Java Servlets

CSCI 201
Principles of Software Development

Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu
Outline

• Java Servlets
• Program
Dynamic Web Content

- Dynamic web content allows different data to be shown on the same web page
  - For example, the Google search result page is the same page that loads different data based on the search terms

**USC Viterbi School of Engineering**

USC Viterbi School of Engineering - Home Page

USC Viterbi School of Engineering - School Home

USC Viterbi School of Engineering - Wikipedia

**USC Viterbi Department of Computer Science**

USC Viterbi Department of Computer Science - Home Page
Front-End vs Back-End

Request web page

Return HTML, CSS, JavaScript
Web Languages

- Most dynamic content is based on a server program running and generating the front-end code in real-time
- There are many back-end web languages
  - Java – Servlets, JSPs, JavaBeans, Enterprise Java Beans (EJBs)
  - .NET – C#, VB
  - PHP
  - Ruby
  - Python
  - Server-side JavaScript – node.js
  - CGI – C, Perl
- There are three front-end web languages
  - HTML
  - CSS
  - JavaScript
  - (Browser plug-ins could be used on the front-end as well)
Which Back-End Language to Choose?

### Most Used Programming Languages (% Usage)

- Python: 23.80%
- JavaScript: 54.40%
- JAVA: 83.30%
- Ruby: 8%
- PHP: 29.70%

### Programming Language Popularity By Github Projects

- JavaScript: 30.6%
- Ruby: 15.5%
- Python: 11.8%
- C++: 11.8%
- C: 11.8%
- Go: 11.8%
- CSS: 11.8%
- Objective-C: 11.8%
- Java: 11.8%
- C#: 11.8%
- PHP: 11.8%
- Node.js: 11.8%
- AngularJS: 11.8%
- Swift: 11.8%
- TypeScript: 11.8%
- PHP: 11.8%

Note: a website may use more than one server-side programming language

Percentages of websites using various server-side programming languages

<table>
<thead>
<tr>
<th>Language</th>
<th>Usage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHP</td>
<td>82.0%</td>
</tr>
<tr>
<td>ASP.NET</td>
<td>17.3%</td>
</tr>
<tr>
<td>Java</td>
<td>2.7%</td>
</tr>
<tr>
<td>ColdFusion</td>
<td>0.7%</td>
</tr>
<tr>
<td>Perl</td>
<td>0.6%</td>
</tr>
<tr>
<td>Ruby</td>
<td>0.5%</td>
</tr>
<tr>
<td>Python</td>
<td>0.2%</td>
</tr>
<tr>
<td>JavaScript</td>
<td>0.1%</td>
</tr>
<tr>
<td>Erlang</td>
<td>0.1%</td>
</tr>
</tbody>
</table>

W3Techs.com, 21 October 2014
Which Back-End Language to Choose?
Servlet 3-Tier Architecture

Client

Server

Web/Application Server

Apache
HTTP SERVER

Apache
Tomcat

Database
Java Servlets

- Servlets are Java classes that can serve dynamic web content
- Servlets get compiled and executed on the server and generate client-side code to send back out to the browser
  - Client-side code is HTML, CSS, and JavaScript
- To create a servlet, extend the HttpServlet class
  - void doGet(HttpServletResponse req, HttpServletResponse resp)
  - void doPost(HttpServletResponse req, HttpServletResponse resp)
  - void service(HttpServletResponse req, HttpServletResponse resp)
    - Dispatches to doGet or doPost if not overridden
  - void init(ServletConfig config)
package csci201;
import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet("/FirstServlet")
public class TestServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public TestServlet() {
        super();
        System.out.println("in constructor");
    }
    public void init(ServletConfig config) throws ServletException {
        System.out.println("in init");
    }
    protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        System.out.println("in service");
    }
}
Annotations

- Annotations allow us to specify configuration parameters that are used by the application server in our .java files.
- Before annotations (or even now if you want), you would have to modify the web.xml file of the application server.

```xml
<servlet>
  <servlet-name>mytest</servlet-name>
  <init-param>
    <param-name>n1</param-name>
    <param-value>v1</param-value>
  </init-param>
  <init-param>
    <param-name>n2</param-name>
    <param-value>v2</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>mytest</servlet-name>
  <url-mapping>/myurl</url-mapping>
</servlet-mapping>
```

- Instead, we can write the following line immediately above the class declaration of our servlet:

```java
@WebServlet(name="mytest",
           urlPatterns={"/myurl"},
           initParams={
             @InitParam(name="n1",
                        value="v1"),
             @InitParam(name="n2",
                        value="v2")
           })
```
package csci201;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet("/SecondServlet")
public class TestServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;
  protected void service(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<title>My Second Servlet</title>");
    out.println("<body>");
    out.println("<h1>Hello CSCI 201</h1>");
    out.println("</body>");
    out.println("</html>");
  }
}
Color Servlet

```java
public class TestServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private String getColor(int r, int g, int b) {
        String color = "";
        color += makeHex(r);
        color += makeHex(g);
        color += makeHex(b);
        return color;
    }

    private String makeHex(int color) {
        String hexString = Integer.toHexString(color);
        if (hexString.length() == 1) {
            hexString = "0" + hexString;
        }
        return hexString;
    }

    protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>My Second Servlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Color Table</h1>");
        out.println("<table>");
        out.println("<tr><th>Red</th><th>Green</th><th>Blue</th><th>Color</th></tr>");
        for (int red=0; red < 255; red+=50) {
            for (int green=0; green < 255; green+=50) {
                for (int blue=0; blue < 255; blue+=50) {
                    String color = getColor(red, green, blue);
                    out.print("<td style="background-color: #" + color + ";">" + red + "+ " + green + "+ " + blue + "</td>" + ");
                    out.println("</tr>");
                }
            }
        }
        out.println("</table>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

<!DOCTYPE html>
<html>
<head>
<title>My Second Servlet</title>
</head>
<body>
<h1>Color Table</h1>
<table>
<tr><th>Red</th><th>Green</th><th>Blue</th><th>Color</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td style="background-color:#000000;"> </td></tr>
<tr><td>0</td><td>0</td><td>50</td><td style="background-color:#000032;"> </td></tr>
<tr><td>0</td><td>0</td><td>100</td><td style="background-color:#000064;"> </td></tr>
<tr><td>0</td><td>0</td><td>200</td><td style="background-color:#0000CC;"> </td></tr>
<tr><td>0</td><td>0</td><td>250</td><td style="background-color:#003399;"> </td></tr>
<tr><td>0</td><td>50</td><td>0</td><td style="background-color:#006633;"> </td></tr>
<tr><td>0</td><td>50</td><td>50</td><td style="background-color:#0080CC;"> </td></tr>
<tr><td>0</td><td>50</td><td>100</td><td style="background-color:#009933;"> </td></tr>
<tr><td>0</td><td>50</td><td>150</td><td style="background-color:#009966;"> </td></tr>
<tr><td>0</td><td>50</td><td>200</td><td style="background-color:#0099CC;"> </td></tr>
<tr><td>0</td><td>50</td><td>250</td><td style="background-color:#003399;"> </td></tr>
<tr><td>0</td><td>100</td><td>0</td><td style="background-color:#006633;"> </td></tr>
<tr><td>0</td><td>100</td><td>50</td><td style="background-color:#0080CC;"> </td></tr>
<tr><td>0</td><td>100</td><td>100</td><td style="background-color:#008080;"> </td></tr>
<tr><td>0</td><td>100</td><td>150</td><td style="background-color:#008099;"> </td></tr>
<tr><td>0</td><td>100</td><td>200</td><td style="background-color:#0080CC;"> </td></tr>
<tr><td>0</td><td>100</td><td>250</td><td style="background-color:#003399;"> </td></tr>
<tr><td>0</td><td>150</td><td>0</td><td style="background-color:#006633;"> </td></tr>
<tr><td>0</td><td>150</td><td>50</td><td style="background-color:#0080CC;"> </td></tr>
... 
</table>
</body>
</html>
Servlets and HTML Forms

- Servlets can be used to process the data submitted from an HTML form through the HttpServletRequest variable in the doGet, doPost, or service method.

- Here are a few of the more commonly used methods:
  - Cookie[] getCookies()
  - String getQueryString()
  - HttpSession getSession()
  - String getParameter(String)
  - Enumeration<String> getParameterNames()
  - String[] getParameterValues(String)
    - Returns all of the values associated with a specific parameter name
Servlet Form Example

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Sample Form</title>
  </head>
  <body>
    <form name="myform" method="GET" action="FormServlet">
      First Name <input type="text" name="fname" /><br />
      Last Name <input type="text" name="lname" /><br />
      <input type="submit" name="submit" value="Submit" />
    </form>
  </body>
</html>
```
Servlet Form Example

```java
// package and imports omitted for space
@WebServlet("/FormServlet")
public class FormServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;
  protected void service(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    String fname = request.getParameter("fname");
    String lname = request.getParameter("lname");
    System.out.println("fname = " + fname);
    System.out.println("lname = " + lname);

    response.setContentType("text/html");
    out.println("<html>");
    out.println("<head><title>Form Submission</title></head>");
    out.println("<body>");
    out.println("<h1>Submitted Data</h1>" );
    out.println("First Name:<strong> " + fname + "</strong>" );
    out.println("Last Name:<strong> " + lname + "</strong>" );
    out.println("</body>");
    out.println("</html>");
  }
}
```
Forwarding from a Servlet

- Because servlets have a lot of overhead when generating client-side code, forwarding to a different page is often used
  - The request and response objects can be forwarded to the page too
  - The servlet can do some processing of the data, possibly even modify or amend it, then forward to another page
  - This is separating the display and business logic (view and controller in MVC)

```java
// omitted package and import statements for space
@WebServlet("/FormServlet")
public class FormServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String username = request.getParameter("username");
        String next = "/invalidUsername.jsp";
        if (username != null && username.equals("csci201")) {
            next = "/validUsername.jsp";
        }
        RequestDispatcher dispatch = getServletContext().getRequestDispatcher(next);
        dispatch.forward(request, response);
    }
}
```
There is a popular Java Servlet framework currently called Spring Boot ([https://projects.spring.io/spring-boot/](https://projects.spring.io/spring-boot/))

- Spring Boot tries to remove all configuration and allow a programmer to focus solely on writing Java code

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run". We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.

Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' POMs to simplify your Maven configuration
- Automatically configure Spring whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration
- Absolutely no code generation and no requirement for XML configuration
More Servlets

- For more information on Servlets
  - Go to https://docs.oracle.com/javaee/7/tutorial/servlets.htm
  - Go through one of the many servlet tutorials online
Outline

• Java Servlets
• Program
Program

- Create the following form and process the submitted data with a servlet to display the page on the right with a servlet.