




Servlets and JSP

Ethan Cerami
New York University

3/11/2003

Intro to Servlets

1




Road Map (Servlets 1)

- Servlet Architecture Overview
- Advantages of Servlets
- Introduction to Java Server Pages (JSP)
 - Servlets v. JSP
- Generic Template for Creating Servlets
 - Servlet 2.3 API
- Hello World Examples
 - Outputting Text, HTML, and the current time.

3/11/2003

Intro to Servlets

2



Road Map (Servlets 2)

- Life of a Servlet
- Overview of Browser/Servlet Communication
- Reading Form Data from Servlets
- Example 1: Reading three parameters
- Reading HTTP Request Headers
- Generating the Server Response

3/11/2003

Intro to Servlets

3



Road Map (JSP 1)

- Introduction to JSP: Hello, JSP!
- JSP Scripting Elements
- JSP Scriptlets
- JSP Declarations

3/11/2003

Intro to Servlets

4



Road Map (JSP 2)

- Page Attributes
 - Import Attribute
 - Content Type Attribute
 - isThreadSafe Attribute
 - JSP Error Pages
- The Include File Directives

3/11/2003

Intro to Servlets

5



Road Map (Session)

- Using the Java Session API
 - Overview of what the Session API provides
 - Extracting Data from the Session
 - Adding Data to the Session
- Example: Per-Client Access Counter

3/11/2003

Intro to Servlets

6



Architectural Overview

3/11/2003

Intro to Servlets

7



What is a Servlet?

- Java's answer to the Common Gateway Interface (CGI).
- Applet: a java program that runs within the web browser.
- Servlet: a java program that runs within the web server.
- Rapidly becoming the standard for building web applications.

3/11/2003

Intro to Servlets

8

Life of a Servlet

- Regardless of the application, servlets usually carry out the following routine:
 - 1) Read any data sent by the user
 - Capture data submitted by an HTML form.
 - 2) Look up any HTTP information
 - Determine the browser version, host name of client, cookies, etc.
 - 3) Generate the Results
 - Connect to databases, connect to legacy applications, etc.

3/11/2003

Intro to Servlets

9

Life of a Servlet (cont.)

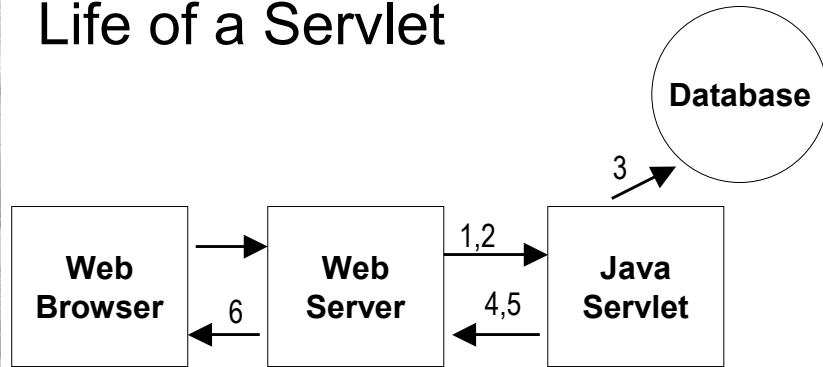
- 4) Format the Results
 - Generate HTML on the fly
- 5) Set the Appropriate HTTP headers
 - Tell the browser the type of document being returned or set any cookies.
- 6) Send the document back to the client

3/11/2003

Intro to Servlets

10

Life of a Servlet



3/11/2003

Intro to Servlets

11

Advantages of Servlets

3/11/2003

Intro to Servlets

12

Advantages of Servlets

- Servlets have six main advantages:
 - Efficient
 - Convenient
 - Powerful
 - Portable
 - Secure
 - Inexpensive

3/11/2003

Intro to Servlets

13

Advantage 1: Efficient

- For each browser request, the servlet spawns a light weight thread.
- This is faster and more efficient than spawning a new operating system process.
- Hence, servlets have better performance and better scalability than traditional CGI.

3/11/2003

Intro to Servlets

14




Advantage 2: Convenient

- Servlets include built-in functionality for:
 - Reading HTML form data
 - Handling cookies
 - Tracking user sessions
 - Setting HTTP headers
- Java is object oriented

3/11/2003

Intro to Servlets

15



Advantage 3: Powerful

- Servlets can talk directly to the web servers.
- Multiple servlets can share data:
 - Particularly important for maintaining database connections.
- Includes powerful techniques for tracking user sessions.

3/11/2003

Intro to Servlets

16

Advantage 4: Portable

- One of the advantages of Java is its portability across different operating systems.
- Servlets have the same advantages.
- You can therefore write your servlets on Windows, then deploy them on UNIX.
- You can also run any of your servlets on any Java-enabled web server, with no code changes.

3/11/2003

Intro to Servlets

17

Advantage 5: Secure

- Traditional CGI programs have a number of known security vulnerabilities.
- Hence, you usually need to include a separate Perl/CGI module to supply the necessary security protection.
- Java has a number of built-in security layers.
- Hence, servlets are considered more secure than traditional CGI programs.

3/11/2003

Intro to Servlets

18



Advantage 6: Inexpensive

- You can download free servlet kits for development use.
- You can therefore get started for free!
- Nonetheless, production strength servlet web servers can get quite expensive.



Java Server Pages

Java Server Pages

- Related to Java Servlets
- Can be used alone or in conjunction with servlets
- Represent (yet) another method for creating server side applications

3/11/2003

Intro to Servlets

21

Servlets v. JSP

- Servlets
 - code looks like a regular Java program.
- JSP
 - embed Java commands directly within HTML
- Let's examine a Servlet program next to a JSP program...
- Each of these prints, "Hello, World!"

3/11/2003

Intro to Servlets

22

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

A Java Servlet :
Looks like a regular
Java program

```
public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<HTML>");
        out.println("<HEAD><TITLE>Hello World</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("<BIG>Hello World</BIG>");
        out.println("</BODY></HTML>");
    }
}
```

3/11/2003

Intro to Servlets

23

```
<html>
<head>
<title>Hello, World JSP Example</title>
</head>
<body>
  <h2> Hello, World!
  The current time in milliseconds is
  <%= System.currentTimeMillis() %>
</h2>
</body>
</html>
```

A JSP Page :
Looks like a regular
HTML page.

↑
Embedded Java
command to
print current time.

3/11/2003

Intro to Servlets

24



Generic Servlet Template

3/11/2003

Intro to Servlets

25



Servlet Template

- First, let's take a look at a generic servlet template.
- The code does not actually do anything, but all your future servlets will follow this general structure.
- The most important pieces are noted in **yellow**.

3/11/2003

Intro to Servlets

26

```

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

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

        // Use "request" to read incoming HTTP headers
        // (e.g. cookies) and HTML form data (e.g. data the user
        // entered and submitted).

        // Use "response" to specify the HTTP response status
        // code and headers (e.g. the content type, cookies).

        PrintWriter out = response.getWriter();
        // Use "out" to send content to browser
    }
}

```

3/11/2003

Intro to Servlets

27

Generic Template

- Import the Servlet API:

```

import javax.servlet.*;
import javax.servlet.http.*;

```

- To create servlets, you must remember to always use these two import statements.

3/11/2003

Intro to Servlets

28

Generic Template

- All your servlets must extend `HTTPServlet`.
- `HTTPServlet` represents the base class for creating Servlets within the Servlet API.
- The Full Servlet API is available at:
 - <http://www.java.sun.com/products/servlet/2.3/javadoc/index.html>
- Once you have extended `HTTPServlet`, you must override one or both:
 - `doGet ()`: to capture HTTP Get Requests
 - `doPost ()`: to capture HTTP Post Requests

3/11/2003

Intro to Servlets

29

doGet and doPost

- The `doGet ()` and `doPost ()` methods each take two parameters:
 - `HttpServletRequest`: encapsulates all information regarding the browser request.
 - Form data, client host name, HTTP request headers.
 - `HttpServletResponse`: encapsulate all information regarding the servlet response.
 - HTTP Return status, outgoing cookies, HTML response.
- If you want the same servlet to handle both GET and POST, you can have `doGet` call `doPost` or vice versa.

3/11/2003

Intro to Servlets

30

Getting an OutputStream

- The `HTTPResponse` object has a `getWriter()` method.
- This method returns a `java.io.PrintWriter` object for writing data out to the Web Browser.

```
PrintWriter out = response.getWriter();
```

Hello World!

```
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");
    }
}
```

3/11/2003

Intro to Servlets

33

Output Stream

- Once you have an `OutputStream` object, you just call the `println()` method to output to the browser.
- Anything you print will display directly within the web browser.
- As we will now see, you can also output any HTML tags.

3/11/2003

Intro to Servlets

34

Generating HTML

- To generate HTML, you need to add two steps:
 - Tell the browser that you are sending back HTML.
 - Modify the `println()` statements to return valid HTML.

3/11/2003

Intro to Servlets

35

HelloWWW.java

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

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>Hello WWW</TITLE></HEAD>\n" +
            "<BODY>\n" +
            "<H1>Hello WWW</H1>\n" +
            "</BODY></HTML>");
    }
}
```

3/11/2003

Intro to Servlets

36

Generating HTML

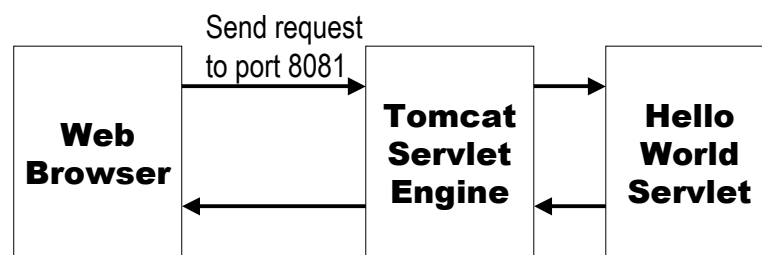
- To return HTML, you must set the content MIME type to text/html:
 - `response.setContentType("text/html");`
- Remember that you must set the content type *before* you output any content.
- Once you have set the MIME type, you can return any HTML document you want.

3/11/2003

Intro to Servlets

37


Tomcat Set-up



3/11/2003

Intro to Servlets

38




Life of a Servlet

3/11/2003

Intro to Servlets

39



Life of a Servlet

- Birth: Create and initialize the servlet
 - Important method: `init()`
- Life: Handle 0 or more client requests
 - Important method: `service()`
- Death: Destroy the servlet
 - Important method: `destroy()`

3/11/2003

Intro to Servlets

40

Life of a Servlet

- The first time a servlet is called, the Servlet is instantiated, and its `init()` method is called.
- Only one instance of the servlet is instantiated.
- This one instance handles all browser requests.

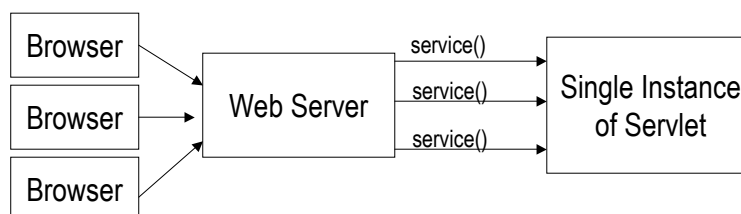
3/11/2003

Intro to Servlets

41

Service() Method

- Each time the server receives a request for a servlet, the server spawns a new thread and calls the servlet's `service ()` method.



3/11/2003

Intro to Servlets

42

Let's Prove it...

- To prove that only one instance of a servlet is created, let's create a simple example.
- The Counter Servlet keeps track of the number of times it has been accessed.
- This example maintains a single instance variable, called count.
- Each time the servlet is called, the count variable is incremented.
- If the Server created a new instance of the Servlet for each request, count would always be 0!

3/11/2003

Intro to Servlets

43

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

```
public class Counter extends HttpServlet {
    // Create an instance variable
    int count = 0;
```

```
    // Handle an HTTP GET Request
```

```
    public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws IOException, ServletException {
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        count++;
        out.println ("Since loading, this servlet has "
            + "been accessed "+ count + " times.");
        out.close();
    }
```

```
}
```

Only one instance of the counter Servlet is created. Each browser request is therefore incrementing the same count variable.

3/11/2003

Intro to Servlets

44

Death of a Servlet

- Before a server shuts down, it will call the servlet's `destroy()` method.
- You can handle any servlet clean up here. For example:
 - Updating log files.
 - Closing database connections.
 - Closing any socket connections.

3/11/2003

Intro to Servlets

45

Overview of Browser/Servlet Communication

3/11/2003

Intro to Servlets

46

Client Request Data

- When a user submits a browser request to a web server, it sends two categories of data:
 - Form Data: Data that the user explicitly typed into an HTML form.
 - For example: registration information.
 - HTTP Request Header Data: Data that is automatically appended to the HTTP Request from the client.
 - For example: cookies, browser type, browser IP address.

3/11/2003

Intro to Servlets

47

Form Data

3/11/2003

Intro to Servlets

48

Form Data

- Based on our understanding of HTML, we now know how to create user forms.
- We also know how to gather user data via all the form controls: text, password, select, checkbox, radio buttons, etc.
- Now, the question is: if I submit form data to a servlet, how do I extract this form data?
- Figuring this out forms the basis of creating interactive web applications that respond to user requests.

3/11/2003

Intro to Servlets

49

Reading Form Data from Servlets

- The `HttpServletRequest` object contains three main methods for extracting form data:
 - `getParameter()`: used to retrieve a single form parameter.
 - `getParameterValues()`: used to retrieve a list of form values, e.g. a list of selected checkboxes.
 - `getParameterNames()`: used to retrieve a full list of all parameter names submitted by the user.
- We will examine each of these and then explore several examples.

3/11/2003

Intro to Servlets

50

Reading Form Data

- All these methods work the same way regardless of whether the browser uses HTTP GET or HTTP POST.
- Remember that form elements are case sensitive. Therefore, “userName” is not the same as “username.”

3/11/2003

Intro to Servlets

51

getParameter() Method

- Used to retrieve a single form parameter.
- Possible return values:
 - String: corresponds to the form parameter.
 - Empty String: parameter exists, but has no value.
 - null: parameter does not exist.

3/11/2003

Intro to Servlets

52

getParameterValues() Method

- Used to retrieve multiple form parameters with the same name.
- For example, a series of checkboxes all have the same name, and you want to determine which ones have been selected.
- Returns an Array of Strings.
 - An array with a single empty string indicates that the form parameter exists, but has no values.
 - null: indicates that the parameter does not exist.

3/11/2003

Intro to Servlets

53

getParameterNames() method

- Returns an Enumeration object.
- By cycling through the enumeration object, you can obtain the names of all parameters submitted to the servlet.
- Note that the Servlet API does not specify the order in which parameter names appear.

3/11/2003

Intro to Servlets

54



Example 1: Reading three explicit parameters

3/11/2003

Intro to Servlets

55



Example 1

- Our first example consists of one HTML page, and one servlet.
- The HTML page contains three form parameters: param1, param2, and param3.
- The Servlet extracts these specific parameters and echoes them back to the browser.
- Before we examine the code, let's try it out:

3/11/2003

Intro to Servlets

56

```

<HTML>
<HEAD>
  <TITLE>Collecting Three Parameters</TITLE>
</HEAD>
<BODY BGCOLOR="#FDF5E6">
<H1 ALIGN="CENTER">Collecting Three Parameters</H1>

<FORM ACTION="/servlet/coreservlets.ThreeParams">
  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>

</BODY>
</HTML>

```

3/11/2003

Intro to Servlets

57

```

package coreservlets;

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

/** Simple servlet that reads three parameters from the
 * form data.
 */

public class ThreeParams extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Reading Three Request Parameters";

```

Continued....

3/11/2003

Intro to Servlets

58

```
out.println(ServletUtilities.headWithTitle(title) +
"<BODY BGCOLOR=\"#FDF5E6\"\>\n" +
"<H1 ALIGN=CENTER>" + title + "</H1>\n" +
"<UL>\n" +
" <LI><B>param1</B>:" +
+ request.getParameter("param1") + "\n" +
" <LI><B>param2</B>:" +
+ request.getParameter("param2") + "\n" +
" <LI><B>param3</B>:" +
+ request.getParameter("param3") + "\n" +
"</UL>\n" +
"</BODY></HTML>");
}
}
```

3/11/2003

Intro to Servlets

59

Reading HTTP Request Headers

3/11/2003

Intro to Servlets

60

Sample HTTP Request

- As a refresher, let's take a look at a sample HTTP Request to Yahoo.com

```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt)
Host: www.yahoo.com
Connection: Keep-Alive
Cookie: B=2td79o0sjf5r&b=2
```

3/11/2003

Intro to Servlets

61

Accessing HTTP Headers

- To access any of these Headers, the use the `HttpServletRequest.getHeader()` method.
- For example:
 - `String connection = req.getHeader("Connection");`
- To retrieve a list of all the Header Names, use the `getHeaderNames()` method.
 - `getHeaderNames()` returns an Enumeration object.
- For example:
 - `Enumeration enum = req.getHeaderNames();`

3/11/2003

Intro to Servlets

62

Additional HTTP Information

- `getMethod()`
 - Indicates the request method, e.g. GET or POST.
- `getRequestURI()`
 - Returns the part of the URL that comes after the host and port. For example, for the URL: <http://randomhost.com/servlet/search>, the request URI would be `/servlet/search`.
- `getProtocol()`
 - Returns the protocol version, e.g. HTTP/1.0 or HTTP/1.1

3/11/2003

Intro to Servlets

63

Example 1

- Our first example echoes all of the HTTP Request Information.
- First, it outputs:
 - Method
 - RequestURI
 - Protocol Version
- Then, it calls `getHeaderNames()` to retrieve a list of all HTTP Header Names.
- For each header name, it then calls `getHeader()`

3/11/2003

Intro to Servlets

64

```

package coreservlets;

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

public class ShowRequestHeaders extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String title = "Servlet Example: Showing Request Headers";
        out.println(ServletUtilities.headWithTitle(title) +
            "<BODY BGCOLOR=#FDF5E6>\n" +
            "<H1 ALIGN=CENTER>" + title + "</H1>\n" +
            "<B>Request Method: </B>" +
            request.getMethod() + "<BR>\n" +
            "<B>Request URI: </B>" +
            request.getRequestURI() + "<BR>\n" +
            "<B>Request Protocol: </B>" +
            request.getProtocol() + "<BR><BR>\n" +
            "<TABLE BORDER=1 ALIGN=CENTER>\n" +
            "<TR BGCOLOR=#FFAD00>\n" +
            "<TH>Header Name<TH>Header Value");
    }
}

```

3/11/2003

Intro to Servlets

65

```

Enumeration headerNames = request.getHeaderNames();
while(headerNames.hasMoreElements()) {
    String headerName = (String)headerNames.nextElement();
    out.println("<TR><TD>" + headerName);
    out.println("    <TD>" + request.getHeader(headerName));
}
out.println("</TABLE>\n</BODY></HTML>");
}

/** Let the same servlet handle both GET and POST. */

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

```

3/11/2003

Intro to Servlets

66

Generating the Server Response

3/11/2003

Intro to Servlets

67

Sample HTTP Response

- As a refresher, here's a sample HTTP response:

```
HTTP/1.1 200 OK
Date: Mon, 06 Dec 1999 20:54:26 GMT
Server: Apache/1.3.6 (Unix)
Last-Modified: Fri, 04 Oct 1996 14:06:11 GMT
Content-length: 327
Connection: close
Content-type: text/html
<title>Sample Homepage</title>

<h1>Welcome</h2>Hi there, this is a simple web page.
  Granted, it may...
```

3/11/2003

Intro to Servlets

68

Generating Responses

- Servlets can return any HTTP response they want.
- Useful for lots of scenarios:
 - Redirecting to another web site.
 - Restricting access to approved users.
 - Return images instead of HTML.

3/11/2003

Intro to Servlets

69

Setting the HTTP Status Code

- Normally, your Servlet will return an HTTP Status code of: 200 OK to indicate that everything went fine.
- To return a different status code, use the setStatus() method of the HttpServletResponse object.
- Be sure to set the status code *before* sending any document content to the client.

3/11/2003

Intro to Servlets

70



Introduction to JSP

3/11/2003

Intro to Servlets

71



Introduction to JSP

- JSP enables you to mix static HTML with dynamically generated content.
- Servlets look like regular Java programs.
- JSP pages look like regular HTML pages.
- JSP pages are therefore a good option for controlling the “look and feel” of the page.

3/11/2003

Intro to Servlets

72

A Simple Example

```
<HTML>
<BODY>

<H1>Hello JSP!</H1>
Current time is: <%= new java.util.Date() %>
</BODY>
</HTML>
```

3/11/2003

Intro to Servlets

73

How it works

- JSP pages look like regular HTML.
- However, the first time a JSP page is invoked, the JSP page is automatically converted to a Java servlet.
- All the static HTML of your JSP page is just output via the `PrintWriter` object, just like a regular Java servlet.

3/11/2003

Intro to Servlets

74

JSP Scripting Elements

3/11/2003

Intro to Servlets

75

JSP Comments

- JSP supports two types of comments:
- JSP Comments:
`<%-- JSP Comment --%>`
Appears only within your JSP page. Will be stripped out during processing. Hence, the end user cannot see these “hidden” comments.
- HTML Comments
`<!-- HTML comment -->`
Appears within your HTML, just like a regular HTML comment. Hence, the end user can see these comments.

3/11/2003

Intro to Servlets

76

JSP Scripting Elements

- JSP allows three types of scripting elements:
 - Expressions: Used to output a variable
`<%= expression %>`
 - Scriptlets: Used to run some small chunk of code `<% code %>`
 - Declarations: Used to declare variables and functions `<%! code %>`

3/11/2003

Intro to Servlets

77

JSP Expressions

`<%= expression %>`

- Used to output values or expressions to your JSP page.
- Expressions are evaluated at run-time (when the page is requested.)
- Simple example:
`<%= new java.util.Date() %>`
Evaluated at run-time. Hence, the time changes each time you go back.

3/11/2003

Intro to Servlets

78

Predefined Variables

- JSP comes with a number of predefined variables that you can use:
 - **request**: same as the `HttpServletRequest` object. Can be used to extract form parameters.
 - **response**: same as the `HttpServletResponse` object.
 - **session**: the `HttpSession` object (more on this later in the semester.)
 - **out**: the `PrintWriter` object for writing directly to the client.
- Let's try a simple example...

3/11/2003

Intro to Servlets

79

```
<HTML>
<HEAD>
<TITLE>JSP Expressions</TITLE>
</HEAD>

<BODY>
<H2>JSP Expressions</H2>
<UL>
  <LI>Current time: <%= new java.util.Date() %>
  <LI>Your hostname: <%= request.getRemoteHost() %>
  <LI>Your session ID: <%= session.getId() %>
  <LI>The <CODE>testParam</CODE> form parameter:
    <%= request.getParameter("testParam") %>
</UL>
</BODY>
</HTML>
```

3/11/2003

Intro to Servlets

80

Alternative XML Syntax

- Instead of using `<%= expression %>`, you can also use the alternative XML syntax.
`<jsp:expression>`
 Java expression
`</jsp:/expression>`
- Unfortunately, this is not set up on I5, but you should at least be aware of the alternative XML syntax.

JSP Scriptlets

JSP Scriptlets

`<% code %>`

- Enables you to embed small chunks of Java code within your JSP pages.
- Good for embedding for loops, while loops, if statements, etc.
- You can also set HTTP headers, content-type, and HTTP return status codes.
- Let's try a simple example...

3/11/2003

Intro to Servlets

83

```
<HTML>
<HEAD>
  <TITLE>Color Testing</TITLE>
</HEAD>
```

```
<%
String bgColor = request.getParameter("bgColor");
boolean hasExplicitColor;
if (bgColor != null) {
  hasExplicitColor = true;
} else {
  hasExplicitColor = false;
  bgColor = "WHITE";
}
%>
```

↑
Get Form
Parameter

3/11/2003

Intro to Servlets

84

```
<BODY BGCOLOR="<%= bgColor %>">
<H2 ALIGN="CENTER">Color Testing</H2>
```

```
<%
if (hasExplicitColor) {
    out.println("You supplied an explicit background color of " +
        bgColor + ".");
} else {
    out.println("Using default background color of WHITE. " +
        "Supply the bgColor request attribute to try " +
        "a standard color, an RRGGBB value, or to see " +
        "if your browser supports X11 color names.");
}
%>
```

```
</BODY>
</HTML>
```

3/11/2003

Intro to Servlets

85


More on Scriptlets

- You can also alternate between scriptlets and regular HTML.
- For example, the next program picks a random number to determine the message displayed to the user.

3/11/2003

Intro to Servlets

86



```
<HTML>
<BODY>
<H1>
<% if (Math.random() < 0.5) { %>
Have a nice day!
<% } else { %>
Have a lousy day!
<% } %>
</H1>
<BODY>
</HTML>
```

3/11/2003

Intro to Servlets

87



JSP Declarations

3/11/2003

Intro to Servlets

88

JSP Declarations

`<%! code %>`

- Enables you to define methods or variables that are defined outside of the `service()` method.
- For example, you can create methods within your JSP pages.
- Or you can define your own instance variables that are shared between JSP invocations.

3/11/2003

Intro to Servlets

89

```
<HTML>
<HEAD><TITLE>
JSP Declarations
</TITLE></HEAD>
<BODY>
<H1>JSP Declarations</H1>
```

This JSP page creates an instance variable, and keeps track of number of hits.

```
<%! private int accessCount = 0; %>
<H2>Accesses to page since server reboot:
<%= ++accessCount %></H2>

</BODY>
</HTML>
```

3/11/2003

Intro to Servlets

90

Another Example

- The next example combines all three concepts:
 - JSP Expressions
 - JSP Declarations
 - JSP Scriptlets

3/11/2003

Intro to Servlets

91

```
<HTML>
<%!
public String getGreeting (String user) {
    String greeting;
    if (user.equals ("ecerami"))
        greeting = new String ("Hello, Ethan!");
    else
        greeting = new String ("Hello there!");
    return greeting;
}
%>
<BODY>
```

Create a getGreeting()
method.

3/11/2003

Intro to Servlets

92

```
<H1>
<%
  String user = request.getParameter ("user");
  if (user == null)
    user = new String ("anonymous");
%>
<%= getGreeting (user) %>
</H1>
</BODY>
</HTML>
```

3/11/2003

Intro to Servlets

93

Page Attributes

3/11/2003

Intro to Servlets

94

Page Attributes

- So far, we have seen the following:
 - Expressions: Used to output a variable
`<%= expression %>`
 - Scriptlets: Used to run some small chunk of code
`<% code %>`
 - Declarations: Used to declare variables and functions `<%! code %>`
- Now, we move to page attributes:
 - Page attributes let you specify global settings for the entire JSP Page. `<%@ page ... %>`

3/11/2003

Intro to Servlets

95

Page Attributes Overview

- Several page attributes exist:
 - import: enables you to import Java libraries; just like the regular import statement.
 - contentType: enables you to set the contentType for your page.
 - isThreadSafe: enables you to specify threading properties.
 - errorPage: enables you to create/designate error pages.
- We will examine each of these in detail...

3/11/2003

Intro to Servlets

96



The Import Attribute

3/11/2003

Intro to Servlets

97



Import Attribute

- The import attribute of the page directive lets you specify Java packages.
- By default, the following packages are automatically included:
 - java.lang.*
 - javax.servlet.*
 - javax.servlet.jsp.*
 - javax.servlet.http.*
- If you want any other classes, you must include them via the import page directive.

3/11/2003

Intro to Servlets

98

Import Example

- For example, to include the `java.util.*` package, use the following line:

```
<%@ page import="java.util.*" %>
```
- Just like any other Java class, place page imports at the top of your page.
- The next page illustrates use of the import directive.

3/11/2003

Intro to Servlets

99

```
<HTML>
<HTML>
<HEAD>
<TITLE>The import Attribute</TITLE>
</HEAD>

<BODY>
<H2>The import Attribute</H2>
<H3>This page imports java.util.*</H3>
<!-- JSP page directive -->
<%@ page import="java.util.*" %>
```

Import Page
Directive

3/11/2003

Intro to Servlets

100

```
<UL>
<%
Vector basket = new Vector();
basket.addElement (new String("oranges"));
basket.addElement (new String("apples"));
basket.addElement (new String("bananas"));
basket.addElement (new String("strawberries"));
for (int i=0; i<basket.size(); i++) {
    String item = (String) basket.elementAt(i);
    out.println ("<LI>" + item);
}
%>
</UL></BODY></HTML>
```

3/11/2003

Intro to Servlets

101

The contentType Attribute

3/11/2003

Intro to Servlets

102

Content Type Attribute

- You can also use the page directive to set the contentType of your JSP page.
- Identical to setting the content type for Java Servlets, except:
 - Servlets: text/plain is the default contentType
 - JSP: text/html is the default contentType

3/11/2003

Intro to Servlets

103

The isThreadSafe Attribute

3/11/2003

Intro to Servlets

104

isThreadSafe Attribute

- JSP pages work just like servlets. Only one instance of your JSP page is instantiated.
- Each time a new request comes in, a new thread invokes the `service()` method of your JSP page.
- All client requests are therefore handled by a single object.

3/11/2003

Intro to Servlets

105

Threading Issues

- Because multiple threads are accessing the same object, your code must be *thread safe*.
- For example, the following code is not thread safe:

```
<%! private int idNum=0; %>
<%
String userID = "userID"+idNum;
out.println ("Your ID is: "+userID+".");
idNum = idNum + 1;
%>
```

The problem: If two users hit this JSP page at the same time, it is possible that they could get the same ID!

That's bad!

3/11/2003

Intro to Servlets

106

Dealing with Thread Issues

- To deal with the userID issue, you could use one of two options.
- Option 1: Use the Page Attribute
 - `<%@ page isThreadSafe="false" %>`
 - Indicates that the page is not thread safe. The servlet engine will therefore make sure that no two requests can simultaneously access the same servlet.
 - Servlet engine can achieve this by either queuing up requests or by creating a pool of instances, each of which handles a single request at a time.
- By default, the `isThreadSafe` is set to "true"

3/11/2003

Intro to Servlets

107

Dealing with Thread Issues

- Option 2: Use a Synchronized Block
 - To block simultaneous access to a portion of code, just add a synchronized block.
 - `synchronized (someObject) {...}`
 - For example, the next slide shows the revised userID code.
 - This new code is now thread safe.

3/11/2003

Intro to Servlets

108

Synchronized Block

```
<%! private int idNum=0; %>
<%
synchronized (this) {
    String userID = "userID"+idNum;
    out.println ("Your ID is: "+userID+".");
    idNum = idNum + 1;
}
%>
```

Problem Solved:
Only one thread
can access the
userID code
at a time.

3/11/2003

Intro to Servlets

109

JSP Error Pages

3/11/2003

Intro to Servlets

110

JSP Error Pages

- JSP pages provide a useful facility for generating error pages.
- When things go wrong, it is important to provide users with meaningful error pages.
- Showing a stack trace of your `NullPointerException` is not good customer service! ;-)

3/11/2003

Intro to Servlets

111

errorPage attribute

- The `errorPage` attribute specifies a JSP page that will process exceptions:
`<%@ page errorPage = "Relative URL" %>`
- If an exception is thrown anywhere within your page, the exception will be passed to the designated error page.
- The exception will be made available to the error page via an `exception` variable.

3/11/2003

Intro to Servlets

112

isErrorPage Attribute

- The isErrorPage attribute indicates whether or not the current page can act as the error page for another JSP Page.
- By default, isErrorPage is set to “false”
- To create an error page, set to “true”:
`<%@ page isErrorPage = “true” %>`
- Once set to true, the page has access to the `exception` variable.

3/11/2003

Intro to Servlets

113

Error Page Example

- The following two JSP pages illustrate basic error handling.
- The first page, `ComputeSpeed.jsp` performs some calculation. If an exception occurs, the exception is thrown to the designated error page.
- The second page, `SpeedErrors.jsp` displays the error, complete with a Java stack trace.

3/11/2003

Intro to Servlets

114

```

<HTML>
<HEAD><TITLE>Computing Speed</TITLE></HEAD>
<BODY>
<%@ page errorPage="SpeedErrors.jsp" %>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
    Computing Speed</TH></TR></TABLE>
<%!
// Note lack of try/catch for NumberFormatException if
// value is null or malformed.
private double toDouble(String value) {
  return(Double.valueOf(value).doubleValue());
}
%>

```

ComputeSpeed.jsp

3/11/2003

Intro to Servlets

115

```

<%
double furlongs = toDouble(request.getParameter("furlongs"));
double fortnights = toDouble(request.getParameter("fortnights"));
double speed = furlongs/fortnights;
%>

<UL>
  <LI>Distance: <%= furlongs %> furlongs.
  <LI>Time: <%= fortnights %> fortnights.
  <LI>Speed: <%= speed %> furlongs per fortnight.
</UL>

</BODY>
</HTML>

```

ComputeSpeed.jsp

3/11/2003

Intro to Servlets

116

```
<HTML>
<HEAD><TITLE>Error Computing Speed</TITLE></HEAD>
<BODY>
<%@ page isErrorPage="true" %>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
    Error Computing Speed</TH>
</TR>
<TR>
<TD>
<P>
ComputeSpeed.jsp reported the following error:
<code><%= exception %></code>. This problem occurred in the
following place:
<PRE>
<% exception.printStackTrace(new PrintWriter(out)); %>
</PRE>
</TD>
</TR>
</BODY>
</HTML>
```

SpeedErrors.jsp

3/11/2003

Intro to Servlets

117

Including JSP Files

3/11/2003

Intro to Servlets

118

Including Files

- Two options for including files:
 - Option 1: Include files at page translation time (file is inserted the first time the page is requested)
`<%@ include file="Relative URL" %>`
 - Option 2: Include files at request time (file is inserted for each client request.) Can only be used for inserting HTML files
`<jsp:include page="Relative URL" flush="true" %>`

3/11/2003

Intro to Servlets

119

Option 1

- Very useful for including: reusable JSP code, navigation bars, contact information, page counts, etc.
- For example, the following page inserts a reusable component for contact information and page counts.

3/11/2003

Intro to Servlets

120

```
<HTML>
<HEAD><TITLE>Some Random Page</TITLE></HEAD>
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
    Some Random Page</TABLE>
<P>
Information about our products and services.
<P>
Blah, blah, blah.
<P>
Yadda, yadda, yadda.
<%@ include file="ContactSection.jsp" %>
</BODY>
</HTML>
```

SomeRandomPage.jsp

3/11/2003

Intro to Servlets

121

```
<%@ page import="java.util.Date" %>
<%-- The following become fields in each servlet that
      results from a JSP page that includes this file. --%>
<%!
private int accessCount = 0;
private Date accessDate = new Date();
private String accessHost = "<|>No previous access</|>";
%>
<P>
<HR>
```

ContactSection.jsp

3/11/2003

Intro to Servlets

122

This page © 2000

```
<A HREF="http://www.my-company.com/">  
my-company.com</A>.
```

This page has been accessed <%= ++accessCount %>
times since server reboot. It was last accessed from
<%= accessHost %> at <%= accessDate %>.

```
<% accessHost = request.getRemoteHost(); %>  
<% accessDate = new Date(); %>
```

ContactSection.jsp

3/11/2003

Intro to Servlets

123

Important Note

- If you change an included JSP file, you must update the modification dates of all the JSP files that use it.
- In other words:
 - If you update ContactSection.jsp, you must resave SomeRandomPage.jsp (just modify one character, and then save.)

3/11/2003

Intro to Servlets

124

Option 2

- The `jsp:include` action includes HTML files at the time of client request.
- Can only be used for importing HTML files.
- Useful for including: common headers, footers, style sheets, navigation bars, etc.
- The next example illustrates its use...

3/11/2003

Intro to Servlets

125

```
<HTML><HEAD><TITLE>What's New</TITLE></HEAD>
<BODY>
<CENTER>
<TABLE BORDER=5>
  <TR><TH>
    What's New at JspNews.com</TABLE>
</CENTER>
<P>
Here is a summary of our four most recent news stories:
<OL>
  <LI><jsp:include page="Item1.html" flush="true" />
  <LI><jsp:include page="Item2.html" flush="true" />
  <LI><jsp:include page="Item3.html" flush="true" />
  <LI><jsp:include page="Item4.html" flush="true" />
</OL>
</BODY>
</HTML>
```

WhatsNew.jsp

3/11/2003

Intro to Servlets

126

An included HTML file: Item1.html

Bill Gates acts humble. In a startling and unexpected development, Microsoft big wig Bill Gates put on an open act of humility yesterday.

[More details...](http://www.microsoft.com/Never.html)

3/11/2003

Intro to Servlets

127

Overview of Session API Functionality

3/11/2003

Intro to Servlets

128

Overview of Session API

- Servlets include a built-in Session API.
- Enables you to very easily create applications that depend on individual user data. For example:
 - Shopping Carts
 - Personalization Services
 - Maintaining state about the user's preferences.

3/11/2003

Intro to Servlets

129

Using the Session API

- Steps to using the Java Session API
 - 1) Get the Session object from the HttpServletRequest object.
 - 2) Extract Data from the user's Session Object
 - 3) Extract information about the session object, e.g. when was the session created?
 - 4) Add data to the user's Session Object.

3/11/2003

Intro to Servlets

130

Getting a Session Object

- To get the user's session object, call the `getSession()` method of the `HttpServletRequest` class.
- Example:

```
HttpSession session = request.getSession();
```
- If this method returns null, the user does not have an existing session object.
 - For example, this may be the user's first time visiting the site.

3/11/2003

Intro to Servlets

131

Getting a Session Object

- If the user does not have an existing session, and you want to automatically create one, just pass **true** to the `getSession()` method.
- For example:

```
HttpSession session = request.getSession(true);
```
- If you want to know if this is a new session, call the `Session isNew()` method.

3/11/2003

Intro to Servlets

132

Behind the Scenes

- When you call `getSession()` there is a lot going on behind the scenes.
 - Each user is automatically assigned a unique session ID.
- How does this sessionID get to the user?
 - Option 1: If the browser supports cookies, the servlet will automatically create a session cookie, and store the session ID within the cookie.
 - Option 2: If the browser does not support cookies, the servlet will try to extract the session ID from the URL.

3/11/2003

Intro to Servlets

133

Extracting Data from the Session

3/11/2003

Intro to Servlets

134

Extracting Data From Session

- The Session object works like a Hash Table that enables you to store any type of Java object.
- You can therefore store any number of keys and their associated values.
- To extract an existing object, use the `getValue ()` method.

3/11/2003

Intro to Servlets

135

Extracting Data from Session

- The `getValue ()` method will return an Object type, so you will need to perform a type cast.
- Example:

```
Integer accessCount =  
(Integer)session.getValue("accessCount");
```

3/11/2003

Intro to Servlets

136

Extracting Data from Session

- Tip:
 - If you want to get a list of all “keys” associated with a Session, use the **getValueNames()** method.
 - This method returns an array of strings.

3/11/2003

Intro to Servlets

137

Additional Session Info.

- The Session API includes methods for determining Session specific information.
- **public String getId();**
 - Returns the unique session ID associated with this user, e.g. gj9xswvw9p
- **public boolean isNew();**
 - Indicates if the session was just created.
- **public long getCreationTime();**
 - Indicates when the session was first created.
- **public long getLastAccessedTime();**
 - Indicates when the session was last sent from the client.

3/11/2003

Intro to Servlets

138



Adding Data to the Session

3/11/2003

Intro to Servlets

139



Adding Data To Session

- To add data to a session, use the `putValue()` method, and specify the key name and value.
- Example:
 - `session.putValue("accessCount", accessCount);`
- To remove a value, you can use the **`removeValue (String name)`** method.

3/11/2003

Intro to Servlets

140

Terminating Sessions

- **public void invalidate()**
 - If the user does not return to a servlet for XX minutes*, the session is automatically invalidated and deleted.
 - If you want to manually invalidate the session, you can call **invalidate()**.
- * The exact number of minutes before automatic expiration is server dependent.

3/11/2003

Intro to Servlets

141

Encoding URLs

- If a browser does not support cookies, you need some other way to maintain the user's session ID.
- The Servlet API takes care of this for you by automatically appending the session ID to URLs if the browser does not support cookies.
- To automatically append the session ID, use the **encodeURL ()** method.

3/11/2003

Intro to Servlets

142

Encoding URLs

- Example:
 - **String url = response.encodeURL(originalURL);**
- Implementation varies for different web servers.
- And, remember that if you do this, every single URL must include the sessionID.
- Since this is hard to ensure, lots of sites (e.g. Yahoo require cookies.)

3/11/2003

Intro to Servlets

143

Example Session Code

3/11/2003

Intro to Servlets

144

Example Overview

- Our example tracks the number of visits for each unique visitor.
 - If this is a first time visit, the servlet creates an `accessCount` Integer variable and assigns it to the Session.
 - If the user has visited before, the servlet extracts the `accessCount` variable, increments it, and assigns it to the Session.
 - Servlet also displays basic information regarding the session, including: creation time and time of last access.

3/11/2003

Intro to Servlets

145

```
package coreservlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.net.*;
import java.util.*;

public class ShowSession extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String title = "Session Tracking Example";
        HttpSession session = request.getSession(true);
        String heading;
```

3/11/2003

Intro to Servlets

146

```

Integer accessCount =
    (Integer)session.getValue("accessCount");
if (accessCount == null) {
    accessCount = new Integer(0);
    heading = "Welcome, Newcomer";
} else {
    heading = "Welcome Back";
    accessCount = new Integer(accessCount.intValue() + 1);
}
session.putValue("accessCount", accessCount);
out.println(ServletUtilities.headWithTitle(title) +
    "<BODY BGCOLOR=#FDF5E6>\n" +
    "<H1 ALIGN=CENTER>" + heading + "</H1>\n" +
    "<H2>Information on Your Session:</H2>\n" +
    "<TABLE BORDER=1 ALIGN=CENTER>\n" +
    "<TR BGCOLOR=#FFAD00>\n" +

```

3/11/2003

Intro to Servlets

147

```

" <TH>Info Type<TH>Value\n" +
    "<TR>\n" +
    " <TD>ID\n" +
    " <TD>" + session.getId() + "\n" +
    "<TR>\n" +
    " <TD>Creation Time\n" +
    " <TD>" +
    new Date(session.getCreationTime()) + "\n" +
    "<TR>\n" +
    " <TD>Time of Last Access\n" +
    " <TD>" +
    new Date(session.getLastAccessedTime()) + "\n" +
    "<TR>\n" +
    " <TD>Number of Previous Accesses\n" +
    " <TD>" + accessCount + "\n" +
    "</TR>" +

```

3/11/2003

Intro to Servlets

148

```
        "</TABLE>\n" +
        "</BODY></HTML>");
    }

    /** Handle GET and POST requests identically. */
    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
        throws ServletException, IOException {
        doGet(request, response);
    }
}
```

3/11/2003

Intro to Servlets

149