Enterprise Application Integration 2009/2010
Assignment #2 – J2EE Enterprise Applications

Objectives

- Gain familiarly with the development of three-tier enterprise applications using the **Java 2 Enterprise Edition (J2EE)** model. This includes the development of applications based on **Enterprise JavaBeans (EJB)**, the development of a **Web Tier** and the use of a **Persistence Engine**.

Due Date

- **30th October 2009**

Scheduled Effort

- **30 hours/student**, according to the following plan:
  - 12 hours → Reading and learning technologies (e.g., doing tutorials)
  - 14 hours → Coding and testing
  - 4 hours → Writing the report

PLEASE WRITE DOWN THE EFFECTIVE NUMBER OF HOURS SPENT IN EACH TASK
THE REPORTED EFFORT SHOULD BE PER STUDENT!

Final Delivery

- You must submit your assignment using Moodle (http://moodle.dei.uc.pt).
- The submission contents are:
  - Source code of the project.
  - Project ready to deploy on the application server (e.g., WAR file).
  - A small report (5 pages max) about the implementation of the project. The report must specify the number of hours spent per student while working on the assignment. Please use PDF. I will not open any word documents.

Software

J2EE Platform

For this project you are required to use a J2EE platform. The two recommended platforms are:

- **GlassFish Enterprise Server v2.1, from Sun**. This platform is available at: http://java.sun.com/javaee/downloads. If you decide to use this platform, it's recommended that you also use **NetBeans IDE 6.7 Java EE** version. It will make your development process much easier!!! Use the packaged version with includes both **NetBeans, Java EE and GlassFish**.

  The main advantage of using this application server is that all Sun's documentation and the J2EE tutorial are made specifically for this server and its demonstration applications.

- **JBoss Application Server 5.1.0.GA**. This platform is open-source and is available at: http://labs.jboss.com/jbossas/downloads/ (current version is **5.1.0.GA)**. In case you decide to use this platform, I seriously recommended that you use **Eclipse IDE for Java EE Developers** edition for developing you applications (http://www.eclipse.org/). Notice that if you have the regular eclipse distribution installed, you must upgrade it to J2EE. It's also recommended that you install **JBoss Tools for Eclipse 3.1** (http://www.jboss.org/tools/download.html).

  The main advantage of using this application server is **its wide acceptance in the market**, being the leading open-source solution for J2EE. Many of the J2EE applications being developed and deployed today are using it. Thus, it terms of your CV, it adds value to have worked with it. The main disadvantage is that the documentation is not as good as Sun's J2EE tutorial and, in fact, extensible refers to it Sun's tutorial.

You can choose to use another J2EE platform. Nevertheless, you should consult with the professor first. **Even so, we recommend that you use the JBoss Application Server.**

EJB Model

Currently there are two EJB models in use: the "old" EBJ 2.1 standard; and the new EBJ 3.0 standard. Consider that:

- Developing with the 2.1 model is much more complicated and troublesome, especially in terms of deployment. The advantages are that it's well established and very well documented, although it's becoming rapidly obsolete. Also, because it's so established, all servers support it directly.

- Developing with the 3.0 model is, in general, much simpler and less error prone, especially in terms of deployment and connecting to databases. This comes mainly because of its use of *annotations* (a feature introduced in JDK 5.0 for supporting declarative programming) and the use of POJO (*Plain Old Java Objects*) for managing persistence to databases. The disadvantages of using EJB 3.0 are that it's not so well documented and not all servers fully supported it. Nevertheless, it's the future. EJB 2.1 is rapidly being phased out.

In light of this, **you must use the EJB 3.0 model** for completing this assignment.

Database

When programming in J2EE it's quite common to use a database or persistence engine for saving the data. In this project you will have to use one. The recommended ones are:

- **Hibernate**, available online at: http://www.hibernate.org/ and which is in fact part of the "JBoss family". Hibernate is a relational persistence engine for Java which allows you to easily write Java code corresponding to data that is transparently saved to a database. It also provides a powerful query engine directly integrated in Java.

  The main advantage of using Hibernate is its **simplicity**, compared with normal databases, and its excellent integration with Java. Also, because it's becoming a *de facto* standard in the industry, it adds value to learn about it. The disadvantages are that it's Java-specific and it's one more thing to learn.

- **MySQL**, available online at: http://www.mysql.com/. It's one of the most popular open-source database engines. (You may also use other SQL engines like PostgreSQL, SQL-Server, etc.)

  The main advantage of using a standard database server is that it is standard. Also, if you don't have much experience with database servers, probably you should learn about them before learning about Hibernate. The main disadvantage is that SQL is not easily and neatly integrated into Java, generating ugly, non-maintainable and non-object oriented code in applications. If you have already used databases in the past, we strongly recommend that you use Hibernate.
- **Java Persistence API (JPA).** The Java Persistence API is now part of the J2EE EJB3 specification. Basically, the approach is the same as in Hibernate. You use annotations in classes

to express how classes will be stored in a database. In fact, the model was inspired in Hibernate. When using these annotations you need to have a database behind where to store the data. If you want you can use Hibernate and/or Hypersonic (an embedded database in Hibernate) in conjunction with JPA. Just refer to https://www.hibernate.org/397.html

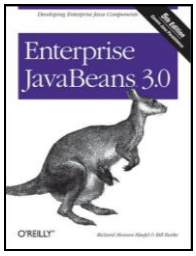
This is the recommended approach since all modern J2EE application servers have to support JPA.

Bibliography

If you are using JBoss, read the next page (*INTRODUCTION*), do the tutorial I'm providing with the assignment, and then move to the documentation bellow!

Books

There is extensible bibliography online about J2EE. Nevertheless, for this assignment, we strongly suggest that you start by reading a book that gives an overview about the basic concepts of J2EE. Contacting with a structured approach to J2EE development, like what is traditionally presented in books, is important. Although you may feel overwhelmed by the size and complexity of these books, don't despair. For this assignment, you don't need to worry about transactional and security requirements. Thus, you can skip the corresponding chapters. **The following book is highly recommended**:



Enterprise JavaBeans 3.0 (5th Edition)
by Bill Burke and Richard Monson-Haefel

O'Reilly Media
ISBN 059600978X
May 2006

The interesting thing about it is that it not only covers EBJ 3.0 quite well, but also the last part of the book is devoted to JBoss 4.0. It's available online using Safari: http://safari.oreilly.com/059600978X. Do note that things are slightly different in JBoss 5.1, but not different enough to make you despair. **If you can, get this book**.

Online Resources

There are many online resources about J2EE and EBJ 3.0. One of the most important resources is the J2EE Tutorial, available at: http://java.sun.com/javaee/5/docs/tutorial/doc/. You should skim, at least:

- Part One, Chapter 1 – Overview
- Part Four – Enterprise Beans (*You can skip Chapter 23*)
- Part Five – Persistence (*You can skip this part if using Hibernate directly. Nevertheless, if you want to use the "real" EJB 3.0 persistence API then you must read it and refer to the Hibernate Entity Manager: http://docs.jboss.org/hibernate/stable/entitymanager/reference/en/html/*).

Besides the J2EE tutorial, you can also read several articles about EJB:

- "*Simplify enterprise Java development with EJB 3.0*", Parts I and II, by Michael Yuan, available online at:
  http://www.javaworld.com/javaworld/jw-08-2005/jw-0815-ejb3_p.html (Part I)
  http://www.javaworld.com/javaworld/jw-09-2005/jw-0912-ejb_p.html (Part II)
- "Enterprise Java Beans (EJB) 3.0", by Steven Haines, available online at:
  http://www.informit.com/guides/printerfriendly.asp?g=java&seqNum=239&rl=1

If you are use JBoss you should start with the following documentation:

- "Installation and Getting Started Guide", available online at:
  https://www.jboss.org/community/wiki/JBossAS5InstallationandGettingStartedGuide
- "JBoss EJB 3.0 Tutorial", available online at:
  http://www.jboss.org/ejb3/docs/

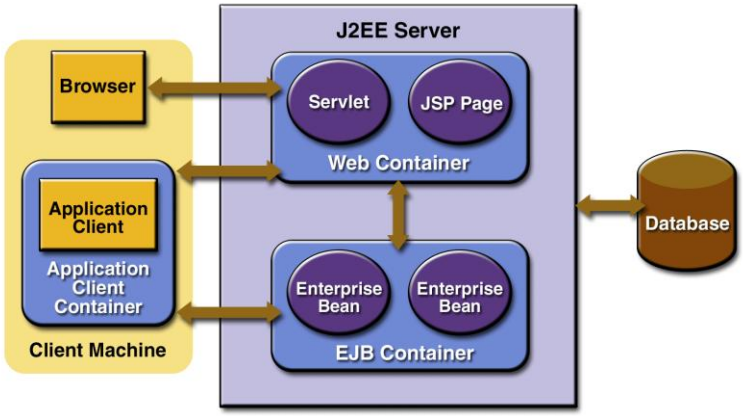
Also, all documentation for JBoss is available at:
http://labs.jboss.com/jbossas/docs/ **(This is an important link.)**

If you have access to the book mentioned above, the JBoss tutorial at Part II is quite practical!
Start at: http://safari.oreilly.com/059600978X/entjbeans5-PART-II

Introduction

Application Integration can be made across many types of different applications and systems. Nevertheless, nowadays, most large-scale Enterprise Systems are being developed either using the J2EE Enterprise JavaBeans model or by using .NET augmented with enterprise services (e.g. COM+ transactional and messaging capabilities). Also, most of the emphasis recently put on SOA (*Service Oriented Architecture*) and ESB (*Enterprise Service Bus*) rely on application servers and enterprise-grade systems. As a consequence, for the purpose of this course, it is important that you contact with the reality of Application Servers before we can focus on SOA and ESB. That's the general objective of this assignment.

The idea of J2EE is that enterprise-level applications are structured in three layers: presentation, business logic and data. This is shown in the next image.



We are mostly concerned with the business code of the application. The business code runs inside a J2EE application server which is fully responsible for managing it. The advantage of using an application server, when compared with developing a stand-alone application, is that the programmer does not have to implement many boilerplate enterprise-level functionalities. These are already provided by the container. In particular, a J2EE server provides for:

- Transactional contexts for guarantying data integrity and recovery in case of crashes;
- Connection polling for large number of invocations and optimized access;
- Integrated security management;
- In many cases, distributed computing, load-balancing and clustering capabilities.

It should be noted that when one starts you use an application server it gets the felling that it's slow, cumbersome, and that it has a difficult programming model. Is almost tempting to use a much simpler framework like JSP+Database or alike (Struts, Spring, etc.). The advantages only become clear in enterprise contexts where distributed transactions, transparent load-balancing across several servers and millions of invocations have to be made with guarantied operation. ***In fact, for small-scale applications, J2EE should not be the way to go.*** ☺

To conclude, note that there are three types of Enterprise Beans:

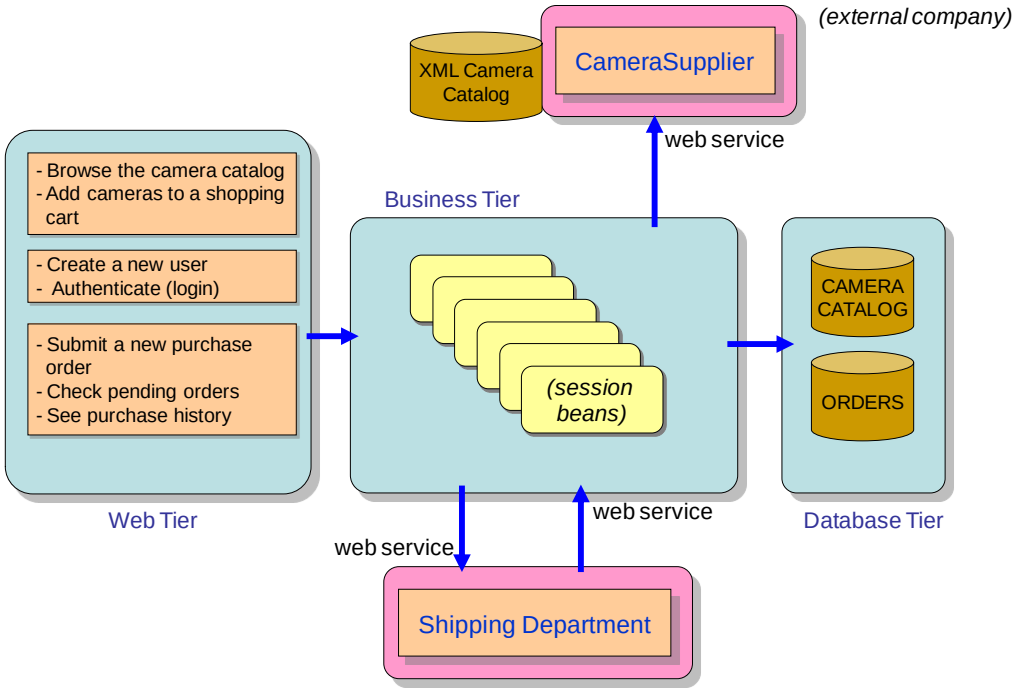
- Session Beans, which represent connections from clients and client invocations;
- Entity Beans, which represent data – normally a row in a database identified by a primary key;
- Messaging Beans, which are activated when incoming messages arrive (covered on the last assignment of this course).

There has been a huge change and simplification from EJB 2.1 to EJB 3.0. In particular, Entity Beans and "Container-Managed Persistence" (CMP) have been quite altered. In fact, in EJB 3.0 the emphasis is mostly on Session Beans and Messaging Beans. Communication with the database is either done directly with POJO and/or Java's Persistence API. Take this in consideration when reading online documentation about EJB: have a very clear view if you are reading about EJB 2.1 or EJB 3.0, they are different!

The Assignment

In this project you will develop an application called "Low-Price Cameras Online" (LPCO) for an online site. The system allows users to browse a camera catalog and submit purchase orders for one or more cameras. It also allows users to search for cameras that are not even present in catalog of the company by using an external searcher. This external searcher is a service made available by a camera importer that has a huge camera database. The system is also able to ship the orders that customers submitted by using a shipping company. At any given time the user can check the status of his pending orders and also access his purchase history.

The next image depicts the high-level architecture of LPCO.



The system works as follows:

1. A user can browse the web site of the company looking for interesting cameras that he wants to buy. The information regarding those cameras is stored in a database ("Camera Catalog").
2. The web site supports a shopping cart to where one or more cameras can be added.
3. If the user is looking for a camera that isn't available in the catalog, he can perform a free text search (e.g., "Canon EOS Rebel"). The system contacts a web service called CameraSupplier, corresponding to an external company which is a large camera importer.
4. CameraSupplier has a large XML database with cameras (generated in your last assignment). The search is performed in the database against the name of the model. Cameras where all search terms are present are returned to LPCO.
5. Any cameras returned by the CameraSupplier are automatically added to the Camera Catalog database.
6. The LPCO web site also allows new users to be created and users to authenticate (i.e., login). When creating a new user the usual information has to be provided (username, password, address, email, etc.)

7. Clients can submit a purchase order for the items in their shopping cart. The order must contain a delivery address.
8. When an order is submitted, the user must necessarily be authenticated.
9. When processing the order, in real life, the credit card of a person would be checked. In your case, that operation will be simulated by generating a random number and making sure that in 75% of cases the user has money. All orders are stored in a database ("Orders").
10. If the user has money, the system contacts and external department called "Shipping Department". This department is responsible for sending the goods to the client. Communication with this department should be made by web services.
11. When the Shipping Department web service is invoked, the invocation should return immediately. Since we are simulating a “real life system”, and shipping can take several days, it wouldn’t make sense to have a blocking call. The Shipping Department should be an “autonomous” entity.
12. Shipping of an order is simulated by passing of time (3-7 seconds represent 3 to 7 days). After shipping occurs the Shipping Department must call-back the LPCO system informing it of the shipping. This is also done by using a web service that LPCO has to provide.
13. Shipping is simulated by sending an email to the client saying that the order has shipped. (It must be the LPCO system doing it and not the Shipping department.)
14. At any given time a user can check the status of an order. This means that the Orders database always contains the most up-to-date status of an order.
15. At any given time a user can check his purchase history. This includes partial and total amounts spent.

Note that the business logic of this information system is implemented in J2EE. It also makes use of “pseudo-external” services that the LPCO invokes, but doesn’t own or control. We say “pseudo-external” because being this a toy project, they will be implemented by you, although they are conceptually external. You should treat them as such.

The business logic of the system will be implemented in session beans inside of the J2EE server. You are responsible to decide which session beans to use. Be prepared to justify their use. Also, carefully think if they should be stateless or stateful. You will be asked about this. By careful to define proper interfaces between the several tiers.

We envision that the database tier has at least information about: a) the camera catalog containing all cameras the company knows about; b) the purchase orders that took and are taking place; c) the users registered in the system. You are free to design this as you see fit, but it must make sense. (Note: you don't need to use two or three different physical databases. Use just one.)

Finally, the following points apply:

- You are free to use as many beans as necessary (either session beans or entity/persistence beans). You shouldn’t use messaging beans (i.e., JMS). That will be covered latter on.
- The web interface doesn’t have to be sophisticated. The emphasis should be on the structuring of the business logic. In particular, the interfaces that are offered and published should be well thought.
- Don’t make the database any more complicated than it must be.
- For this assignment, in terms of security, you just need to make sure that users are authenticated and are able to access only their information.
- Be very careful about choosing between stateless beans or stateful beans. Also, be very careful about not passing huge amounts of information between application layers when you don’t need to. Also, be sure you understand the difference between a local interface and a remote interface when using session beans.
- For this assignment you are not required to think about transactional properties. Nevertheless, if you do it, it will be considered in the grading of the project.
- Finally, don’t make the assignment complicated. But, make it realistic in terms of implementation assumptions! E.g., external services are external companies which operate autonomously and shouldn’t, in general, block. Pictures take time to download. Databases are not shared across organizations. Etc.

Good Luck!