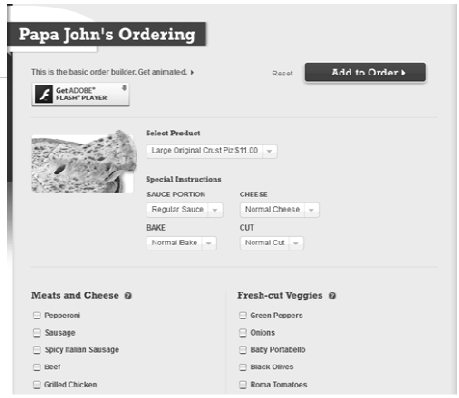
CS520 Web Programming
Servlet and JSP Review

Chengyu Sun
California State University, Los Angeles



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 ☐

Add to Order

UI – Review Order

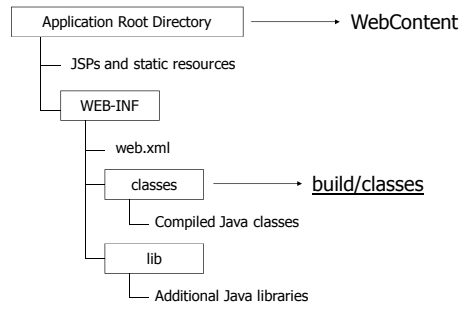
Pizza	Quantity	Price
Large Original Crust, Normal Cheese, with Pepperoni, Bacon	1	$13
Large Thin Crust, No Cheese, with Pineapple	2	$22
	Total	$35

Add Another Pizza | Update Order | Place Order

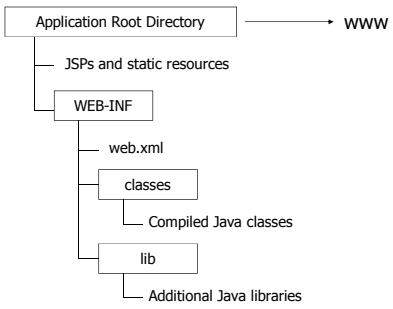
Create and Deploy a Web Project

- Eclipse
  - http://csns.calstatela.edu/wiki/content/cysun/course_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

Directory Structure of an Eclipse Dynamic Web Project



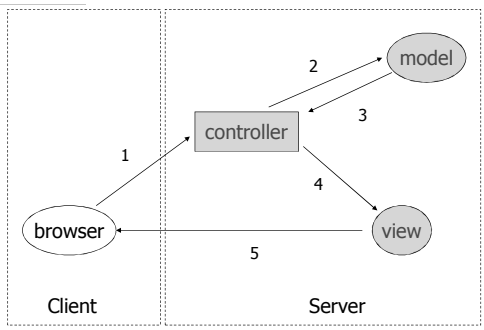
Directory Structure on CS3



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



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

- http://download.oracle.com/javaee/6/api/javax/servlet/annotation/WebServlet.html

@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

```
@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)

Library	URI	Prefix
Core	http://java.sun.com/jsp/jstl/core	c
XML Processing	http://java.sun.com/jsp/jstl/xml	x
I18N Formatting	http://java.sun.com/jsp/jstl/fmt	fmt
Database Access	http://java.sun.com/jsp/jstl/sql	sql
Functions	http://java.sun.com/jsp/jstl/functions	fn

http://java.sun.com/products/jsp/jstl/1.1/docs/tlddocs/index.html

JSTL Core

- Flow control
  - <c:if>
  - <c:choose>
    - <c:when>
    - <c:otherwise>
  - <c:forEach>
  - <c:forToken>
- Variable support
  - <c:set>
  - <c:remove>
- URL
  - <c:param>
  - <c:redirect>
  - <c:import>
  - <c:url>
- Output
  - <c:out>
- Exception handling
  - <c:catch>

Format Date and Time

```
<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/javase/6/docs/api/java/text/SimpleDateFormat.html for the date formatting patterns.

JSTL Functions

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

Readings

- A more detailed Servlet and JSP review
  - Slides http://csns.calstatela.edu/download?fileId=4466513
  - Video 1 http://mediasite.calstatela.edu/mediasite/Viewer/?peid=dc6740e088454b70b9226e388dadf3381d
  - Video 2 http://mediasite.calstatela.edu/mediasite/Viewer/?peid=6a480a0bd8f84a61a89993be85d1faf01d