Java EE Intro

CSE 305 – Principles of Database Systems
Paul Fodor
Stony Brook University

http://www.cs.stonybrook.edu/~cse305
JavaEE Setup is much of the battle
Lots and lots of stuff

- **Tools**
  - i.e. NetBeans, Glassfish, etc.

- **Core Libraries**
  - JSF, JPA, Ajax, etc.

- **Related APIs**
  - PrimeFaces, Struts, Spring, etc.
Common Java Frameworks

1. Spring MVC (23%)
2. Struts 1.x (15%)
3. Apache Axis (15%)
4. Apache Xerces (14%)
5. Hibernate (12%)
6. JDOM (12%)
7. Java Applet (8.1%)
8. Apache Velocity (7.9%)
9. Apache ORO (7.0%)
10. JAX-WS (6.5%)

Source: VeraCode Blog

http://www.veracode.com/blog/2012/01/top-ten-java-frameworks-observed-in-customer-applications/
A good place to start

• Core Libraries
  – Java Server Faces
  – JSTL & EL
  – Java Persistence API
  – Enterprise Java Beans
  – JavaScript & Ajax
A good place to start

• Related AP
  – RichFaces
  – PrimeFaces
  – Spring
  – Struts
  – JQuery
Servers, Servers, and More Servers

• What are:
  – Web Servers?
  – Application Servers?
  – Enterprise Servers?
  – Database Servers?
  – Java EE Servers?

• Abstraction, abstraction, abstraction
Multi-Tier JavaEE Applications
First: Annotations

- Provide data about a program to:
  - compiler  
    ```java
    @Override
    void mySuperMethod() {  }
    ```
  - Tools
    ```java
    @SuppressWarnings("unchecked")
    void myMethod() {  }
    ```
  - JVM

- Can be applied to:
  - classes
  - fields
  - Methods

- Can contain elements with values
Annotations Look-Up

- Scattered in the Java EE 6 API. Ex:

- Via cheat sheet:
CDI?

- Context & Dependency Injection
  - f.k.a Web Beans

- Contexts?
  - lets you use JavaBeans, EJBs, etc. in other contexts

- Dependency Injection?
  - context polymorphism

- CDI will do much of the work behind our annotations
@Named

- makes a bean accessible via a Facelet page. Ex:

```java
@Named("cart")
@SessionScoped
public class ShoppingCart
```

- `bookcatalog.xhtml`:

```xml
<h:commandLink id="check" action="bookshowcart"
    immediate="true"
    rendered="#{cart.numberOfItems > 0}">
    <h:outputText value="#{bundle.CartCheck}"/>
</h:commandLink>
```
So what are facelets?

• A page declaration language
  – used to build JSF views

• And tag libraries:
  <ui: for templating
  <h: for HTML components
  <f: for custom actions
  <c: for JSTL (Java language features)
  <fn: more JSTL (Java API features)

• JSTL: JavaServerPages Standard Tag Library
Facelets use EL

- Expression Language

- For what?
  - evaluate expressions
  - get/set data
  - invoke methods

- EL defines:
  - how to write expressions: ${customer.age + 20}$
  - how to reference get/set: ${customer.name}$
  - how to invoke methods: val="#{customer.validateName}"
What's a managed bean?

• In JSF apps, typically each page connects to one. Why?
  – defines methods/properties associated with the page's components
  – Why?
We've Seen Front-End JavaEE

- JavaServerFaces
- XHTML
- CSS
Now for the Back-End

• What's the Java Persistence API (JPA)?

• What's an Enterprise Java Bean (EJB)?
  – server-side component
  – encapsulates business logic
  – Ex:
    • check inventory
    • order products
Java Persistence API (JPA)

• Provides object/relational mapping to relational DB

• What does that mean?
  – makes it easy to talk to Dbs

• Why?
  – separate roles and employ abstraction
The JPA Entity

• A JSP Domain

• It represents a DB Table
  – an Entity Instance would be a row

• Mapping done via annotations

```java
@Entity
public class Book {
}
```
Why use EJBs?

- Simplify development of large, distributed apps

- Scalability

- EJB Containers provide system-level services to EJBs:
  - transaction management
  - concurrency
  - security features

- Again, separation of roles
  - thinner clients
Two types of EJBs

• Session EJBs
  – performs a task for a client

• Message-Driven EJBs
  – acts as message listener (like for JMS)
Session Beans

- Clients (i.e. facelets) invoke session bean methods
- Does its work on the server
- Not persistent
  - its data not saved to database
- Three Types:
  - stateful, stateless, & singleton
Stateful Session Beans

• Not shared
  – belongs to a single client
  – can store info about clients

• Lasts for duration of client/server session
  – when client terminates, bean terminates
Stateless Session Beans

• Support multiple clients

• Methods do not “remember” clients

• Scalability advantages over stateful beans
Singleton Session Beans

- Lives for duration of application
- Only one of them
- Shared among clients
Why use stateful beans?

• The bean's state represents the interaction between the bean and a specific client

• The bean needs to hold info about the client across method invocations

• The bean mediates between the client and the other components of the application, presenting a simplified view to the client

• Behind the scenes the bean manages the workflow of several EJBs
Why use stateless beans?

- The bean has no data for a specific client
- In a single method invocation, the bean performs a generic task for all clients
- The bean implements a web service
Why use singleton beans?

• State needs to be shared across the application

• A single enterprise bean needs to be accessed by multiple threads concurrently

• The application needs an enterprise bean to perform tasks upon application startup and shutdown

• The bean implements a web service
How does a client use EJBs?

• Dependency injection
  – i.e. @EJB

OR

• JNDI Lookup
  – for non Java EE apps
The IDE

<table>
<thead>
<tr>
<th>Supported technologies</th>
<th>Java SE</th>
<th>Java EE</th>
<th>C/C++</th>
<th>PHP</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetBeans Platform SDK</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java SE</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java FX</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java EE</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java ME</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTML5</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java Card™ 3 Connected</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C/C++</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Groovy</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHP</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bundled servers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GlassFish Server Open Source Edition 4.0</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apache Tomcat 7.0.34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- We want EVERYTHING!
Hook-Up NetBeans to MySQL

- Services Tab
- New Connection
Hook-Up NetBeans to MySQL

- Now we can start building databases
- Connect to it
CRUD?

- Create Read Update Delete

- Basic functions of persistent storage

- *Create* or add new entries
- *Read*, retrieve, search, or view existing entries
- *Update* or edit existing entries
- *Delete/deactivate* existing entries

- If you have these, you can make any complex Web app
Generating a JavaServer Faces 2.x CRUD Application from a Database

- [https://netbeans.org/kb/docs/web/jsf20-crud.html](https://netbeans.org/kb/docs/web/jsf20-crud.html)

JavaServer Faces (JSF) 2.x for front-end web pages, validation handling, and management of the request-response cycle.

Java Persistence API (JPA) 2.0 using EclipseLink to generate entity classes from the database, and manage transactions. (EclipseLink is the reference implementation for JPA, and is the default persistence provider for the GlassFish server.)

Enterprise JavaBeans (EJB) 3.1, which provides you with stateless EJBs that access the entity classes, and contain the business logic for the application.

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)
First create the database

  - mysql_create_consult.sql
  - mysql_insert_data_consult.sql
- Start the database server
- Create Database (called consult)
- Connect to database
- File → Open File → mysql_create_consult.sql
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';

DROP SCHEMA IF EXISTS consult;
CREATE SCHEMA consult;
USE consult;

CREATE TABLE address (  
  address_id INTEGER NOT NULL AUTO_INCREMENT,  
  line1 VARCHAR(50) NOT NULL,  
  line2 VARCHAR(50) NULL,  
  city VARCHAR(50) NOT NULL,  
  region VARCHAR(50) NOT NULL,  
  country VARCHAR(50) NOT NULL,  
  postal_code VARCHAR(50) NOT NULL,  
  CONSTRAINT address_pk PRIMARY KEY ( address_id )  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE consultant_status (  
  status_id CHAR NOT NULL,  
  description VARCHAR(50) NOT NULL,  
  CONSTRAINT consultant_status_pk PRIMARY KEY ( status_id )  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE consultant (  
  consultant_id INTEGER NOT NULL AUTO_INCREMENT,  
  status_id CHAR NOT NULL,  
  email VARCHAR(50) NOT NULL,  
  password VARCHAR(50) NOT NULL,  
  hourly_rate DECIMAL(6,2) NOT NULL,  
  billable_hourly_rate DECIMAL(6,2) NOT NULL,
  ...
Run the SQL script

- This will build the tables

... Executed successfully in 0.316 s, 0 rows affected.
Line 100, column 1

... Executed successfully in 1.193 s, 0 rows affected.
Line 101, column 1

Executed successfully in 0 s, 0 rows affected.
Line 108, column 1

Execution finished after 17.665 s, 0 error(s) occurred.
Examine the Database
Create the Web App

- File ➔ New Project ➔ ConsultingAgency
Project Setup

- Specify your server
  - GlassFish 4.0

- Select your Framework
  - we’ll use JavaServerFaces
  - note the other choices

- Now what?
  - We want our Web app to be able to talk to the DB
Generate Entity Classes from the Database

• We’ll need to create a New Data Source
New Data Source: jdbc/consult
Add All >> to jpa.entities package
Examine the generated Entity Classes

- Note the generated SQL & Annotations
- DB Note:
  - no entity class generated for join tables
  - additional classes generated for tables with composite primary keys

(©) Pearson Education Inc. and Paul Fodor (CS Stony Brook)
Generate JSF Pages from Entity Classes

• What are the JSF Pages for?
  • viewing & modifying the data
  • that’s what web apps do

• For each entity class:
  • a stateless session bean that extends AbstractFacade.java
  • a JSF session-scoped, managed bean
  • a directory containing four Facelets files for CRUD capabilities

• Also:
  • AbstractFacade.java class that contains the business logic for creation, retrieval, modification and removal of entity instances
  • utility classes, default stylesheet, localized message properties, etc.
Generate JSF Pages from Entity Classes
Add All >> to jpa.session package
Add All >> to jpa.session package

- Note the JSF classes package and /resources/Bundle
Facelets & JSF Managed Beans

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)
You can change the language.
Run the Project

Hello from Facelets
Show All Address Items
Show All Billable Items
Show All Client Items
Show All Consultant Items
Show All ConsultantStatus Items
Show All Project Items
Show All Recruiter Items

Create New Address

AddressId:
Line1:
Line2:
City:
Region:
Country:
PostalCode:
Client:

Save

Show All Address Items

Index
So what are the pieces & what are they doing?

- **Entity classes**
  - map Java data to database
  - provides queries for getting querying table

- **Facade Bean classes**
  - provides means for Creating, Retreiving, Updating, Deleting elements
  - Done through JPA’s EntityManager

- **Session Scoped managed bean controller classes:**
  - AbstractFacade.java class that contains the business logic for creation, retrieval, modification and removal of entity instances
  - utility classes, default stylesheet, localized message properties, etc.