

Introduction to Servlets

Thanks: Ethan Cerami (New York, University)

1/19/01

First Servlet

1

Generic Servlet Template

1/19/01

First Servlet

2

The Servlet Development Kits

- To develop servlets, you will need three pieces of software:
 - Text Pad: a simple, text editor.
 - Java 2 Software Development Kit, Version 1.3
 - Java Servlet Development Kit
 - Contains the Servlet Runner for running Servlets on your own machine.
- The tutorial includes instructions on downloading and installing each of these.

1/19/01

First Servlet

3

Servlet Template

```
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
    }
}
```

1/19/01

First Servlet

4

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.

1/19/01

First Servlet

5

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/2.2/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

1/19/01

First Servlet

6

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.

1/19/01

First Servlet

7

Getting an OutputStream

- The HttpServletResponse 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();
```

1/19/01

First Servlet

8



Hello World!

1/19/01

First Servlet

9

```
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");
    }
}
```

1/19/01

First Servlet

10

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

1/19/01

First Servlet

11

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.

1/19/01

First Servlet

12

Invoking a Packaged Servlet

- To invoke a packaged servlet, you need to specify the package name and the servlet name:
 - <http://host/servlet/packageName.servletName>
- For example, to access the HelloWWW2 servlet on ecerami.com:
 - <http://ecerami.com/servlet/coreservlets.HelloWWW2>

1/19/01

First Servlet

13

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()`

1/19/01

First Servlet

14

Birth of a Servlet

1/19/01

First Servlet

15

The `init()` method

- The `init()` method is called when the servlet is first requested by a browser request.
- It is not called again for each request.
- Used for one-time initialization.
- There are two versions of the `init()` method:
 - Version 1: takes no arguments
 - Version 2: takes a **`ServletConfig`** object as an argument.

1/19/01

First Servlet

16

Property Files

- To understand the difference between the two `init()` options, you need to first understand property files.
- All Web Servers/Servlet Runners maintain a central properties file for storing constants.
- You can add your own properties in `servlet.properties` file. For example:
 - Database settings, user names, passwords, URLs, etc.

1/19/01

First Servlet

17

`servlet.properties` file

```
# $Id: servlets.properties,v 1.2 1999/04/02 02:04:01 duncan Exp $  
  
# Define servlets here  
  
# <servletname>.code=<servletclass>  
# <servletname>.initparams=<name=value>,<name=value>  
  
snoop.code=SnoopServlet  
snoop.initparams=initarg1=foo,initarg2=bar
```

Example Initialization Parameters

1/19/01

First Servlet

18

■ servlet.properties Rules

- To add your own properties, you need to follow the servlet.properties rules.
- You first need to register your servlet within the property file:
`<servletname>.code=<servletclass>`

1/19/01

First Servlet

19

■ servlet.properties Rules

- You can then add your own properties:
`<servletname>.initparams=<name=value>,<name=value>`
- For example, the following registers the Birth servlet, and sets its password parameter to "bluemoon":
`Birth.code=Birth`
`Birth.initparams=password=bluemoon`

1/19/01

First Servlet

20

■ Version 1: init() method

- No parameters
- Used when the servlet does not need to read any property files.
- Here's an example:

```
public void init() throws ServletException {  
    ...  
}
```

1/19/01

First Servlet

21

■ Version 2: init() method

- Used when the servlet needs to read from a properties file.
- Here's an example:

```
public void init (ServletConfig config)  
    throws ServletException {  
    super.init (config);  
    ...  
}
```

1/19/01

First Servlet

22

■ ServletConfig Object

- Provides access to the servlet properties file.
- Has a `getInitParameter()` method for retrieving specific properties.
- For example:

```
String message = config.getParameter ("message");  
String password = config.getInitParameter ("password");
```

1/19/01

First Servlet

23

■ Version 2: init() method Cont.

- Let's examine version 2 again:

```
public void init (ServletConfig config)  
    throws ServletException {  
    super.init (config);  
    ...  
}
```
- It is important to call `super.init()`.
- The `init()` method of the superclass registers the `ServletConfig` object so you can access it later.
- Therefore, if you do not call `super.init()`, you will never have access to the `ServletConfig` object.

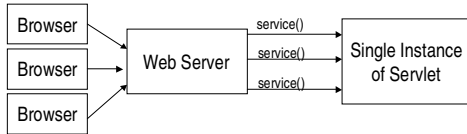
1/19/01

First Servlet

24

Service() Method

- Only one instance of a servlet is generated, which is used multiple times
- Each time the server receives a request for a servlet, the server spawns a new thread and calls the servlet's service () method.



1/19/01

First Servlet

25

```
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.

1/19/01

First Servlet

26

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.
- This next example illustrates the use of the destroy() method.
- While alive, the servlet will say "I am alive!".
- When the server is stopped, the destroy() method is called, and the servlet records its time of death in a "rip.txt" text file.

1/19/01

First Servlet

27

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

```
public class Death extends HttpServlet {

    // Handle an HTTP GET Request
    public void doGet(HttpServletRequest request, HttpServletResponse
        response) throws IOException, ServletException
    {
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        out.println ("I am alive!");
        out.close();
    }
}
```

Continued....

1/19/01

First Servlet

28

```
// This method is called when one stops
// the Java Web Server
public void destroy() {
    try {
        FileWriter fileWriter = new FileWriter ("rip.txt");
        Date now = new Date();
        String rip = "I was destroyed at: "+now.toString();
        fileWriter.write (rip);
        fileWriter.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

1/19/01

First Servlet

29


Example rip.txt file

I was destroyed at: Thu Aug 24 11:10:58 CDT 2000

1/19/01


First Servlet

30



Form Data

1/19/01 First Servlet 31



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.

1/19/01 First Servlet 32

```

<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>

```

1/19/01 First Servlet 33

```

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....


1/19/01 First Servlet 34

```

    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>");
  }
}

```

1/19/01 First Servlet 35



Example 2: Reading all Parameters

1/19/01 First Servlet 36

Example 2

- Example 1 will only read explicit parameters.
- Now, let's look at a Servlet that echoes back all the form parameters you send it.
- You will probably remember this servlet from our discussions of HTML forms.

1/19/01

First Servlet

37

```
package coreservlets;
```

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

```
public class ShowParameters extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String title = "Reading All Request Parameters";
        out.println(ServletUtilities.headWithTitle(title) +
            "<BODY BGCOLOR=#FDF5E6>\n" +
            "<H1 ALIGN=CENTER>" + title + "</H1>\n" +
            "<TABLE BORDER=1 ALIGN=CENTER>\n" +
            "<TR BGCOLOR=#FFAD00>\n" +
            "<TH>Parameter Name<TH>Parameter Value(s)");
```

Output a simple HTML table for displaying the form parameters.

Continued....

```
Enumeration paramNames = request.getParameterNames();
while(paramNames.hasMoreElements()) {
    String paramName = (String)paramNames.nextElement();
    out.print("<TR><TD>" + paramName + "\n<TD>");
    String[] paramValues = request.getParameterValues(paramName);
    if (paramValues.length == 1) {
        String paramValue = paramValues[0];
        if (paramValue.length() == 0)
            out.println("<!--No Value-->");
        else
            out.println(paramValue);
    } else {
        out.println("<UL>");
        for(int i=0; i<paramValues.length; i++) {
            out.println("<LI>" + paramValues[i]);
        }
        out.println("</UL>");
    }
}
}
```

1. First, use `getParameterNames()` to retrieve an Enumeration of all form parameters.
2. Then, iterate through each element within the Enumeration.

Continued....

```
out.println("</TABLE>\n</BODY></HTML>");
}
```

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

doPost calls doGet(). Therefore the servlet will work just as well for HTTP POSTS or GETS.

1/19/01

First Servlet

40