**Servlets**

- A *servlet* is a Java class that responds to HTTP requests
- Used for generating dynamic web content
- Runs in a *web container*
  - Manages lifecycle of servlets
  - Handles mapping of URLs to servlets
- A web server forwards requests to the servlet container
  - Some web containers include web server functionality
Apache Tomcat

• Implementation of the Java Servlet and JavaServer Pages technologies
  – Include web server functionality but can also work together with the Apache HTTP Server
HttpServlet

- Servlet classes extend HttpServlet and implement one or more of the doGet, doPost, doPut, doHead, or doDelete methods

```java
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request,
                          HttpServletResponse response)
           throws ServletException, IOException {
        // do something...
    }
}
```
HTTP Methods

- GET, POST, PUT, HEAD, and DELETE are HTTP request types
  - GET and POST are the most common
  - In theory, GET should only retrieve data while POST submits data to be processed
  - POST can include data in request body
  - (Not the only HTTP request types)
Servlet URL Mapping

• Used to be done by XML configuration files
  – Still often done that way
• New way uses Java annotations

```java
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    // ...
}
```

A call to "http://host/webapp/login" will call the appropriate doGet or doPost, etc., method of this class.

For our lab, the webapp name is "lab-2" so the URL path would be "/lab-2/login".
• The web server uses the URL path to decide to forward to the servlet container
• The servlet container uses the URL path to decide which web app to forward to
• The web app is configured to forward each URL to a servlet class
• The servlet saves and/or retrieves data and forwards the request to the templating engine
Tomcat Settings

• By default, Tomcat runs an HTTP connector on port 8080
  – So, for our lab, the full path to the login servlet would be "http://localhost:8080/lab-2/login" when calling from the same machine

• The HTTP connector port (and other ports Tomcat uses) can be changed by editing Tomcat's server.xml file
  – The machines in the CEC labs sometimes have another program using port 8080
Servlets

• The servlet methods need to create an HTTP response
  – It is possible to write the output directly to the HttpServletResponse output stream
    • Considered bad form to return HTML data this way
  – Generally, some sort of templating engine is used to create HTML
JavaServer Pages

• JavaServer Pages (JSPs) are a popular templating engine
  – Most servlet web containers implement support for JSPs
• JSPs combine HTML template files with dynamic data to create HTML output
  – A JSP file (usually ends in ".jsp") contains HTML with special *tags* to indicate where the dynamic data goes
Login.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Bookstore Login</title>
</head>
<body>
    <h1>Bookstore Login</h1>

    <form action="${pageContext.request.contextPath}/login" method="POST">
        <p>Username: <input type="text" name="username" /></p>
        <p>Password: <input type="password" name="password" /></p>
        <p><input type="submit" name="submit" value="Submit" /></p>
    </form>
</body>
</html>
```

- The `${pageContext.request.contextPath}` is a dynamic-content tag. It will be replaced when the template engine processes this file.
Welcome.jsp

```html
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <title>Bookstore Welcome</title>
    </head>
    <body>
        <h1>Bookstore Welcome</h1>
        <p>Welcome ${user.username}!</p>
        
        <form action="" method="POST">
            <p><input type="submit" name="shop" value="Shop"
                onClick="form.action='${pageContext.request.contextPath}/list';" />
                <input type="submit" name="cart" value="Show Cart"
                onClick="form.action='${pageContext.request.contextPath}/viewCart';" />
            </p>
        </form>
    </body>
</html>
```
Servlets

• Servlet methods generally end by forwarding the request to the JSP processing engine

```java
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // do something...
    request.getRequestDispatcher("WEB-INF/jsp/Welcome.jsp")
    .forward(request, response);
}
```

– Will cause the Welcome.jsp file to be sent back as the body of the HTTP response after processing

• Tags will be replaced by actual data
Servlets

• Servlets make data available to the JSP by adding it as an attribute to the request or session object

```java
request.setAttribute("books", books);

request.getSession().setAttribute("user", user);
```
Servlets

• Request object
  – Only exists for the lifetime of this HTTP request
  – Put data here that is only needed immediately in the JSP

• Session object
  – Exists for the lifetime of the user's session
    • Usually multiple HTTP requests
  – Put data here that is needed for the entire session
  – Data can be retrieved in subsequent servlet calls in the same session
Sessions

• HTTP is a *stateless* protocol
  – Each request is independent
  – How can multiple requests be grouped into a session?
    • Send session ID as a cookie
    • Rewrite URLs to include session ID as a request parameter
Servlets

• The web container and HttpServlet superclass handle the parsing of form data
• Data from form submissions are available as parameters in the request object

String username = request.getParameter("username");
String password = request.getParameter("password");
Servlets

- Servlet classes are instantiated only once
  - The same instance of a servlet class is used for every request to that URL
  - Since requests can happen simultaneously, this means that multiple threads can be using a servlet object at the same time
  - Servlets must be reentrant
    - No instance variables!
LoginServlet.java

@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private static final Logger LOGGER = Logger.getLogger(LoginServlet.class.getName());

    @Resource(name = "jdbc/postgres")
    private DataSource dataSource;

    public LoginServlet() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        process(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        process(request, response);
    }

    ...
private void process(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    LOGGER.info("servlet /login");

    String username = request.getParameter("username");
    String password = request.getParameter("password");

    LOGGER.info("Got username " + username + " and password "+ password + "");

    if (username == null || password == null) {
        request.getRequestDispatcher("WEB-INF/jsp/Login.jsp").forward(request, response);
        return;
    }

    DatabaseManager dbm = new DatabaseManager(dataSource);
    Connection conn = dbm.getConnection();

    try {
        User user = UserDao.retrieveUser(conn, username);

        if (user == null) {
            // username not found
            request.getRequestDispatcher("WEB-INF/jsp/Login.jsp").forward(request, response);
            return;
        }
    }
}

...
// We should be salting and hashing the passwords, but we'll skip it.

if (!user.getPassword().equals(password)) {
    // password doesn't match
    request.getRequestDispatcher("WEB-INF/jsp/Login.jsp").forward(request, response);
    return;
}

request.getSession().setAttribute("user", user);

} catch (SQLException e) {
    LOGGER.log(Level.SEVERE, "error retrieving user", e);
}

finally {
    DatabaseManager.closeConnection(conn);
}

request.getRequestDispatcher("WEB-INF/jsp/Welcome.jsp").forward(request, response);

private void process(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    LOGGER.info("servlet /list");

    User user = (User) request.getSession().getAttribute("user");
    if (user == null) {
        // not properly logged in
        request.getRequestDispatcher("WEB-INF/jsp/Login.jsp").forward(request, response);
        return;
    }

    DatabaseManager dbm = new DatabaseManager(dataSource);
    Connection conn = dbm.getConnection();
    try {
        List<Book> books = BookDao.retrieveBooks(conn);
        request.setAttribute("books", books);
    } catch (SQLException e) {
        LOGGER.log(Level.SEVERE, "error retrieving books", e);
    } finally {
        DatabaseManager.closeConnection(conn);
    }

    request.getRequestDispatcher("WEB-INF/jsp/BookList.jsp").forward(request, response);
}
BookList.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix ="c" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
     "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Bookstore Books</title>
</head>
<body>
<h1>Bookstore Books</h1>
<form name="cartForm" action="${pageContext.request.contextPath}/viewBook" method="POST">
    <input type="hidden" name="isbn" value="" />
    <table>
        <tr>
            <th>Title</th>
            <th>Author</th>
            <th>ISBN</th>
            <th>Price</th>
        </tr>
    </table>
</form>
</body>
</html>
<c:forEach var="book" items="${books}">
<tr>
<td>${book.title}</td>
<td>${book.author}</td>
<td>${book.isbn}</td>
<td>${book.formattedPrice}</td>
<td><input type="button" name="${book.isbn}" value="Add to Cart"
onClick="addToCart('${book.isbn}');" /></td>
</tr>
</c:forEach>
</table>
<p><input type="submit" name="cart" value="View cart"
onClick="form.action='${pageContext.request.contextPath}/viewCart';" />
<input type="submit" name="logout" value="Logout"
onClick="form.action='${pageContext.request.contextPath}/logout';" /></p>
<script type="text/javascript">
function addToCart(isbn) {
    document.cartForm.isbn = isbn;
    document.cartForm.action = '${pageContext.request.contextPath}/addToCart';
    document.cartForm.submit();
}
</script>