CMPUT 391
Database Management Systems

Web based Applications, Tomcat and Servlets
- Lab 3 -
The Basic Web Server
CGI

- CGI – Common Gateway Interface
- Processing forms & Generating dynamic content (early solution)
- Spawning an external program, pass the data from the HTML form to the program
- Limitation: An expensive process (Spawning an external program)
CGI: Illustration

Browser

GET
Form.cgi

Web Server

Spawn an external program

Start
form.cgi

Web Server

CGI Program

Read environment variables

Response

Web Server

CGI Program

Response

Web Server

Response

form.cgi terminates
ASP

- ASP – Active Server Pages
- Microsoft’s answers to CGI programming
- ASP page - a Web page with code embedded inside, interpreted by the Web server
- Using special HTML tags, VBScript/JavaScript code can be inserted
Servlets

- Java technology's answer to CGI programming
- **A Java class** handling forms on Java Web servers
- Applet -- a little piece of program (client side)
- Servlet -- a little piece of program (server side)
- A servlet can do everything that a CGI program can
- Running inside the JVM along with the Web server itself
Servlets: Illustration

Browser

GET /Form

Java Web Server

Create an Instance of the servlet if it is not loaded

Form Servlet

Request

Java Web Server

Response

Form Servlet

Response

Form Servlet

Remains loaded

Lab 3

Tomcat and Servlets

CMPUT 391 – Database Management Systems
Servlets Support in Existing Servers

- Apache Web Server
  & some other Java-enabled Web servers
- Servlet engine
  Usually a separate process, not within
  the same process at the Web server
  e.g. Tomcat
Typical Servlet Engine Implementation

- **Tomcat**
  A Servlet Engine for Apache Web Server
JSP

- JSP – Java Server Pages
  - Java Version of ASP
  - A Web page with Java code embedded inside it and running on the Web server
- Java’s answer to both CGI and ASP
- Compiled into servlets
- JSP & servlets can easily interact
Tomcat

- Servlet & JSP engine for Apache Web Server
- An open-source package
  part of the Apache Software Foundation’s Jakarta project
- The Tomcat Web site:
Installing Tomcat

- Setting the CLASSPATH:
  the same classpath settings as your JVM
- Change the CLASSPATH environment variable from your login shell
  - Follow the instructions in:
    http://ugweb.cs.ualberta.ca/~c391/tutorial/softwlnstall.html
- Test your installation
Installation Tips

- Use “echo $SHELL” to double check your shell
- When editing catalina/conf/server.xml, 
  <Server ... port=...> is **Port 2** and 
  <Connector ... port=...> is **Port 1**
- Use **Port 1** when you input the URL
- Take advantage of the alias to start and stop Tomcat (commands are the same in Bourne shell and C shell)
URL Mapping

- Unlike ASP, JSP, etc., servlets have no extension:
  
  `<form name="FirstServlet" method="GET" action="firstservlet">`

- A special URL mapping is needed for each servlet you create. These mappings are defined in `web.xml` file.

- In our lab, the special URL is like:
  
  `http://<ui???>.cs.ualberta.ca:<Port1>/proj1/servlet`
Mapping URL to Machine Directory

- Catalina
  - Put your plain HTML files here
- Webapps
  - Put your servlets here
  - http://ui???.cs.ualberta.ca:portNum/proj1
    - <form name="FirstServlet" method="GET" action="firstServlet">
- proj1
- WEB-INF
  - Put your servlets here
- classes
  - http://ui???.cs.ualberta.ca:portNum/proj1/servlet
Your First Servlet

```java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class FirstServlet extends GenericServlet {
    public void service(ServletRequest request, ServletResponse response)
        throws IOException {
        // Tell the Web server that the response is HTML
        response.setContentType("text/html");

        // Get the PrintWriter for writing out the response
        PrintWriter out = response.getWriter();

        // Write the HTML back to the browser
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Welcome to CMPUT391 Lab!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
http://ugweb.cs.ualberta.ca/~c391/tutorial/examples/FirstServlet.java
```
Compiling the Servlet

- Edit the file “FirstServlet.java”
- `javac FirstServlet.java`
- Start your Tomcat

- Launch a browser, input the URL like:
  `http://ui00.cs.ualberta.ca:16410/proj1/firstservlet`
What Does Your First Servlet Look Like?

Welcome to CMPUT391 Lab!
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FirstServlet extends GenericServlet {
    public void service(ServletRequest request, ServletResponse response)
        throws IOException {

        // Tell the Web browser that the response is HTML
        response.setContentType("text/html");

        // Get the PrintWriter for writing out the response
        PrintWriter out = response.getWriter();

        // Write the HTML back to the browser
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Welcome to CMPUT391 Lab!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FirstServlet extends GenericServlet {
    public void service(ServletRequest request, ServletResponse response)
        throws IOException {

        // Tell the Web browser that the response is HTML
        response.setContentType("text/html");

        // Get the PrintWriter for writing out the response
        PrintWriter out = response.getWriter();

        // Write the HTML back to the browser
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Welcome to CMPUT391 Lab!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FirstServlet extends GenericServlet {
    public void service(ServletRequest request, ServletResponse response)
            throws IOException {

        // Tell the Web browser that the response is HTML
        response.setContentType("text/html");

        // Get the PrintWriter for writing out the response
        PrintWriter out = response.getWriter();

        // Write the HTML back to the browser
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Welcome to CMPUT391 Lab!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FirstServlet extends GenericServlet {
    public void service(ServletRequest request, ServletResponse response)
    throws IOException {

        // Tell the Web browser that the response is HTML
        response.setContentType("text/html");

        // Get the Printwriter for writing out the response
        PrintWriter out = response.getWriter();

        // Write the HTML back to the browser
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Welcome to CMPUT391 Lab!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FirstServlet extends GenericServlet {
    public void service(HttpServletRequest request, HttpServletResponse response)
        throws IOException {

        // Tell the Web browser that the response is HTML
        response.setContentType("text/html");

        // Get the PrintWriter for writing out the response
        PrintWriter out = response.getWriter();

        // Write the HTML back to the browser
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Welcome to CMPUT391 Lab!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
The *FirstServlet* In-Depth

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

public class FirstServlet extends GenericServlet {
    public void service(ServletRequest request, ServletResponse response)
        throws IOException {

        // Tell the Web browser that the response is HTML
        response.setContentType("text/html");

        // Get the PrintWriter for writing out the response
        PrintWriter out = response.getWriter();

        // Write the HTML back to the browser
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Welcome to CMPUT391 Lab!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

You don’t need to worry about closing the output stream when you have done.
Servlet Life Cycle

1. Loads the servlet (if it has not yet been loaded)
2. Initializes the servlet instance - run init method
3. Servlets accept request from Clients and return data back - run service method
4. If the engine needs to remove the servlet, it finalizes the servlet - run destroy method
Definition of The Servlet Interface

Package javax.servlet;

Public interface Servlet
{
    public void destroy();
    public ServletConfig getServletConfig();
    public String getServletInfo();
    public void init (ServletConfig config)
        throws ServletException;
    public void service (ServletRequest request,
        ServletResponse response)
        throws ServletException,
        java.io.IOException;
}

Servlet Interface

HttpServletRequest Class

Your servlet class

Tomcat and Servlets
The `HttpServlet` Class

- It has extra methods and special request-and-response objects that are geared toward the HTTP protocol
- The `HttpServlet` provides separate methods for handling the different type of HTTP requests. The two most common ones are GET and POST:
  - public void `doGet`(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, java.io.IOException;
  - public void `doPost`(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, java.io.IOException;
The *HttpServlet* Class

- The **doGet**, **doPost**, **doPut**, **doDelete** methods are of the same general form as the **service** method.
- The **service** method of *HttpServlet* looks at the type of the HTTP request and then calls the appropriate handler methods.
The Second Servlet Example

- Refer to `Asn2Sample.java` and `Asn2Sample.html` in
  `http://ugweb.cs.ualberta.ca/~c391/tutorial/servletbasics.html`

- Access servlet using the HTML form
- “Subclass” the `HttpServlet` class rather than the `GenericServlet` class
Servlet Resources

- http://java.sun.com/webservices/docs/1.0/tutorial/doc/Servlets.html
- Other resources:
  - http://ugweb.cs.ualberta.ca/~c391
Servlet Exercise

- create and populate a table in your Oracle account (At least 2 columns & 4 records)
- extend Example 3.3 in servletbasics.html
  - servlet should connect to Oracle database
  - execute the SQLStatement received from the browser (e.g. select * from “YourTable”;
  - display ResultSet on the browser(Tip: use ResultSetMetaData presented in JDBC tutorial)
  - display an error if the SQL statement was not executed correctly in Oracle (e.g. select * from “Wrong table”)