

## Servlet Overview

MSIS 531 – Spring 2006

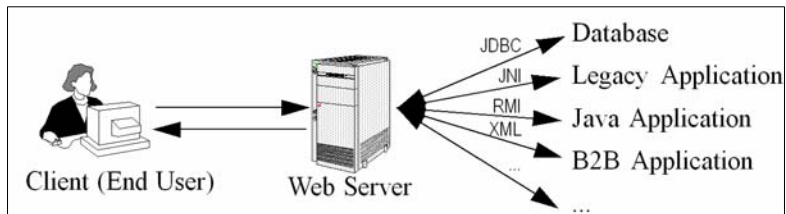
## Agenda

2

- The basic structure of servlets
- A simple servlet that generates plain text
- A servlet that generates HTML
- The servlet life cycle
- Servlets and packages
- Servlets and forms
- Processing request parameters

## A Servlet's Job

- Read explicit data sent by client (form data)
- Read implicit data sent by client (request headers)
- Generate the results
- Send the explicit data back to client (HTML)
- Send implicit data to client (status codes, response headers)

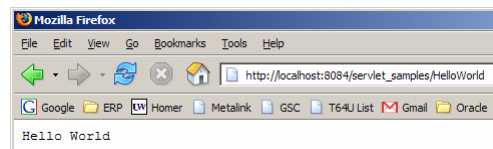


## Plain-Text Servlet (no HTML)

```
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");
    }
}
```

**Example: HelloWorld**



## A Servlet That Generates HTML

- Tell the browser that you're sending it HTML
  - `response.setContentType("text/html");`
- Modify the `println` statements to build a legal Web page
  - Print statements should output HTML tags
- Responsibility is on developer to write code that produces the correct result in the browser
  - Issue: if presentation modifications need to be made, servlet classes must be recompiled
- Consider the difference between the use of servlets (explicit Java classes) and JSPs (implicit Java classes)

## A Servlet That Generates HTML

```
public class HelloWWW extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String docType = "<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 " +
            "Transitional//EN">\n";
        out.println(docType +
            "<HTML>\n" +
            "<HEAD><TITLE>Hello WWW</TITLE></HEAD>\n" +
            "<BODY>\n" +
            "<H1>Hello WWW</H1>\n" +
            "</BODY></HTML>");
    }
}
```

Example: HelloWWW

## The Servlet Life Cycle

- **init**
  - Executed once when the servlet is first loaded  
*Not* called for each request
- **service**
  - Called in a new thread by server for each request  
Dispatches to doGet, doPost, etc.
- **doGet, doPost, doXxx**
  - Handles GET, POST, etc. requests
  - Override these to provide desired behavior
- **destroy**
  - Called when server deletes servlet instance  
*Not* called after each request

## The Role of Form Data

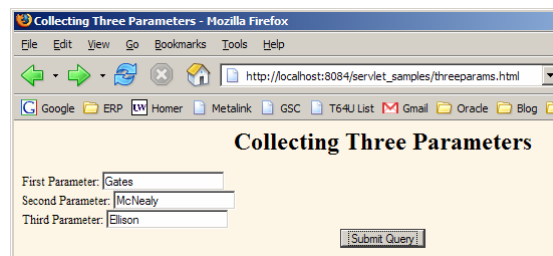
- Example URL at online travel agent
  - `http://host/path?user=Russ+Fish&origin=sea&dest=lax`
  - Names come from HTML author; values from end user
- Parsing form (query) data in traditional CGI:
  - Read the data one way (QUERY\_STRING) for GET requests, another way (standard input) for POST requests
  - Chop data at ampersands, then separate parameter names (left of =) from parameter values (right of =)
  - Decode URL values (e.g., "%7E" becomes "~")
- Greatly simplified in servlets
  - Use `request.getParameter` in all cases
  - Gives URL-decoded result

## Reading Form Data In Servlets

- `request.getParameter("name")`
  - Returns URL-decoded value of first occurrence of *name* in query string
  - Works identically for GET and POST requests
  - Returns null if parameter not found in query data
- `request.getParameterValues("name")`
  - Returns an array of the URL-decoded values of all occurrences of *name* in query string
  - Returns a one-element array if param not repeated
  - Returns null if parameter not found in query data
- `request.getParameterNames()`, `request.getParameterMap()`
  - Returns Enumeration or Map of request params
  - Example using Enumeration this shortly

## An HTML Form With Three Parameters

```
<FORM ACTION="/msis531-
  examples/servlet/servlet_examples.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>
```



## Reading the Three Parameters

```

public class ThreeParams extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        ...
        out.println(docType +
            "<HTML>\n" +
            "<HEAD><TITLE>"+title + "</TITLE></HEAD>\n" +
            "<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>");
    }
}

```

Example: threeparams.html, ThreeParams

## Reading All Request Parameters

- Previous example assumed specific parameter names
  - Here we get any and all passed from form
- A built-in Java class (Enumeration) is used to capture and process the request variables
  - request.getParameterNames returns the Enumeration
  - Convenient Enumeration method (hasMoreValues()) used to manage iterations through a while loop
- Test for null value is included—some form values may not have been entered by the users
- request.getParameterValues returns an array of values for each parameter name

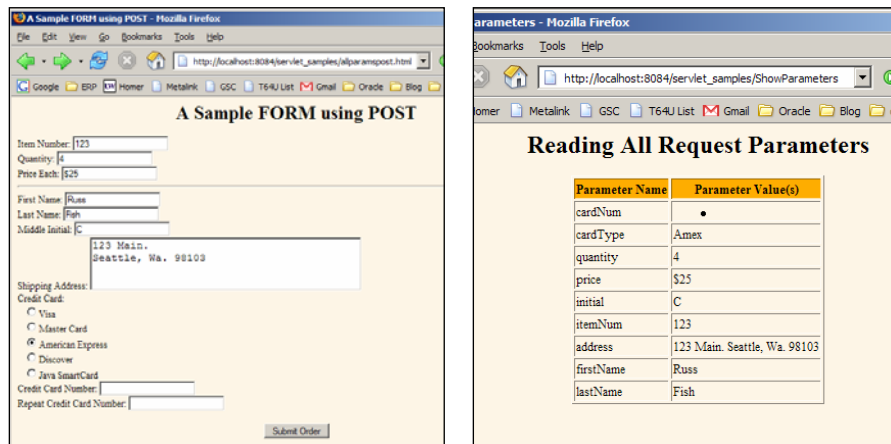
## Reading All Parameters (Cont.)

```

...
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("<I>No Value</I>");
        else
            out.println(paramValue);
    } else {
        out.println("<UL>");
        for(int i=0; i<paramValues.length; i++) {
            out.println("<LI>" + paramValues[i]);
        }
        out.println("</UL>");
    }
}
...

```

## Reading Parameters (form & results)



The left screenshot shows a web form titled "A Sample FORM using POST" with the following fields:

- Item Number: 123
- Quantity: 4
- Price Each: \$25
- First Name: Russ
- Last Name: Fish
- Middle Initial:
- Shipping Address: 123 Main, Seattle, Wa. 98103
- Credit Card:
  - Visa
  - Master Card
  - American Express
  - Discover
  - Java SmartCard
- Credit Card Number:
- Repeat Credit Card Number:
- Submit Order

The right screenshot shows the results page titled "Reading All Request Parameters" with the following table:

Parameter Name	Parameter Value(s)
cardNum	•
cardType	Amex
quantity	4
price	\$25
initial	C
itemNum	123
address	123 Main, Seattle, Wa. 98103
firstName	Russ
lastName	Fish

Example: allparamspost.html, ShowParameters