Session 9

Introduction to Servlets

Lecture Objectives

- Understand the foundations for client/server Web interactions
- Understand the servlet life cycle
Reading & Reference

Reading
- Java Servlet
  en.wikipedia.org/wiki/Java_servlet
- Java Annotation
  en.wikipedia.org/wiki/Java_annotation
docs.oracle.com/javase/tutorial/java/annotations/
- Excellent tutorial, explaining how to set up a servlet in NetBeans

Reference
- Use the on-line Servlet API documentation at:
  http://docs.oracle.com/javaee/7/api/

Java Servlet Releases

- Servlet 3.0 – 2009
- Servlet 3.1 – May 2013
  (consistent with Java EE 7)
- Servlet 4.0 – September 2017
  (consistent with Java EE 8 and HTTP/2)

Java EE now maintained by the
Eclipse foundation, and renamed
as Jakarta EE
Typical Client/Server Interaction

- WWW resources, each named with a URL
- Standard content formats (e.g., HTML)
- Standard network protocols connecting any browser with any server

Server Strategies

- **Generation of HTML/CSS**
  - Server responds with a dynamically generated page that includes HTML, CSS, and data (inserted in the page)
  - Data insertion usually performed by a server-side scripting engine
- **Web services**
  - Server responds with data (no HTML and CSS)
  - Data structured based on some coordination between client and server (e.g., JSON, XML, text)
Servlets

- Conforms to the Java Servlet API
- Superseded by JAX-RS (Java API for RESTful Web Services) API
- A servlet:
  - Is a Java class that can be loaded dynamically to expand the capability of the Web server
  - Runs inside the Java Virtual Machine on the server (safe and portable)
  - Is able to access all Java APIs supported in the server
  - Does not have a main method

Servlet Implementation

- Web Server (Web Container) Market Share of active sites

<table>
<thead>
<tr>
<th>Developer</th>
<th>March 2018</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>76M</td>
<td>43.0%</td>
</tr>
<tr>
<td>nginx</td>
<td>37M</td>
<td>21.0%</td>
</tr>
<tr>
<td>Google</td>
<td>12M</td>
<td>7.7%</td>
</tr>
<tr>
<td>Microsoft</td>
<td>12M</td>
<td>6.8%</td>
</tr>
</tbody>
</table>

1. Data from NetCraft
Servlet / Web Server Interface

The Web Server static interface primarily serves resources, and passes complex request to the Web Container

Frequently both servers are packaged together, but it is good to think of them as separate systems

ServletResponse

Objects are used to pass information to the server and to return information from the server

Service methods:

protected void doGet(HttpServletRequest req,
HttpServletResponse resp)
throws ServletException, java.io.IOException

protected void doPost(HttpServletResponse resp)
throws ServletException, java.io.IOException

Response data includes HTTP version, status code, MIME type of data, document size, document
Server Generation of an HTML page

The following HTML can be returned to the browser directly by the Web server (static file on the Web Server) – or the same html page can be generated (on the fly) by the Web Container

```html
<html>
<head>
    <title>Hello World</title>
</head>
<body>
    <p>Hello World</p>
</body></html>
```

Invoking the Hello World Servlet

The URL in this link maps to the servlet

```html
<!doctype html>
<html>
<head> <title>Link to Hello World Servlet</title> </head>
<body>
<p>
        Click here to say hello to the world
    </a>
</p>
</body> </html>
```

Port 8080 is typical for a test http server

Name of the Web application

Verify the port number used by your test server

This is not really a file – it maps to your servlet
Hello World Servlet Method

protected void processRequest(
    HttpServletRequest request,
    HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    String docTyp= "<!DOCTYPE html >";
    out.println(docTyp);
    out.println("<html>");
    out.println("<head><title>" + "Hello World</title></head>");
    out.println("<meta charset="UTF-8">");
    out.println("<body>");
    out.println("<h1>Hello World</h1>");
    out.println("</body></html>");
    out.close();
}

HelloWorld Servlet Class

package lectures;
import java.io.*;import java.net.*;import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
}

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

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

    The Web container calls either doGet or doPost, which then calls processRequest
Web Application

- Your Web application contains all the servlets, JSPs, files, etc. associated with your application.
- Your Web application is stored in a directory (and deployed as a war file)
- Top level directory of the Web application is the document root of the application, containing JSP pages and static Web resources (or subdirectories of JSP, etc.)
- Document root contains a sub-directory called WEB-INF, containing
  - `web.xml` - the deployment descriptor
  - Sub-directories containing classes
  - Meta data for the Web Application

Take a look at your Web App in your NetBeans or Eclipse project pane

How to Specify the Servlet in Your HTML

- A URL is used to request that the container run your servlet (in an anchor tag or form tag)
- URL contains the host name, port (optional), and path
- In a servlet container, the path can be mapped (what you see is not always what you get)


There is no HelloWorld resource
How URLs Run Servlets

The servlet container evaluates the URL request to see if the first part of the path matches a context name.

If the path matches a context name, the context name is mapped to a Web application root directory (using the web.xml deployment descriptor).

Mapping Requests to Servlets

Two approaches:
- Web.xml (Deployment Descriptor)
- Annotation
Annotation Recap ...

- Part of Java language, starting with Java 5
- Annotations are tags that you insert into source code so that some tool can process it
  (not part of the normal execution of the program)
- Proper style places the annotation on a line preceding the statement it relates to

```java
@Entity
public class Team implements Serializable {
    // Think of it as a modifier for the declaration
    @ManyToOne(cascade=CascadeType.PERSIST)
    public Team getTeam() { ... }

    // You can annotate classes, methods, fields, and local variables
}
```

... Annotation recap

- Annotations can be defined to have elements

```java
@ManyToOne(cascade=CascadeType.PERSIST)
public Team getTeam() { ... }
```

- Examples
  - Unit testing (JUnit)
  - Java Persistence API (JPA)
  - Servlets
  - Java Beans
Annotation Example - JPA

Java Persistence allow you to specify the DB design in your code

```java
@Entity
public class Player implements Serializable {
    ...  
    @ManyToOne(cascade=CascadeType.PERSIST)
    public Team getTeam() {
        return team;
    }
}
```

Servlet Mapping with Annotations

- New with Java EE 5
- Keeps information about mapping a URL pattern to a servlet class in one place - servlet code
- Example

```java
@WebServlet(name = "HelloWorld",
    urlPatterns = {"/HelloWorld"})

public class HelloWorld extends HttpServlet {
```
## Servlet Annotations

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Optional Element and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td><code>asyncSupported</code> - Declares whether the servlet supports asynchronous operation mode.</td>
</tr>
<tr>
<td>String</td>
<td><code>description</code> - The description of the servlet</td>
</tr>
<tr>
<td>String</td>
<td><code>displayName</code> - The display name of the servlet</td>
</tr>
<tr>
<td>WebInitParam[]</td>
<td><code>initParams</code> - The init parameters of the servlet</td>
</tr>
<tr>
<td>String</td>
<td><code>largeIcon</code> - The large-icon of the servlet</td>
</tr>
<tr>
<td>int</td>
<td><code>loadOnStartup</code> - The load-on-startup order of the servlet</td>
</tr>
<tr>
<td>String</td>
<td><code>name</code> - The name of the servlet</td>
</tr>
<tr>
<td>String</td>
<td><code>smallIcon</code> - The small-icon of the servlet</td>
</tr>
<tr>
<td>String[]</td>
<td><code>urlPatterns</code> - The URL patterns of the servlet</td>
</tr>
<tr>
<td>String[]</td>
<td><code>value</code> - The URL patterns of the servlet</td>
</tr>
</tbody>
</table>

### Deployment Descriptor (web.xml)

- **Alternate (older) technique to deploy your Web application** (i.e., servlets, JSPs, etc.)
- **NetBeans display of Deployment Descriptor (below)**

You can use a URL pattern that is different from the servlet name.
web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javae" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <servlet>
    <description>A Hello World servlet</description>
    <servlet-name>HelloWorld</servlet-name>
    <servlet-class>lectures.HelloWorld</servlet-class>
  </servlet>
  ...
  <servlet-mapping>
    <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/HelloWorld</url-pattern>
  </servlet-mapping>
  ...
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```

Are We on Track?

- Create a HelloWorld servlet that displays Hello World in your browser
- Invoke it from your IDE
- Invoke it from your browser

This is just to discover anomalies in your lab or personal implementation of your IDE

Be sure you have installed the Web components of your IDE

© Robert Kelly, 2018
Were We on Track?

```java
@WebServlet(name = "HelloWorld", urlPatterns = {"/HelloWorld")
public class HelloWorld extends HttpServlet {
  protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
      out.println("<!DOCTYPE html>\n\n<html>\n<head>\ntitle=Servlet HelloWorld\n</head>\n<body>\n<h1>Hello World</h1>\n</body>\n</html>\n"; } }
```

Request Message Format

- The http request is specified by the request line, a variable number of header fields, and the entity body.

<table>
<thead>
<tr>
<th>method</th>
<th>URL</th>
<th>Version</th>
<th>cr</th>
<th>lf</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>field name</td>
<td>value</td>
<td>cr</td>
<td>lf</td>
</tr>
<tr>
<td>header</td>
<td>field name</td>
<td>value</td>
<td>cr</td>
<td>lf</td>
</tr>
<tr>
<td>header</td>
<td>field name</td>
<td>value</td>
<td>cr</td>
<td>lf</td>
</tr>
</tbody>
</table>

- request line (like a method call)
- header lines (like a parameter list of Strings)
- Entity body (like a single object parameter)
Calling Sequence

Includes all name value pairs in the form field plus server data in the URL query strings, header data, and cookies

Includes header data and all name value pairs in the request object

NetBeans uses a processRequest method (not a standard)

HttpServletRequest Methods

Some HttpServletRequest Methods

ServletResponse

getRemoteUser

getRequestedSessionId()
HttpServletResponse

Some methods:
- getWriter - from ServletResponse
- sendError(int sc)
- addCookie(Cookie cookie)
- sendRedirect(String location)

Some fields:
- SC_GONE
- SC_INTERNAL_SERVER_ERROR
- SC_NOT_FOUND

Why doGet and doPost?

HTTP - a simple stateless protocol (Web browser makes a request and the server responds)

The request from the browser specifies an HTTP method, along with client data

HTTP methods - GET, POST, HEAD, PUT, DELETE, etc.

Method called (doGet or doPost) corresponds to the HTTP method requested by the browser
**doGet / doPost Practice**

- A servlet usually does not distinguish between a GET and a POST method call
- One of the methods usually invokes the other

```java
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    this.doGet(request, response);
}
```

Or NetBeans generates a processRequest method

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

**Servlet Generation of HTML**

- A servlet will generate 2 kinds of output
  - Information about the transmission to the browser - this is stored in the http header of the response
  - Data (e.g., HTML) that is stored in the http body of the response
Server Stream Caching

- Header data – headers can be set in any order and are not sent until the first buffer fills
  - Response header data includes age, cache control, language, message digest, MIME type, expiration date, last modified date, etc.
  - isCommitted method – returns boolean indicating that headers have been sent

- Buffered stream data
  - setBufferSize
  - flushBuffer

Have You Satisfied the Lecture Objectives?

- Understand the foundations for client/server Web interactions
- Understand the servlet life cycle