Create Your Own Pizza

- **Crusts**
  - Large Original, $11
  - Medium Original, $9
  - Large Thin, $11
- **Cheese**
  - Normal cheese, no cheese (-$1)
- **Toppings** ($1 each)
  - Pepperoni, sausage, bacon, pineapple

UI – Customize Pizza

- **Crust:**
  - Large Original $11
- **Cheese:**
  - Normal
  - No cheese
- **Toppings:**
  - Pepperoni
  - Sausage
  - Bacon
  - Pineapple

UI – Review Order

<table>
<thead>
<tr>
<th>Pizza</th>
<th>Quantity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large Original Crust, Normal Cheese, with Pepperoni, Bacon</td>
<td>1</td>
<td>$13</td>
</tr>
<tr>
<td>Large Thin Crust, No Cheese, with Pineapple</td>
<td>1</td>
<td>$22</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>$35</strong></td>
</tr>
</tbody>
</table>

Create and Deploy a Web Project

- **Eclipse**
  - [http://cns.cs.calstate.edu/wiki/content/cysun/courses_materials/cs520/development](http://cns.cs.calstate.edu/wiki/content/cysun/courses_materials/cs520/development)
- **Dynamic Web Project**
  - Add JSTL jar files
  - `web.xml`
- **Deploy project on CS3**
  - Understand directory structure
  - "touch" or re-upload `web.xml` to force Tomcat to reload your application
Eclipse Dynamic Web Project

Directory Structure of an Application Root Directory
- WEB-INF
  - web.xml
  - classes
    - Compiled Java classes
  - lib
    - Additional Java libraries
- build/classes

Directory Structure on CS3
- Application Root Directory
  - JSPs and static resources
  - WEB-INF
    - web.xml
    - classes
      - Compiled Java classes
    - lib
      - Additional Java libraries
  - www

Web Application Design
- Web application = Data + Operations
- Identify the data
  - E.g. pizza, crust, topping, order, ...
- Identify the operations, e.g.
  - Add to order
  - Update order
  - Place order

MVC Architecture
- Model
- Controller
- View
- Browser
- Server

About MVC
- Models represent data in the application
- Controllers implement the actions
  - Handle user input
  - Access and process data
  - Implement business logic
  - Pass data to views for display
- Views render the display to the users

MVC Using Servlet and JSP
- Model: Bean (a.k.a. POJO)
- Controller: Servlet
- View: JSP
  - HTML, CSS, JavaScript
  - Expression Language (EL)
  - Custom tags (e.g. JSTL)
  - No scripting elements
    - <% %>
    - <%= %>
    - <% %>
Create Model Classes

- *Nouns* in the problem description usually indicate classes and fields
- Follow good OO design practice

Understand Bean Properties

- Bean properties are defined by getters and/or setters
  - E.g. `getFoo()` → `foo`
  - Read-only: only getter
  - Write-only: only setter
  - Read-write: both getter and setter
- For a boolean property, the getter starts with `is` instead of `get`
  - E.g. `isFoo()` instead of `getFoo()`

Create Controllers

- `@WebServlet`
- Servlet methods
  - `init()`, `doGet()`, and `doPost()`
- Access request parameters
- Scopes
  - Application, session, request, and page
- Forward and redirect

@WebServlet


@WebServlet Elements for URL Patterns

- `value`
  - URL pattern(s) of the servlet
  - The default element
- `urlPatterns`
  - Same purpose as `value`
  - Usually used when more than one element is specified
  - Only one of `value` and `urlPatterns` can be specified

@WebServlet Examples

```java
@WebServlet("HelloServlet")
@WebServlet("HelloServlet", "(/member/*)")
@WebServlet(name="Hello", urlPatterns=("HelloServlet", "/html") )
@WebServlet( urlPatterns="/MyPattern", 
  initParams={@WebInitParam(name="ccc", value="333")} )
```
Scopes and Their Common Usage

- Application scope
  - Store data shared by all users
- Session scope
  - Store data associated with a session, e.g., login credentials, shopping cart
- Request scope
  - Pass data from controller to view
- Page scope
  - Local variables in a JSP

Access Scoped Variables in Servlet

- Application scope
  - ServletContext
- Session scope
  - HttpSession
- Request scope
  - HttpServletRequest
- Page scope (in JSP scriptlet)
  - pageContext

Create Views

- HTML, CSS, and JavaScript
- Expression Language (EL)
- JSP Standard Tag Library (JSTL)

EL Operators

- Arithmetic
  - +, -, *, /, %
  - div, mod
- Logical
  - &&, ||, !
  - and, or, not
- Relational
  - ==, !=, <, >, <=, >=
  - eq, ne, lt, gt, le, ge
- Conditional
  - ?: empty
  - check whether a value is null or empty
- Other
  - [], ..

Implicit Objects

- pageContext
- servletContext
- session
- request
- response
- param, paramValues
- header, headerValues
- cookie
- initParam
- pageScope
- requestScope
- sessionScope
- applicationScope

Common Usage of EL

- Access scoped variables
  - ${applicationScope.foo}
  - ${sessionScope.foo}
  - ${requestScope.foo}
  - ${pageScope.foo}
  - ${foo}
- Access object properties, e.g., ${foo.bar}
- Simple operations, e.g., ${not empty param.foo}
JSP Standard Tag Library (JSTL)

<table>
<thead>
<tr>
<th>Library</th>
<th>URL</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td><a href="http://java.sun.com/jsp/jstl/core">http://java.sun.com/jsp/jstl/core</a></td>
<td>c</td>
</tr>
<tr>
<td>XML Processing</td>
<td><a href="http://java.sun.com/jsp/jstl/xml">http://java.sun.com/jsp/jstl/xml</a></td>
<td>x</td>
</tr>
<tr>
<td>I18N Formatting</td>
<td><a href="http://java.sun.com/jsp/jstl/fmt">http://java.sun.com/jsp/jstl/fmt</a></td>
<td>fmt</td>
</tr>
<tr>
<td>Database Access</td>
<td><a href="http://java.sun.com/jsp/jstl/sql">http://java.sun.com/jsp/jstl/sql</a></td>
<td>sql</td>
</tr>
<tr>
<td>Functions</td>
<td><a href="http://java.sun.com/jsp/jstl/functions">http://java.sun.com/jsp/jstl/functions</a></td>
<td>fn</td>
</tr>
</tbody>
</table>

Library URI Prefix


Flow control
- `<c:if>`
- `<c:choose>`
  - `<c:when>`
  - `<c:otherwise>`
- `<c:forEach>`
- `<c:forToken>`
- `<c:catch>`

URL
- `<c:param>`
- `<c:redirect>`
- `<c:import>`
- `<c:out>`

Flow control

Output
- `<c:out>`

Exception handling
- `<c:catch>`

Variable support
- `<c:set>`
- `<c:remove>`

Variable support

Format Date and Time

```xml
<fmt:formatDate value="${date}" type="date" />
<fmt:formatDate value="${date}" type="time" />
<fmt:formatDate value="${date}" type="both" />
<fmt:formatDate value="${date}" pattern="yyyy-MM-dd hh:mm:ss a"/>
```

See http://download.oracle.com/javaee/6/docs/api/javax/faces/FormatDate.html for the date formatting patterns.

JSTL Functions

- fn:length()
- fn:contains()
- fn:containsIgnoreCase()
- fn:startWith()
- fn:endsWith()
- fn:indexOf()
- fn:replace()
- fn:split()
- fn:join()
- fn:escapeXML()
- fn:toUpperCase()
- fn:toLowerCase()
- fn:substring()
- fn:substringAfter()
- fn:substringBefore()
- fn:trim()

Readings

- A more detailed Servlet and JSP review
  - Slides http://csns.calstatela.edu/download?fileId=44665
  - Video 1 http://mediasite.calstatela.edu/mediasite/Viewer/?peid=dce740a088454b7db9226e338dadf3381d
  - Video 2 http://mediasite.calstatela.edu/mediasite/Viewer/?peid=6a48a0d88f84a611a89993be85d1fa01d

Video 1

Video 2