Objectives

- Project Code
- Debugging

Project Code Organization

Name packages in a way that avoids conflicts

ContextListener: LogicListener

- Sets up ServletContext
  - What all servlets in web app can see
- Context Parameters from web.xml
  - Set up DB Connection using parameters
- Attribute: LogicData
  - Object to get information from the DB
  - DB Connection Pool

Filter: AuthorizationFilter

- Verifies that user has the authority (permission) to access certain pages
- Requests go through filter before going to the requested Servlet
  - Write code once in the filter
  - Used by all servlets
- web.xml file says which Servlets need to be filtered

Java Model Classes

- In general, we want to minimize the use of JDBC in the Servlets
  - Same queries in multiple Servlets
  - Don’t want to duplicate code
  - If DB tables or queries change, only change in one place
- Instead, we want to have Java classes (model) that communicate with the DB
  - Convert ResultSets to objects that Servlets/JSPs can use

Java Model Classes

- LogicData
  - Holds Database Connection Pool
  - In the ServletContext
  - Entry point for Servlets to get information from the database
- Other classes to represent DB data
  - User
    - Subclasses: Student, Professor
  - Quiz, Question, Answer
Java Model Classes

- Example

LogicData class:
```java
public User getUser(String username, String password) {
    // assumes authorized user
    // use a connection from the pool of connections
    User user = new User(username, password, getConnection());
    return user;
}
```

User class:
```java
public User(String username, String password, Connection con) {
    // Populate the user's data from the database
}
```

Orbital API

- Use to handle verifying answers on quizzes
- Used mainly in Answer class
  - Example: showing that two expressions are equivalent
  - Code courtesy of Professor Levy
  - Has different syntax for conjunctions
    - & - and; | - or; ~ - not

Goals for Wednesday

- High priority:
  - Show how to create a quiz
  - Show how to take a quiz
  - Show that checking answers to be correct/incorrect
  - Login/logout
  - Challenge: symbol tables, convert to form that Orbital wants
  - Interface: using sidebar, headers, etc.
- Middle priority:
  - Create user through interface
    - Email account information?

Midterm Review

- Distributed Architecture of Web

Midterm

- Overall, did well
- Different problems were challenging to each of you
- Some general trends
**Attributes vs Parameters**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added by Java code</td>
<td>From forms, web.xml (init-param, context-param)</td>
</tr>
<tr>
<td>Type: Objects</td>
<td>Type: Strings</td>
</tr>
<tr>
<td>Associated with requests, sessions, or ServletContext</td>
<td>Associated with request, ServletConfig, ServletContext</td>
</tr>
</tbody>
</table>

**Sessions**

- One user’s requests to a Web application
  - Times out after a period of inactivity

**Life of a Servlet**

- **init**
  - Set up variables associated with a servlet
- Process requests (**doGet, doPost**)
  - May set up request, session variables
- **destroy**
  - Clean up servlet’s variables

**web.xml**

- Web app’s deployment description
- Used by web application server
- Maps Servlets to their class name
- Maps URLs to the Servlets

<table>
<thead>
<tr>
<th><code>&lt;servlet-name&gt;</code></th>
<th>canonical name of the deployed Servlet</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;servlet-class&gt;</code></td>
<td>fully qualified class name of the Servlet</td>
</tr>
<tr>
<td><code>&lt;servlet-mapping&gt;</code></td>
<td>elements maps URLs to Servlets</td>
</tr>
</tbody>
</table>

**How to Handle Request**

Midterm/Average?student=smith&ptsPoss=50&ptsEarned=47&assignment=proj2

- Implement **doGet** method
  - Know it’s a GET request because parameters are in the URL
- Get parameters from request object
  - Need to convert points parameters to numbers because they are strings
- Do calculation
- Have Response object write HTML response
  - Or forward to JSP to handle

**Testing Web Applications**

- Test cases: input - user’s data
- Verifying correctness: expected output
TODO

- Project Implementation deadline 1
  - Demo Wednesday
  - Organize code
    - Protect JSPs
    - Decide which Servlets to filter
    - Mapping of roles to permissions