Module 3

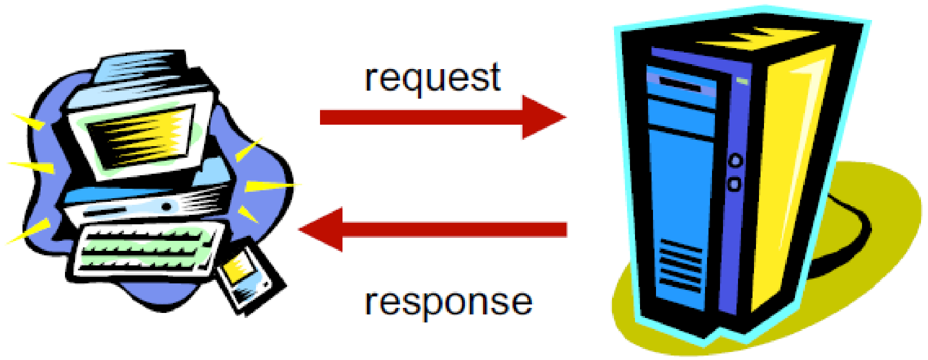
Servlet Overview

Course Objectives

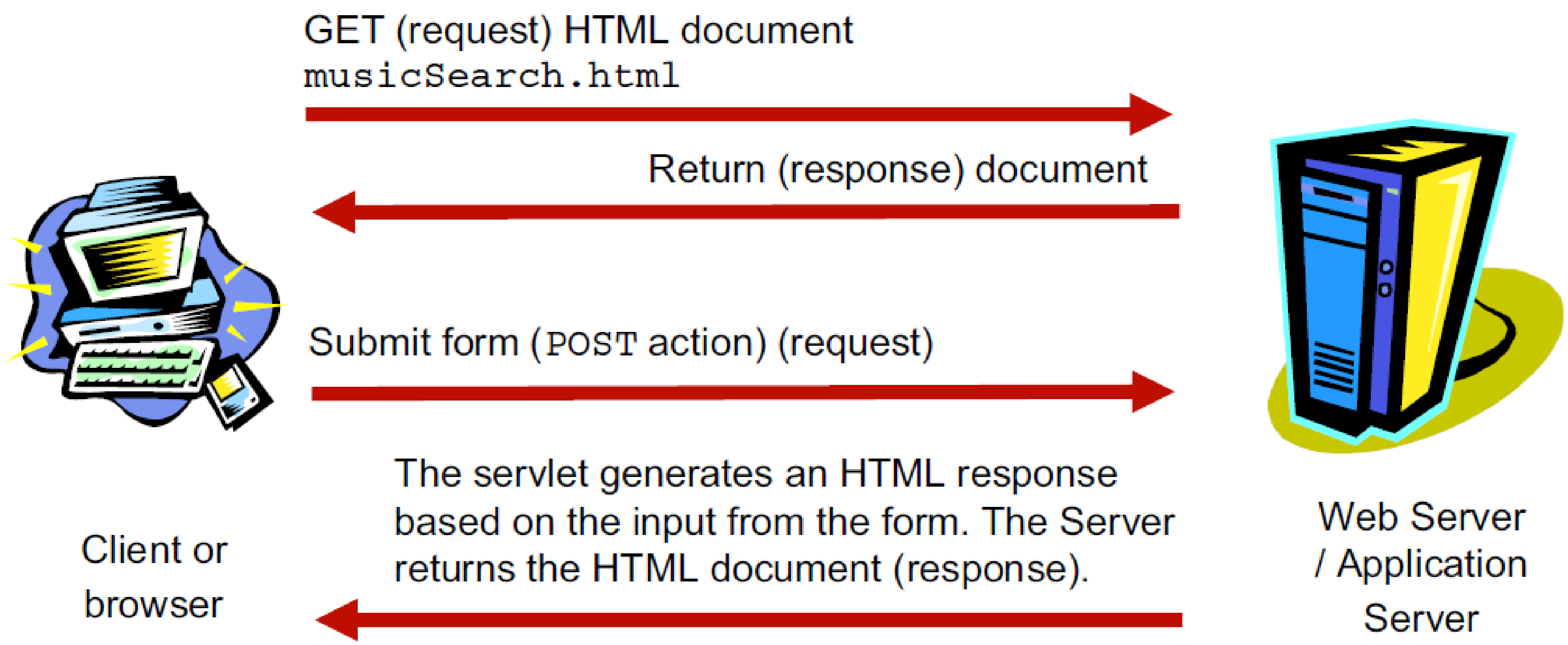
- After completing this module, you should be able to:
  - Describe servlets
  - Apply the servlet process flow
  - Develop a simple Java servlet example

What is a Servlet?

- A servlet is a standard, server-side component of a Java EE application that executes business logic on behalf of an HTTP request
  - It runs in the server tier (and not in the client)
  - It is a pure Java alternative to other technologies, such as CGI scripts
  - It is managed by the Web container

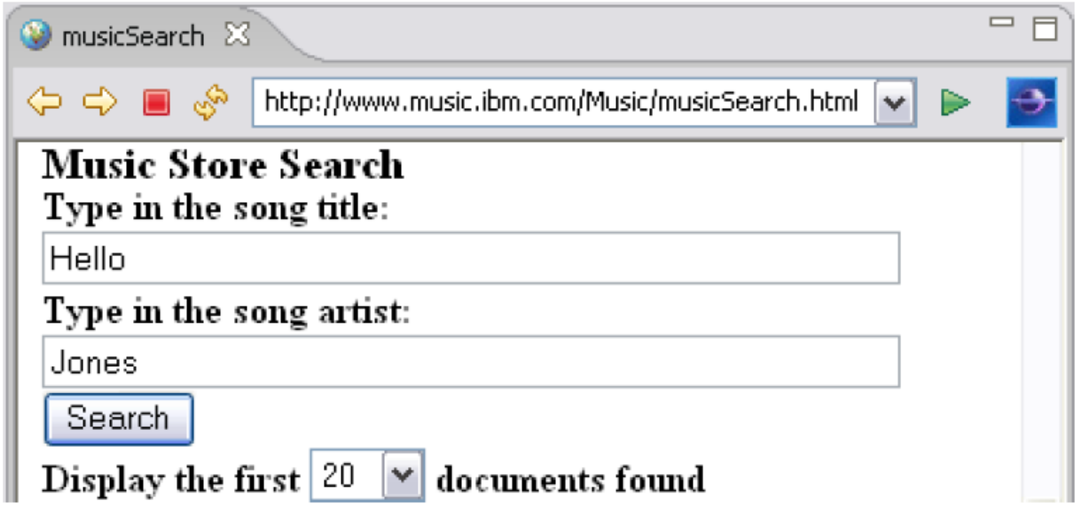


HTTP flows example: Forms / POST request


GET (request) HTML document
musicSearch.html
Return (response) document
Submit form (POST action) (request)
The servlet generates an HTML response based on the input from the form. The Server returns the HTML document (response).
Client or browser
Web Server / Application Server

HTTP flows example: HTML

```
...
<form action="/Music/SearchServlet" method="POST">
<h3>Music Store Search</h3><br><strong>Type in the song title:</strong>
<input type="text" size="55" name="song_title"><br>
<strong>Type in the song artist:</strong>
<input type="text" size="55" name="song_artist"><br>
<input type="submit" value="Search"><br><strong>Display the first </strong>
<select name="limit_number" value=20 size="1">
<option>250</option><option>100</option><option>50</option><option>40</option>
<option>30</option><option selected>20</option><option>10</option></select>
<strong>documents found</strong></form>...
```



Music Store Search
Type in the song title:
Hello
Type in the song artist:
Jones
Search
Display the first 20 documents found

5

HTTP protocol: sample request

- Conversation between a browser and a server
- Request phase
  - Request (POST in this example)
  - Header values
  - A blank line
  - Posted data (if the request method is POST)

```
POST /Music/SearchServlet HTTP/1.1
Accept: */*
Referer: http://www.music.ibm.com/Music/musicSearch.html
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; …)
Host: localhost:9080
Content-Length: 50
Connection: Keep-Alive
Cache-Control: no-cache

song_title=Hello&song_artist=Jones&limit_number=20
```

HTTP protocol: sample response

- Response phase
  - Status information (200 in this example)
  - Header values
  - A blank line
  - Output document (HTML)

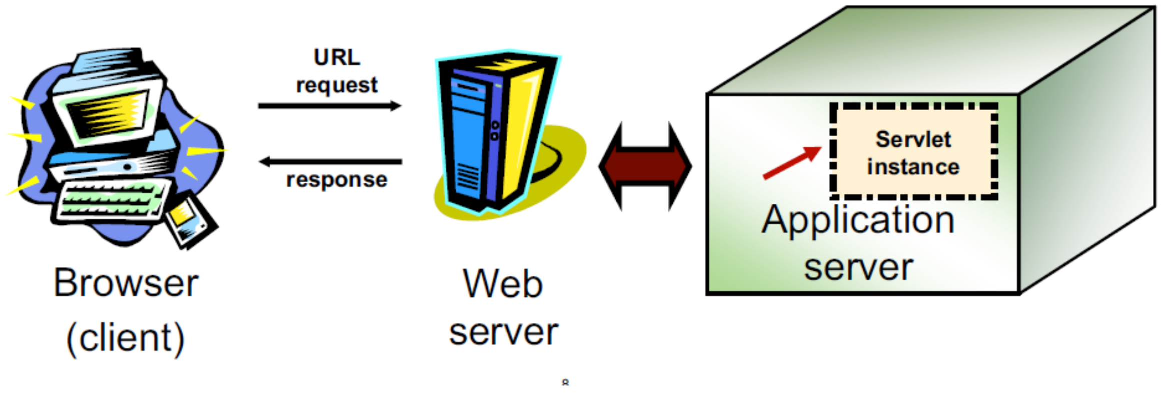
```
HTTP/1.1 200 OK
Content-Type: text/html; charset=ISO-8859-1
Content-Language: en-US
Set-Cookie: JSESSIONID=000093gEM_2OosI_mR6GBkZLJy9:-1; Path=/
Transfer-Encoding: chunked
Date: Mon, 12 Mar 2007 18:32:27 GMT
Server: WebSphere Application Server/6.1
Expires: Thu, 01 Dec 1994 16:00:00 GMT
Cache-Control: no-cache="set-cookie, set-cookie2"

<HTML>
<BODY>
<H1>Very simple dynamic document created on 01-Jun-2001</H1>
</BODY>
</HTML>
```

7

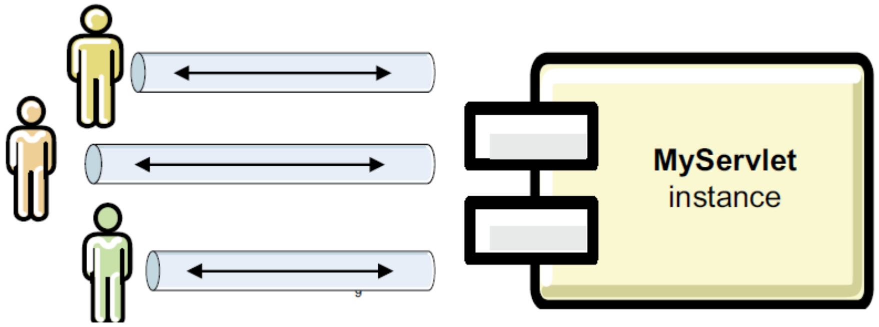
Servlet process flow

- The client makes a request naming a servlet as part of the URL.
- The Web server forwards the request to a Web container, which locates an instance of a servlet class.
- The Web container calls the servlet's service method.
- The servlet builds a response dynamically, and passes it to the Web server. External resources may also be used.
- The Web server sends the response back to the client.



Building a simple Java servlet

- To create a servlet that responds to HTTP requests, you must:
  - Create a class that extends javax.servlet.http.HttpServlet
  - Override the doGet or doPost methods to process HTTP GET or POST requests
    - Process HttpServletRequest input values
    - Invoke the business process
    - Set the HttpServletResponse values
    - Output HTML to the output PrintWriter
  - Servlets are inherently multithreaded



A simple Java servlet example

```java
package com.ibm.example.servlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;
import java.io.PrintWriter;
public class VerySimpleServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                         throws ServletException, IOException {
        String browser = request.getHeader("User-Agent");
        response.setStatus(HttpServletResponse.SC_OK);  // default
        response.setContentType("text/html");          // default
        PrintWriter out = response.getWriter();
        out.println("<HTML><HEAD><TITLE>Simple servlet");
        out.println("</TITLE></HEAD><BODY>");
        out.println ("Browser details: " + browser);
        out.println("</BODY></HTML>");
  }
}
```

Checkpoint

1. What class should your Java servlet extend?
2. What method needs to be overridden to process POST requests?
3. What method needs to be overridden to process GET requests?
4. What techniques could you use to process both GET and POST requests?

Module Summary

- This unit covered the following topics:
  - The basic steps needed to build a Java servlet for processing a request from a Web browser (client)
  - The subclassing requirements of a Java servlet
  - The methods that are overridden to handle HTTP GET or POST requests
  - It also presented a simple Java servlet