

Introduction to Java Servlets

Jeff Offutt

<http://www.cs.gmu.edu/~offutt/>

SWE 642

Software Engineering for the World Wide Web

sources: Building Web Applications with UML, Conallen, Addison-Wesley
Professional Java Server Programming, Patzer, Wrox
Java for the Web with Servlets, JSP, and EJB, Kurniawan, New Riders
Web Technologies : A Computer Science Perspective, Jackson, Prentice-Hall

Web Applications

- A web application uses enabling technologies to
 1. make web site contents dynamic
 2. allow users of the system to affect business logic on the server
- Web applications allow users to affect state on the server
 - Some rule out search engines because they do not affect the server's state

A web application is a program deployed on the web

Enabling Technologies

- Web server responses can be static or dynamic
 - Static : HTML document is retrieved from the file system and returned to the client
 - Dynamic : HTML document is generated by a program in response to an HTTP request
- Dynamic web pages are created by enabling technologies

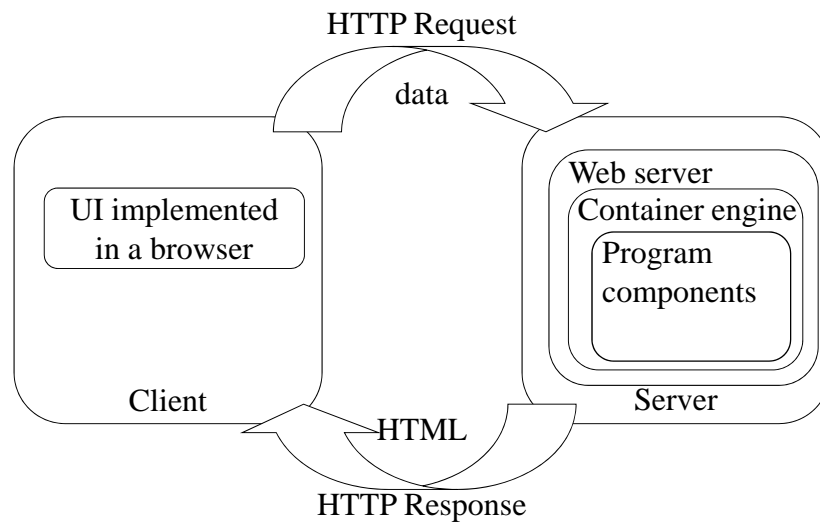
An enabling technology is a software technology that is used to make web pages interactive and responsive to user inputs

10/27/2011

© Offutt

3

Server Side Processing

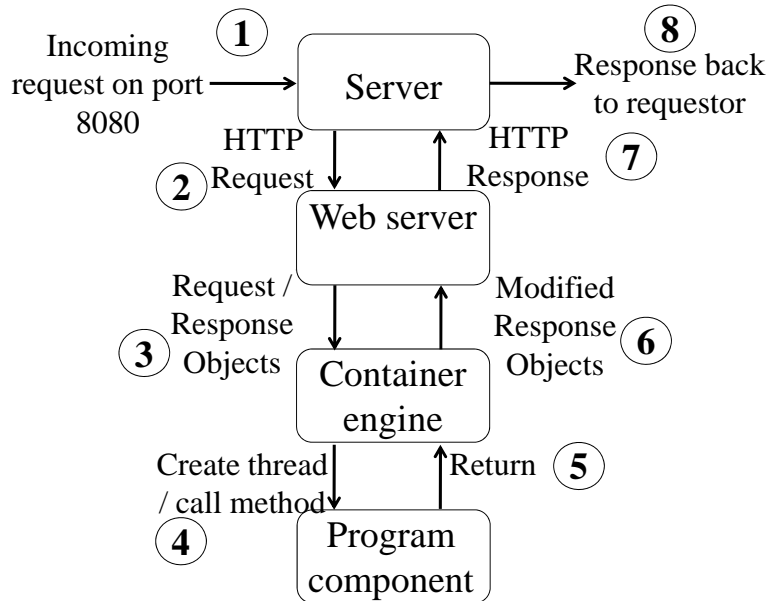


10/27/2011

© Offutt

4

Execution Overview

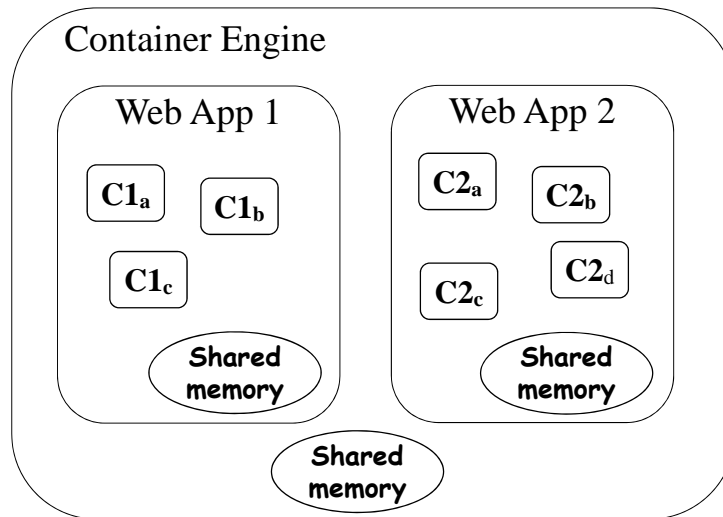


10/27/2011

© Offutt

5

Web Software Container Engine



10/27/2011

© Offutt

6

Session Management

- HTTP client-server communication is connectionless
 - As soon as the request is made and fulfilled, the connection is terminated
 - Communication is simple and resistant to network problems
- How can servers keep track of state of different clients?
 1. Session : A single coherent use of the system by the same user
 - Example : shopping carts
 2. Cookies : A string of characters that a web server places on a browser's client to keep track of a session
 - Usually used as an index into a table (dictionary) on the server
 - Most dictionaries expire after a period of time (15 to 30 minutes)

We will come back to this later ...

10/27/2011

© Offutt

7

Enabling Technologies – CGI

- CGI : Common Gateway Interface allows clients to execute applications on the server
- The first enabling technology
- CGI applications usually reside in a special “safe” directory
- Can be written in any language; PERL was most common
- CGI apps typically :
 1. process data
 2. modify server state
 3. return information (usually an HTML page)

10/27/2011

© Offutt

8

Enabling Technologies Problems with CGI

- Each and every execution of a CGI module requires a new process on the web server
- CGI posed some significant security risks
- CGI does not automatically provide session management services
- Perl is a difficult language in which to write large applications

Solution : Plug-ins on the Web server

10/27/2011

© Offutt

9

Enabling Technologies Web Server Plug-ins

- A plug-in is an extension to a web server that allows a different program to handle certain types of requests
 - images, sound, video
 - compiled module applications
 - scripted page applications
- Plug-ins typically keep an active process as long as the web server is active

10/27/2011

© Offutt

10

Enabling Technologies - Plug-ins

Compiled Modules

- Compiled modules are executable program components that the server uses
- Common compiled module application plug-ins :
 - Microsoft's .NET ASP
 - J2EE Java servlets
- Compiled modules are efficient and very effective
- They allow programmers to clearly separate the front-end from the back-end
 - Aids design
 - Complicates implementation

10/27/2011

© Offutt

11

Enabling Technologies - Plug-ins

Scripted Pages

- Scripted pages look like HTML pages that happen to process business logic
- Execution is on the server, not on the client
 - unlike JavaScripts
- They have HTML with program statements that get and process data
- JSPs are compiled and run as servlets
 - very clean and efficient

10/27/2011

© Offutt

12

Enabling Technologies - Plug-ins

Scripted Pages (2)

- Common scripted pages:
 - Adobe's ColdFusion
 - Microsoft's Active Server Pages (ASP)
 - Java Server Pages (JSP)
- Scripted pages are generally easy to develop and deploy
- They mix logic with HTML, so can be difficult to read and maintain
- Not as effective for heavy-duty engineering

10/27/2011

© Offutt

13

Summary Web Programming

- The major difference is deployment
 - Software is deployed across the Web using HTTP
 - Other deployment methods include bundling, shrink-wrapping, embedding, and contracting
- New software technologies
- New conceptual language constructs for programming
 - Integration
 - Data management
 - Control connections

These differences affects every aspect of how to engineer high quality software

10/27/2011

© Offutt

14

What are Servlets?

- Servlets are small Java classes that
 - Process an HTTP request
 - Return an HTTP response
- Servlet container or engine
 - Connects to network
 - Catches requests
 - Produces responses
 - Creates object instances of servlet classes
 - Hands requests to the appropriate object
- Programmers use an API to write servlet classes

10/27/2011

© Offutt

15

Servlets vs. Java Applications

- Servlets do not have a **main()**
 - The **main()** is in the server
 - Entry point to servlet is via call to a method (**doGet()** or **doPost()**)
- Servlet interaction with end user is indirect via request / response object APIs
 - Actual HTTP request / response processing is handled by the server
- Servlet output is usually HTML

10/27/2011

© Offutt

16

Servlet Container (or Engine)

- Servlet container is a plug-in for handling Java servlets
- A servlet container has five jobs :
 1. Creates servlet instance
 2. Calls `init()`
 3. Calls `service()` whenever a request is made
 1. `service()` calls a method written by a programmer to handle the request
 2. `doGet()` to handle GET requests, `doPost()` to handle POST requests
 3. More on this later ...
 4. Calls `destroy()` before deleting the servlet object
 5. Destroys instance

10/27/2011

© Offutt

17

Servlet Container (2)

When a request comes to a servlet, the servlet container does one of two things:

1. If there is an active object for the servlet, the container creates a Java thread to handle the request
2. If there is no active object for the servlet, the container instantiates a new object of that class, then creates a Java thread on the object to handle the request

10/27/2011

© Offutt

18

Servlet Container (3)

A servlet instance runs until the container decides to destroy it :

- When it gets destroyed is not specified by the servlet rules
- Most servlet containers destroy the object *N minutes* after the last request
- *N* is usually 15 or 30, and can be set by the system administrator
- Container can also be configured to never destroy a servlet object

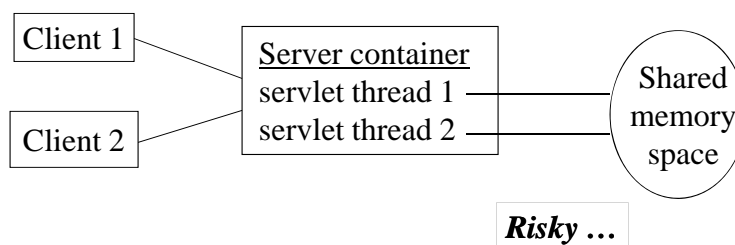
10/27/2011

© Offutt

19

Servlet Container (4)

- What if the same servlet gets multiple requests ?
- More than one thread for the same servlet may be running at the same time, using the same memory



10/27/2011

© Offutt

20

Servlet Container (5)

- By default, there is only one instance of a servlet class per servlet definition in the servlet container
- Distributable : If the application is *distributable*, there is one instance of a servlet class per virtual machine
 - Sometimes each VM is on a different computer in the cluster
 - Sometimes multiple VMs are on one computer

10/27/2011

© Offutt

21

Allowing Concurrency (SingleThreadModel)

- Container may send multiple requests to a single instance, using Java threads
 - Simply put, threads are Java's concurrency mechanism
- Thus, your service methods (**doGet**, **doPost**, etc.) should be thread-safe
 - Loosely speaking, "*thread-safe*" means that if two or more requests are running at the same time, they will not interfere with each other
- If the service methods are not thread-safe, use the **SingleThreadModel**

10/27/2011

© Offutt

22

SingleThreadModel (2)

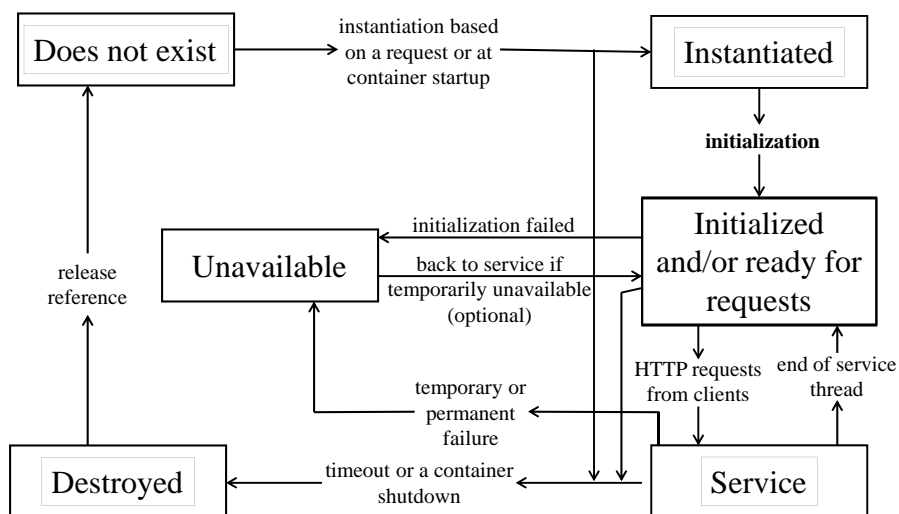
- The `SingleThreadModel` ensures that only one thread may execute the `service()` method at a time
- Containers may implement this in two ways :
 1. Instance Pooling : A “pool” of several servlet instances are available that do not share memory
 2. Request Serialization : Only one request is handled at a time
 3. Combination : A pool is used, and if there are more requests than servlets in the pool, they are serialized
- This is resource intensive and can be slow
- Better to synchronize only the statements or methods that might interfere with each other

10/27/2011

© Offutt

23

Servlet Object Thread Lifecycle UML State Diagram (Kurniawan)



10/27/2011

© Offutt

24

Common Servlet Containers

- Tomcat (open source, most common, installed on hermes)
- Oracle's WebLogic
- IBM's WebSphere (uses Tomcat)
- Adobe's JRun
- JBoss (open source)
- Wikipedia has a longer list:
http://en.wikipedia.org/wiki/List_of_Servlet_containers

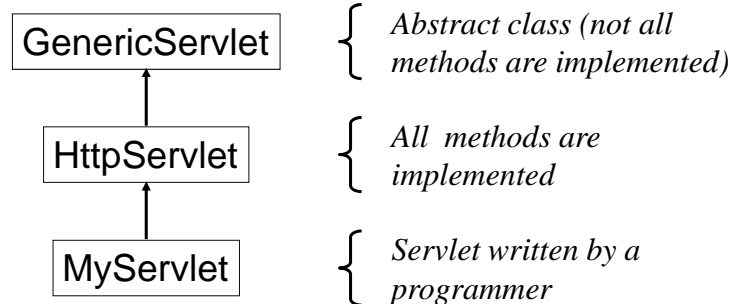
10/27/2011

© Offutt

25

Servlet API

- `javax.servlet` – primarily for containers
- `javax.servlet.http` – methods to service requests



10/27/2011

© Offutt

26

Generic Servlet & HTTP Servlet

Servlets can have five principal methods :

1. `init()` – called when servlet starts
2. `service()` – called to process requests
3. `destroy()` – called before servlet process ends
4. `getServletConfig()` – servlet can access information about servlet container
5. `getServletInfo()` – servlet container can access info about servlet

10/27/2011

© Offutt

27

Generic Servlet & HTTP Servlet (2)

- These methods are defined by the library classes **GenericServlet** and **HttpServlet**
- We write servlets by extending (inheriting from) them
- **GenericServlet** does not implement `service()`
(it is abstract)
- **HttpServlet** extends **GenericServlet** with :
`service (HttpServletRequest req, HttpServletResponse res)`
throws `ServletException, IOException`

10/27/2011

© Offutt

28

1. `init ()`

- Read configuration data
- Read initialization parameters (`javax.servlet.ServletConfig`)
- Initialize services :
 - Database driver
 - Connection pool
 - Logging service
- Seldom used in simple applications

10/27/2011

© Offutt

29

2. `service ()`

- The entry point for the servlet – this is the method that is called from the servlet container
- Called after the initialization (`init ()`)
- Primary purpose is to decide what type of request is coming in and then call the appropriate method
 - `doGet ()`
 - `doPost ()`
 - ...

10/27/2011

© Offutt

30

Types of HTTP Requests

• GET	<code>doGet ()</code>	} same signatures as <code>service()</code>
• POST	<code>doPost ()</code>	
• HEAD	<code>doHead ()</code>	
• OPTIONS	<code>doOptions ()</code>	
• DELETE	<code>doDelete ()</code>	
• PUT	<code>doPut ()</code>	
• TRACE	<code>doTrace()</code>	

10/27/2011

© Offutt

31

Types of HTTP Requests (2)

- **HttpServlet** implements these methods as “stubs” that print error messages

```
doGet ()
{
    print ("Error HTTP 405, doGet() not implemented");
}
```

- Implement servlets by overriding these methods
 - Usually `doGet()` and `doPost()`

10/27/2011

© Offutt

32

3. `destroy()`

- Called by container before the servlet instance is killed
- The threads from the `service()` method are given time to terminate before `destroy()` is called
- Can be used to clean up the state of the servlet :
 - Un-registering a database driver
 - Closing a connection pool
 - Informing another application the servlet is stopping
 - Saving state from the servlet

10/27/2011

© Offutt

33

4. `getServletConfig()`

- Returns a `ServletConfig` object, which stores information about the servlet's configuration
- The `ServletConfig` object was passed into `init()`

10/27/2011

© Offutt

34

5. `getServletInfo ()`

- Returns a **String** object that stores information about the servlet :
 - Author
 - Creation date
 - Description
 - Usage
 - ...
- This string should be formatted for human readability

10/27/2011

© Offutt

35

Simple Servlet Example

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class hello extends HttpServlet
{
    public void doGet (HttpServletRequest req,
                      HttpServletResponse res)
                      throws ServletException, IOException
    {
        res.setContentType ("text/html; charset=\"UTF-8\"");
        PrintWriter out = res.getWriter ();
        out.println ("<HTML>");
    }
}
```

10/27/2011

© Offutt

36

Simple Servlet (2)

```
out.println("<HEAD>");
out.println("<TITLE>Servlet example</TITLE>");
out.println("</HEAD>");
out.println("<BODY>");
out.println("<P>My first servlet. </P>");
out.println("</BODY>");
out.println("</HTML>");
out.close ();
} // end doGet()
} // end hello
```

10/27/2011

© Offutt

37

Servlet Parameters – requests

Parameters are conveniently stored in objects

- **String req.getParameter (String KEY)**
 - Returns value of field with the name = KEY
 - Names are defined in HTML, and values supplied by the users
- **String[] req.getParameterValues (String KEY)**
 - Returns all values of KEY
 - For example checkboxes
- **Enumeration req.getParameterNames ()**
 - Returns an Enumeration object with a list of all parameter names
- **String req.getQueryString ()**
 - Returns the entire query string

10/27/2011

© Offutt

38

Servlet Parameters—Transmission

- Parameter data is the Web analog of arguments in a method call :
 - `System.out.println ("aString");`
 - `http://www.example.com/servlet/PrintThis?arg=aString`
- Query string syntax and semantics
 - Multiple parameters are separated by '&'
`http://www.example.com/servlet/PrintThis?color=red&arg=aString`
 - Order of parameters does not matter
`http://www.example.com/servlet/PrintThis?arg=aString&color=red`
 - All parameter values are strings
`http://www.example.com/servlet/PrintThis?arg=&age=39`
Empty string

10/27/2011

© Offutt

39

Servlet Parameters—Creation

- HTML forms generate query strings when submitted
- Parameter names are specified as the value of the *name* attributes in the form controls
 - `<input type="text" name="username" size="35" />`
- Parameter values depend on control type
 - `<input type="checkbox" name="daysFree" value="Mon" />Mon`
Value sent to the server :
daysFree=Mon

10/27/2011

© Offutt

40

Servlet Parameters—Creation

Controls	Value
input/text input/password textarea	Text that the user has entered into the control field when the form is submitted
input/checkbox input/radio input/submit input/image button/submit	String assigned to the value attribute in the HTML tag The control must be selected or clicked for the parameter to be sent
input/hidden	String assigned to the value attribute Not rendered on the client
select	String assigned to the value attribute of the selected options, or content of any selected option for which the value is not defined

10/27/2011

© Offutt

41

Servlet Parameters—Creation

Enter your name: `<input type="text" name="username" size="35" />`
`<p>`

Check all the days that you are free:

`<label>`

`<input type="checkbox" name="daysFree" value="Mon" />Mon`

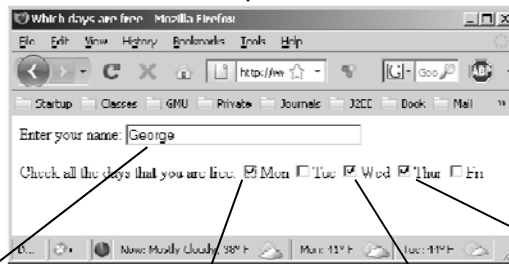
`<input type="checkbox" name="daysFree" value="Tue" />Tue`

`<input type="checkbox" name="daysFree" value="Wed" />Wed`

`<input type="checkbox" name="daysFree" value="Thur" />Thur`

`<input type="checkbox" name="daysFree" value="Fri" />Fri`

`</label>`



`username=George&daysFree=Mon&daysFree=Wed&daysFree=Thur`

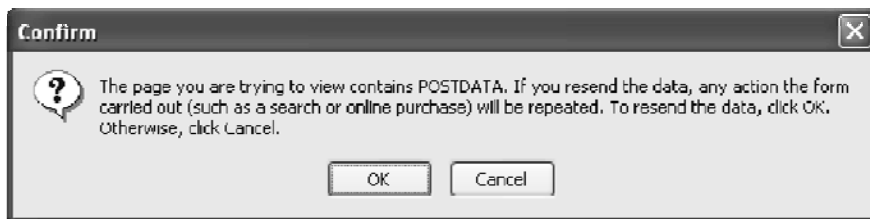
10/27/2011

© Offutt

42

Servlet Parameters—Re-Transmission

- Most browsers give a warning before submitting POST data for the second time
 - Avoid duplicate submissions and updates
 - Avoid duplicate purchases
- Users should be very careful before overriding this hesitation
- However ... how many users understand this message?



10/27/2011

© Offutt

43

Servlet Output – responses

Standard output is sent directly back to the client browser

- `res.setContentType (String type)`
 - “`text/html; charset=\“UTF-8\”`” is an HTML page with robust encoding
- `PrintWriter res.getWriter()`
 - Use `print()` and `println()` to write HTML to browser

10/27/2011

© Offutt

44

Servlet Performance

- Some servlets will run a lot
- Servlets run as *lightweight threads*, so are fast
- The network speeds usually dominate, but two techniques can add speed:
 - avoid concatenation (“+”), a slow operation
 - `out.flush()` – Sends current output to user’s screen while servlet continues processing

10/27/2011

© Offutt

45

GET and POST Requests

- An HTTP GET request is generated when the URL is entered directly into the browser’s address field
 - `doGet ()` is called from `service()`
- An HTML form can generate either a GET or a POST request
 - “... Method=POST” or “... Method=GET”
- GET requests put form data on the URL as parameters
 - `http://www ... /RunForm?NAME=Jeff&TITLE=prof`
- The length of GET parameters is limited by some browsers
- POST requests put form data in body of request
- POST requests can be arbitrarily long

10/27/2011

© Offutt

46

GET and POST Requests (2)

- Book says :
 - Use GET to retrieve data
 - Use POST to change state on server (update file or DB)
 - Use POST when there are a lot of data items
- This is a little ambiguous and incomplete ...
- Prof's suggestion :
 - Use POST when sending data to server
 - Use GET when no data is sent
- GET is also useful when the entire request needs to be bookmarked
 - Google maps

10/27/2011

© Offutt

47

GET and POST Requests (3)

If a servlet is primarily based on processing data and it uses POST, good engineering says to implement a simple `doGet()` method as a filler :

```
...  
<BODY>  
<CENTER>A Title ...</CENTER>  
<HR>  
  
<P>  
You should run this from  
<A Href="http://... .html"> http://... .html</A>  
</BODY>
```

10/27/2011

© Offutt

48

Sending Mail Messages from Servlets

Common to gather data from form and send through email

- Import mail utilities:
 - `import sun.net.smtp.SmtpClient;`
- Setup mail header :
 - `send = new SmtpClient ("gmu.edu");`
 - `send.from ("offutt@gmu.edu");`
 - `send.to ("offutt@gmu.edu");`
- Send message :
 - `out = send.startMessage ();`
 - `out.println ("... message header and body ...");`
 - `out.flush ();`
 - `out.close ();`
 - `out.closeServer ();`

10/27/2011

© Offutt

49

Sending Mail Messages (2)

- This is the simplest mechanism for sending email, but is not very powerful
 - Plus, my compiler keeps complaining that it might go away soon
 - Okay, javac has been complaining about this for 7 years now
- JavaMail is a collection of abstract library classes for handling mail with a number of different protocols

10/27/2011

© Offutt

50

Redirecting to Another URL from Servlets

Servlets usually generate an HTML file as a response, but sometimes you may want to send the client to a different servlet

- `res.sendRedirect ("http://www.cs.gmu.edu/...");`
- Do not need to set content type (`setContentType()`)
- The client will be “sent” to the specified URL
 - Server tells the client to generate another request to the new URL
 - Browser then repeats request to the new URL
 - Invisible to users ... mostly ... both requests are logged in browsers’ history list

This is a new *control* mechanism, similar to a method call

10/27/2011

© Offutt

51

Writing to Files From Servlets

Common job is to save data into a file

- File must be in a publicly writeable directory :
 - `/apps/tomcat/swe64201/WEB-INF/data/`
 - This directory is available to all of us on hermes
- Open a file, write to it, and close it:
 - `FileWriter outfile = new FileWriter`
`("/apps/tomcat/swe64201/WEB-INF/data/info-file.txt");`
 - `outfile.write (... the data to save ...);`
 - `outfile.close ();`
- Open a file in append mode:
 - `FileWriter outfile = new FileWriter`
`("/apps/tomcat/swe64201/WEB-INF/data/info-file.txt", true);`
- Remember Unix / Windows path differences !!
- Remember that we all share the same directory ... include your user name as part of the file name !!!!

10/27/2011

© Offutt

52

Deployment Testing

- Development and deployment computers often differ
- Web apps must be tested on final deployment platform
 - Must test just as real users use it
- Issues to check for :
 - Different platforms (DOS / Unix / Linux / Mac ...)
 - File names and path names (local/nonlocal, DOS/Unix)
 - Upper case dependencies
 - Incomplete deployment
 - Compiler and runtime system version
 - Permissions (data and DB)

10/27/2011

© Offutt

53

Summary—Examples

<http://www.cs.gmu.edu/~offutt/classes/642/examples/servlets/>

1. hello : Prints lots of hellos
2. name : Accepts and prints a name from a form
3. goldGetPost : Differences between GET and POST
4. formHandler : Displays arbitrary data from a form
5. twoButtons : Processing two submit buttons
6. abstracts : Processes form data and sends through email
7. loan : Compute time to pay off a loan
8. convert : Convert values
9. convert2 : Better value conversion
10. fileLoad : Uploads a file to a server
11. studInfo : Our student info system – small web app
12. showRequestHeaders : Shows information about the requests

10/27/2011

© Offutt

54