you receive the contract specifications.

Features at a glance

- Customer and Product Data Management
  a) Business Logic (5 points)
  b) User Interface (5 points)
- Order Process Support
  a) Order Process (4 points)
  b) Shipment Status Update (2 points)
  c) User Interface (4 points)
- Shipment tracking via JMS messages
  a) Automated shipment status updates (4 points)
  b) Shipment position updates (6 points)
- Notification on exceptional events
  a) On shipment status page (5 points)
  b) By email (5 points)
- Mobile client support (10 points)
- Consistency guarantees (10 points)

Customer and Product Data Management

Your ERP system needs to manage Omazans master data, that is *customer* and *product* data. Thus, your system must support the following:

- **Business Logic:**
  Define the necessary beans in your application server to support the adding and editing of customer data and product data.
- **User Interface:**
  Omazan employees should be able to add and edit the respective data using a web interface (e.g., JSF/JSP) and a native Java client.

You are required to use an ORM mapper for working with the database.

Note: Although we ask for a web-based interface, you are not required to build anything fancy. The same is true for the native Java client. A console-based application will perfectly do. However, we will not stop you from coming up with something elaborate.

Order Process Support

This feature will allow customers to issue orders and track orders' statuses. Specifically, you must implement the following functionality:

- **Order Process:**
  The system has to support a basic order and delivery process. Customers place an order that can contain multiple products. Each order is assigned a **Shipment ID** by which customers can later track the status of their shipment.
- **Shipment status update:**
  After successful delivery the shipment status is marked as *delivered* by an Omazan employee, which completes the order and delivery process. Customers can check the status anytime.
- **User Interface:**
  Build a web-based interface that allows customers to place orders and check their status as described above. Also, employees should have a means of updating the status of an order.

Note: You do not have to deal with authentication or other security issues for this exercise.

Reactive ERP – Shipment Tracking

The next step is the integration of reactive behavior in your ERP solution (e.g., with message-driven beans). In the basic version Omazan employees mark shipments manually as *delivered*. However, recently all trucks of Omazan were equipped with monitoring technology and emit position and delivery data. It is now your task to integrate those events with your ERP solution. During the order, a shipment is assigned to a truck which transports it to the customer. A truck can hold multiple shipments.

As messaging technology the Java Message Service (JMS) is used. Two different event types are provided for shipment tracking (XML is payload in JMS text messages):

Delivery Event

```
<deliveryevent>
      <shipmentId> … </shipmentId>
</deliveryevent>
```

Position Event

```
<positionEvent>
      <truckId> … </truckId>
      <long> … </long>
      <lat> … </lat>
</positionsEvent>
```

The desired functionality of the system is as follows:

- **Automated Shipment Status Updates:**
  Delivery events cause an automated update of the shipment status to *delivered*.
- **Shipment Position Updates:**
  Position events are used to provide customers with the latest position of their shipments. With each position event the current position of each shipment inside the truck is updated.
  Customers can check the shipment status and see the current position of their shipment.

To test and demonstrate this functionality, you will also have to write a JMS producer sending the appropriate events. For demonstration purposes these may be triggered manually or automatically.

Note: You may use any method of assigning shipments to trucks (e.g., randomly, selected during ordering, etc.). You do not have to check any consistencies like capacity or if the truck is already on the road.

Reactive ERP – Notification on exceptional events

Additionally, Omazan wants to notify its customers about any exceptions, such as delayed delivery because of a traffic jam. You have access to the following additional event:

Exception Event

```
<exceptionEvent>
      <truckId> … </truckId>
      <exceptionDescription> … </exceptionDescription>
</exceptionEvent>
```

For each *exception event* all customers should receive a notification in case their shipment is concerned. Notifications shall be given in two ways:

- **Shipment Status Page:**
  The shipment status page should display the exception description in addition to the information already there. For this prototype it is sufficient if you display the notification the next time the page is loaded.
- **Email:**
  Notifications should be sent to the customer's email address including the exception description.

Mobile Clients

Omazan Inc. is on its way to become a global player in logistics. Employees travel all over the globe to constantly monitor and optimize the shipment processes. They require local access to the master data to be able to judge process progress. It is your task to provide clients with a way to query a current snapshot of the customer and shipment database.

online mode offline mode

Note: You do not have to implement a real mobile client (e.g., Android). A crude simulation of a mobile client is sufficient, i.e., a regular Java application that acts as mobile client is sufficient.

Consistency Guarantees

Ideally this snapshot should be synchronized between all clients currently connected to the system. Extend your solution in a way that all connected mobile clients have an identical snapshot of the database if one client queries a snapshot.

Note: Use 2PC for this task.

Notes

- The project is targeted to be implemented with different Java EE components. For the development we suggest to use NetBeans or Eclipse and the Glassfish application server (http://glassfishplugins.java.net).
- The order example in the Java EE Tutorial is a good starting point. It is fine to build upon existing example code. However, you have to be able to answer questions about all code in your project.
- All deadlines are **hard deadlines** and are not negotiable!
- We expect that solutions implement all required features, not just some of them. Please make sure to test your application.