Servlets and Java Server Pages

Prem Melville

The Machine Learning Group
Department of Computer Sciences
The University of Texas at Austin

What's a Servlet?

- Java's answer to CGI programming
- Program runs on Web server and builds pages on the fly
- When would you use servlets?
  - Page is based on user-submitted data e.g search engines
  - Data changes frequently e.g. weather-reports
  - Page uses information from a databases e.g. on-line stores

Tomcat

- Freely available web server with servlet support
  - http://jakarta.apache.org/tomcat/
- Currently installed in (tomcat_dir)
  - /u/ml/tomcat/jakarta-tomcat-3.2.2
- Starting and stopping the server
  - `tomcat_dir/bin/startup.sh` and `shutdown.sh`
- Currently running on `titan` on port 8080
  - http://titan.cs.utexas.edu:8080/

Setting Up Your Environment

- Add to your Classpath
  - `tomcat_dir/lib/servlet.jar` and `jsp.jar`
- Place your servlet classes in
  - `tomcat_dir/classes/`
- Place your HTML and JSP files in
  - `tomcat_dir/webapps/ROOT/`

Basic Servlet Structure

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SomeServlet extends HttpServlet {
  // Handle get request
  public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    // request – access incoming HTTP headers and HTML form data
    // response - specify the HTTP response line and headers
    // (e.g. specifying the content type, setting cookies).

    PrintWriter out = response.getWriter(); //out - send content to browser
  }
}
```


A Simple Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

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

      PrintWriter out = response.getWriter();
      out.println("Hello World");
  }
}
```

Running the Servlet

- Place your classes in tomcat_dir/classes
- Run servlet using http://host/servlet/ServletName e.g.
  - http://titan.cs.utexas.edu:8080/servlet/HelloWorld
  - …/servlet/package_name.class_name
- Restart the server if you recompile

Generating HTML

```
public class HelloWWW extends HttpServlet {
  public void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
    IOException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<HTML>\n" +
          "<HEAD><TITLE>HelloWWW</TITLE></HEAD>\n" +
          "<BODY>\n" + "<H1>Hello WWW</H1>\n" +
          "</BODY></HTML>");
   }
}
```

http://titan.cs.utexas.edu:8080/servlet/HelloWWW

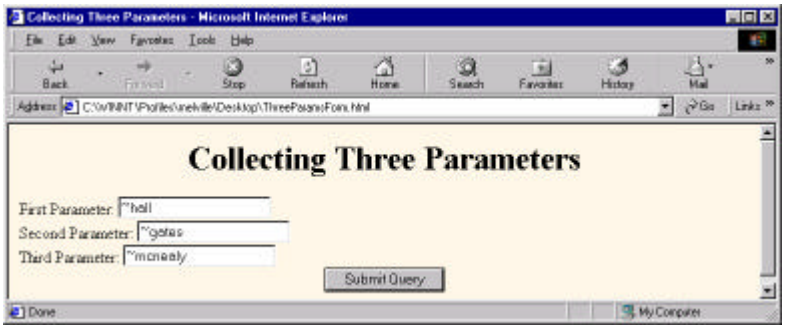
HTML Post Form

```
<FORM ACTION="/servlet/hall.ThreeParams"
      METHOD="POST">
  First Parameter:  <INPUT TYPE="TEXT"
  NAME="param1"><BR>
  Second Parameter: <INPUT TYPE="TEXT"
  NAME="param2"><BR>
  Third Parameter:  <INPUT TYPE="TEXT"
  NAME="param3"><BR>
  <CENTER>
    <INPUT TYPE="SUBMIT">
  </CENTER>
</FORM>
```

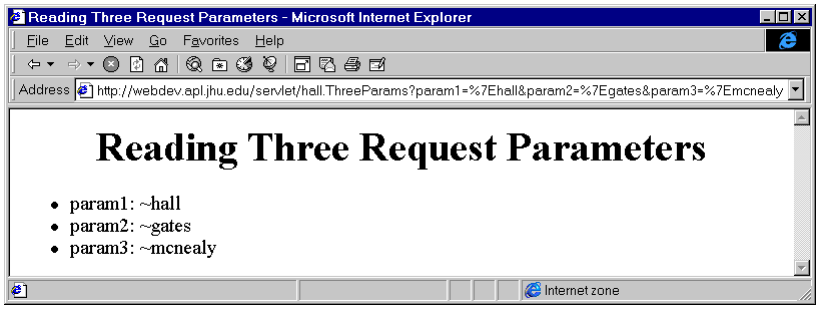

Reading Parameters

```
public class ThreeParams extends HttpServlet {
 public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println(… +"<UL>\n" +
         "<LI>param1: " + request.getParameter("param1") + "\n" +
         "<LI>param2: " + request.getParameter("param2") + "\n" +
         "<LI>param3: " + request.getParameter("param3") + "\n" +
         "</UL>\n" + …);
}
public void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
    IOException {
     doGet(request, response);
  }
}
```

Form Example



Servlet Output



Reading All Params

- `Enumeration paramNames = request.getParameterNames();`
  - parameter names in unspecified order
- `String[] paramVals = request.getParameterValues(paramName);`
  - Array of param values associated with *paramName*

Session Tracking

- Typical scenario – shopping cart in online store
- Necessary because HTTP is a "stateless" protocol
- Common solutions: Cookies and URL-rewriting

- Session Tracking API allows you to
  - look up session object associated with current request
  - create a new session object when necessary
  - look up information associated with a session
  - store information in a session
  - discard completed or abandoned sessions

Session Tracking API - I

- Looking up a session object
  - `HttpSession session = request.getSession(true);`
  - Pass *true* to create a new session if one does not exist
- Associating information with session
  - `session.setAttribute("user",request.getParameter("name"))`
  - Session attributes can be of any type
- Looking up session information
  - `String name = (String) session.getAttribute("user")`

Session Tracking API - II

- **`getId`**
  - the unique identifier generated for the session
- **`isNew`**
  - `true` if the client (browser) has never seen the session
- **`getCreationTime`**
  - time in milliseconds since session was made
- **`getLastAccessedTime`**
  - time in milliseconds since the session was last sent from client
- **`getMaxInactiveInterval`**
  - # of seconds session should go without access before being invalidated
  - negative value indicates that session should never timeout

Java Server Pages

- Mixes dynamic content with static HTML
  - Write the regular HTML
  - Enclose dynamic parts in special tags
- JSPs are equivalent to Servlets
  - Convenient if a lot of HTML is involved
- Must be located in same directory as html

JSP Syntax - I

- Expression - `<%= expression %>`
  - `Current time: <%= new java.util.Date() %>`
- Scriptlet- `<% code %>`

```
<% String queryData = request.getQueryString();
   out.println("Attached GET data: " + queryData);
%>
```

- Declaration - `<%! code %>`

```
<%! private int accessCount = 0; %>
Number of accesses to page: <%= ++accessCount %>
```

JSP Syntax - II

- Page directive - `<%@ page att="val" %>`
  - `<%@ page import="java.util.*" %>`
  - `<%@ page extends="package.class" %>`
  - `<%@ page errorPage="url" %>`
- Include directive - `<%@ include file="url" %>`
  - `<%@ include file="/navbar.html" %>`
- Predefined variables
  - `request, response, out, session`

JSP Example

```
<html>
<%@ page import="libra.*" %>

<%@ include file="/libra/Navbar.html" %>

<%= (request.getParameter("username")==null ? ""
   : (request.getParameter("username")+", "))
  %>Welcome to Libra!

<a href="/servlet/libra.Recommend">Produce
   recommendations.</a>

For more information view
<a href="/libra/help.html">help file</a>.
</html>
```

References

- Core Servlets and JavaServer Pages - Marty Hall
- Sample code
  - http://www.coreservlets.com/
- Online tutorial
  - http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/