

Lecture 5

Generating the Server Responses: HTTP Status Codes

Lecture Agenda

Applied

- 1 ➤ Format of the HTTP response.
- 2 ➤ How to set the response status code.
- 3 ➤ What the status codes are good for.
- 4 ➤ A servlet that redirects users to browser-specific pages.
- 5 ➤ A front-end to various search engines.

Anatomy of HTTP Request/Response

A Typical HTTP GET Request Header

Review of HTTP GET Request Header

GET request with data passed (**color=dark**, **taste=malty**) on request header line.

HTTP GET Request

GET /select/selectBeerTaste.jsp?**color=dark&taste=malty** HTTP/1.1

Host: www.wickedlysmart.com

User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv:1.4) Gecko/20030624 Netscape/7.1

Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

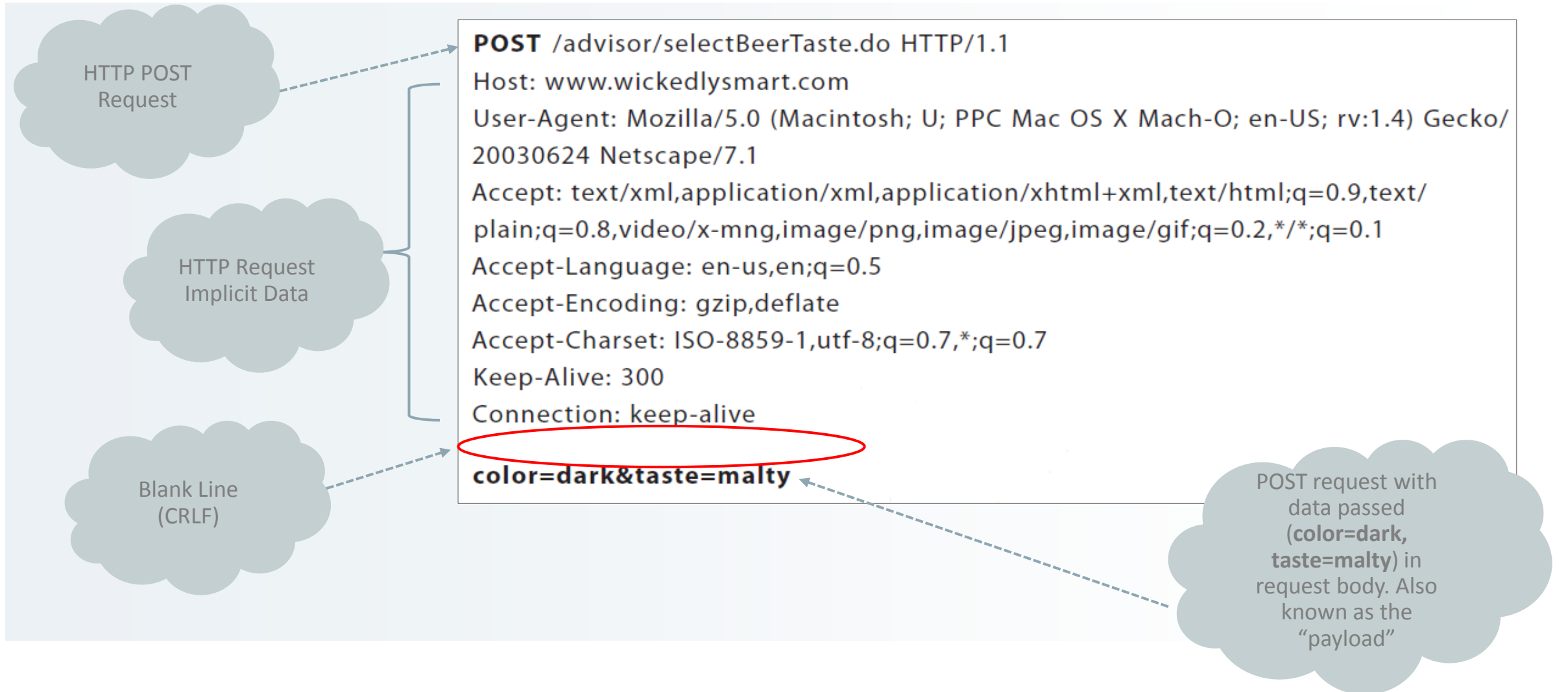
Connection: keep-alive

HTTP Request Implicit Data

Blank Line (CRLF)

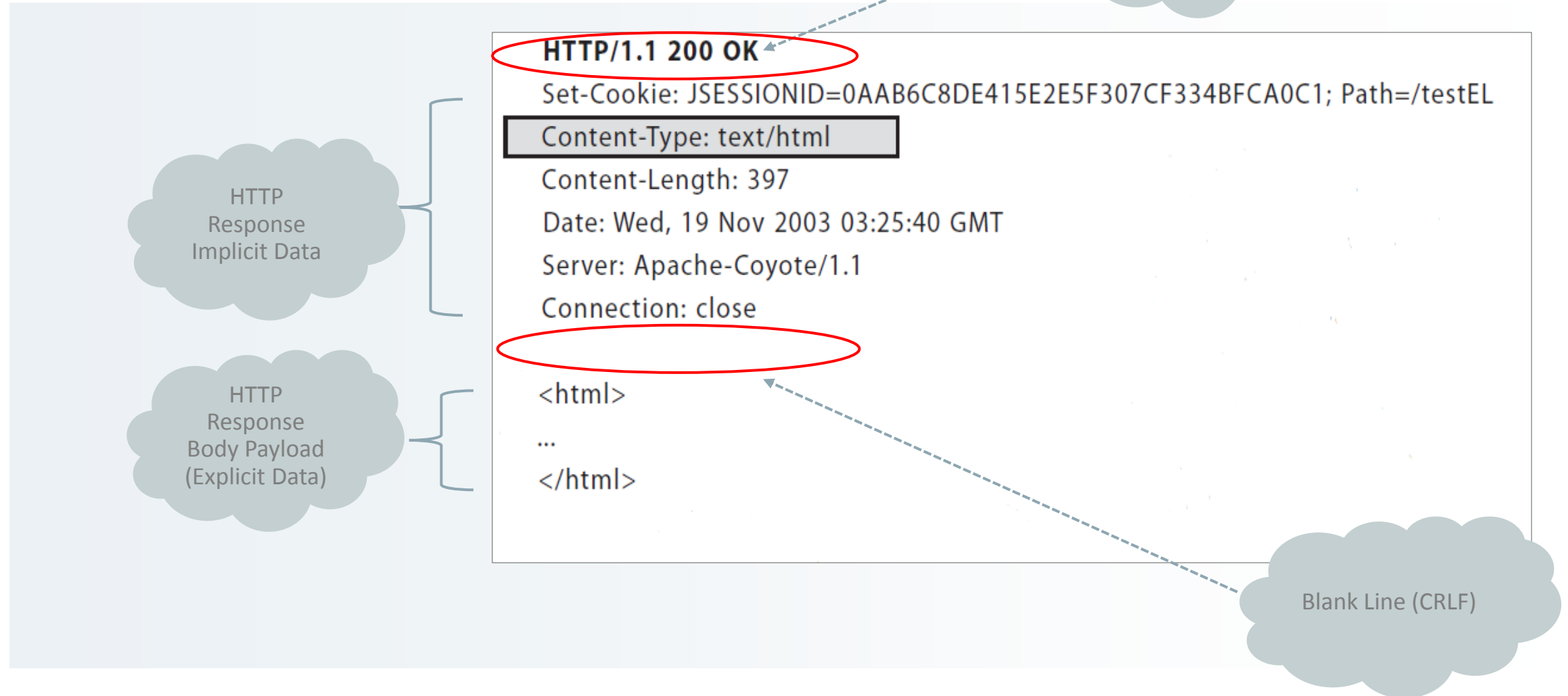
A Typical HTTP POST Request Header

Review of HTTP POST Request Header



A Typical HTTP Response

Review of HTTP Response



Status Codes

Setting Status Codes

- `response.setStatusCodes(int statusCode)`
 - Use a constant for the code, not an explicit int.
 - Constants are in `HttpServletResponse`
 - Names derived from standard message.
 - `SC_OK`
 - `SC_NOT_FOUND` etc ...
- `response.sendError(int code, String message)`
 - Wraps message inside a small HTML document.
- `response.sendRedirect(String url)`
 - Sets status code to 302
 - Sets Location response header also

Common HTTP Status Codes

Status Code Summary

Number	Type	Description
100-199	Informational	Request was received and is being processed.
200-299	Success	Request was successful.
300-399	Redirection	Further action must be taken to fulfill the request.
400-499	Client errors	Client has made a request that contains an error.
500-599	Server errors	Server has encountered an error.

Common HTTP Status Codes

Status Codes

Number	Name	Description
200	OK	Default status when the response is normal.
301	Moved Permanently	Requested resource has been permanently moved.
302	Found	Requested resource resides temporarily under a new URL.
400	Bad Request	Request could not be understood by the server due to bad syntax.
401	Unauthorized	Request requires authentication. Response must include a www-authenticate header.
403	Forbidden	Access to requested resource has been denied.

Common HTTP Status Codes

Status Codes Continued

Number	Name	Description
404	Not Found	Server could not find requested URL.
405	Method Not Allowed	Method specified in request line is not allowed for requested URL.
414	Request-URI Too Long	Typically caused by trying to pass too much data in a GET request. Usually resolved by converting the GET request to a POST request.
500	Internal Server Error	Server encountered an unexpected condition that prevented it from fulfilling the request.

Redirection

Redirection Analysis

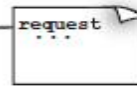
Sometimes you don't want to deal with the response yourself

Redirect

① Client types a URL into the browser bar...



② The request goes to the server/Container.



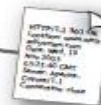
③ The servlet decides that the request should go to a completely different URL.



⑥ The browser gets the response, sees the "301" status code, and looks for a "Location" header.



⑤ The HTTP response has a status code "301" and a "Location" header with a URL as the value.



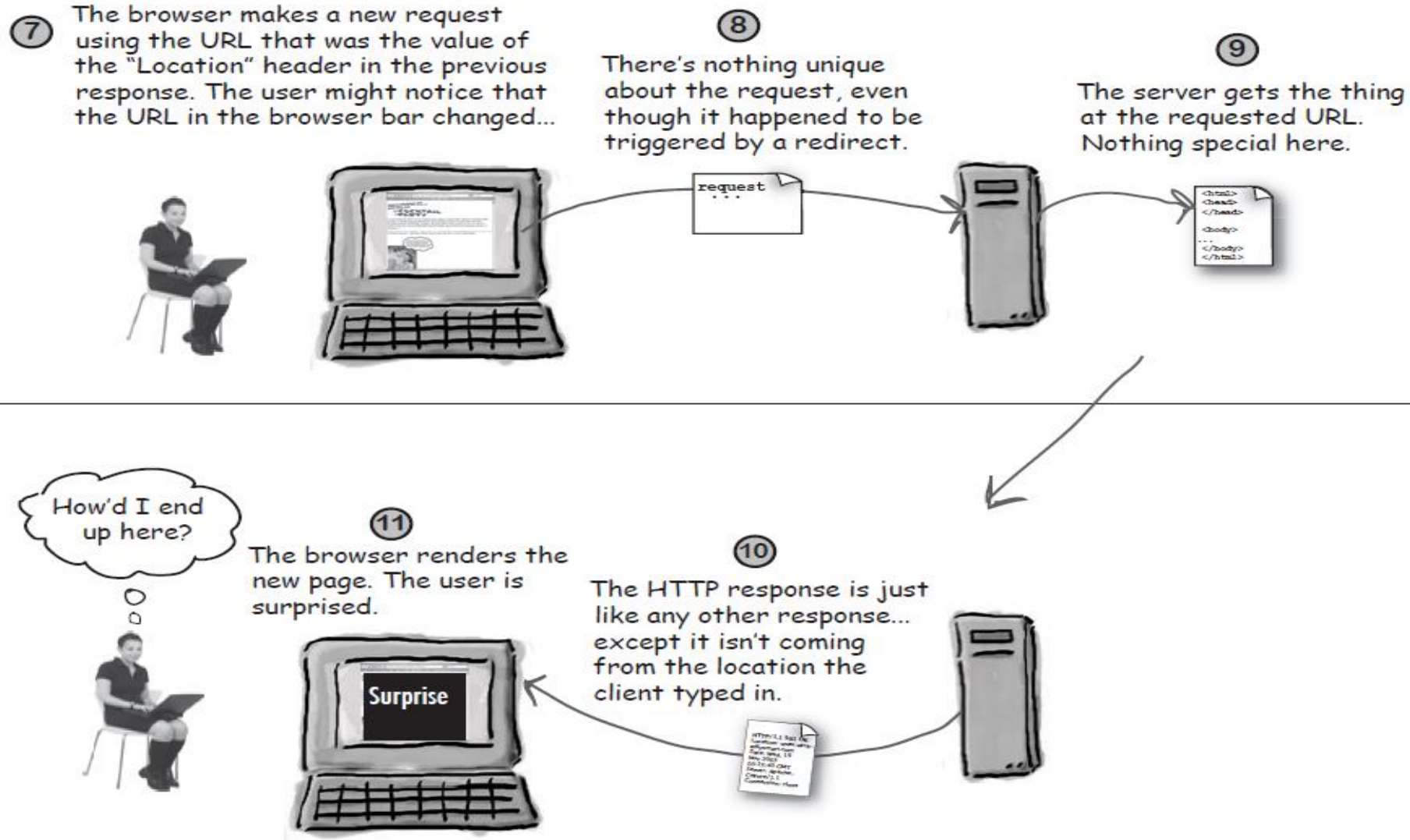
④ The servlet calls `sendRedirect(aString)` on the response and that's it.



response

Redirection Analysis

Sometimes you don't want to deal with the response yourself



Servlet Redirect

A Servlet that Redirects Users to Specific-Page

Redirection Example

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
    String userAgent = request.getHeader("User-Agent");  
    if( (userAgent!=null) && (userAgent.contains("Chrome")) ){  
  
        response.sendRedirect("http://www.chrome.com/");  
  
    } else {  
  
        response.sendRedirect("http://mozilla.com/");  
  
    }  
  
}
```

sendRedirect() Points Making the Browser do the Work

- Servlet Redirect Code

```
if (worksForMe) {  
    // handle the request  
} else {  
    response.sendRedirect("http://www.oreilly.com");  
}
```

- Servlet Redirect makes the browser do the work
 - A redirect lets the servlet off completely, After deciding that it can't do the work, the servlet simply calls the **sendRedirect()** method.

sendRedirect() Relative URLs

Using Relative URLs in sendRedirect()

- Using relative URLs as argument in sendRedirect()
 - A relative URL can be used instead of a full URL (“http://www ...”) as the argument.
 - Relative URLs come in two flavors
 1. Without starting forward slash ... `sendRedirect(“foo/stuff.html”)`
 2. With starting forward slash (/) ... `sendRedirect(“/foo/stuff.html”)`

sendRedirect() Relative URLs

Relative URL without forward slash

- Servlet Redirect without starting forward slash (/)
 - When the request comes into the servlet, the servlet calls sendRedirect() with a relative URL that does NOT start with forward slash.

Example: Imagine the client originally typed in:


http://wickedlysmart.com/myApp/cool/bar.do

Then servlet redirects with a relative URL (no starting forward slash) to:

sendRedirect("foo/stuff.html");

The container builds the full URL relative to the original request URL:

http://wickedlysmart.com/myApp/cool/foo/stuff.html



Relative
redirection
relative to original
request

sendRedirect() Relative URLs

Relative URL with forward slash

- Servlet Redirect without starting forward slash (/)
 - When the request comes into the servlet, the servlet calls sendRedirect() with a relative URL that does NOT start with forward slash.

Example: Imagine the client originally typed in:


http://wickedlysmart.com/myApp/cool/bar.do

Then servlet redirects with a relative URL to:

sendRedirect("/foo/stuff.html");

The container builds the full URL relative to web container itself:

http://wickedlysmart.com/foo/stuff.html



Relative redirection
relative to web
container

Use Case

Front-End to Various Search Engines

```
import java.io.*;

@WebServlet("/search-engines")
public class SearchEngines extends HttpServlet {

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String searchString = request.getParameter("searchString");
        if ((searchString == null) || (searchString.length() == 0)) {
            reportProblem(response, "Missing search string");
            return;
        }

        searchString = URLEncoder.encode(searchString, "utf-8");
        String searchEngineName = request.getParameter("searchEngine");
        if ((searchEngineName == null) || (searchEngineName.length() == 0)) {
            reportProblem(response, "Missing search engine name");
            return;
        }
        String searchURL = SearchUtilities.makeURL(searchEngineName, searchString);
        if (searchURL != null) {
            response.sendRedirect(searchURL);
        } else {
            reportProblem(response, "Unrecognized search engine");
        }
    }

    private void reportProblem(HttpServletResponse response, String message) throws IOException {
        response.sendError(HttpServletResponse.SC_NOT_FOUND, message);
    }
}
```

Translates string
into **application/x-
www-form-
urlencoded**
format

Builds URL given
search engine
name and search
string.

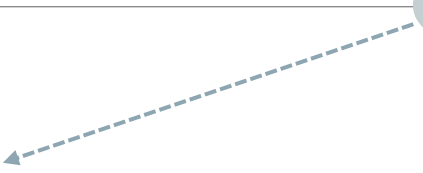
Report Error
SC_NOT_FOUND
(Status Code 404)

Use Case

Front-End to Various Search Engines Continued ...

```
public class SearchUtilities {  
  
    private static SearchSpec[] commonSpecs =  
    { new SearchSpec("Google", "http://www.google.com/search?q="),  
      new SearchSpec("Bing", "http://www.bing.com/search?q="),  
      new SearchSpec("Yahoo", "http://search.yahoo.com/bin/search?p="),  
      new SearchSpec("Ask", "http://www.ask.com/web?q="),  
      new SearchSpec("AllTheWeb", "http://www.alltheweb.com/search?q="),  
      new SearchSpec("AltaVista", "http://www.altavista.com/web/results?q="),  
      new SearchSpec("Lycos", "http://search.lycos.com/default.asp?query="),  
      new SearchSpec("HotBot", "http://hotbot.com/default.asp?query="),  
    };  
  
    public static SearchSpec[] getCommonSpecs() {  
        return (commonSpecs);  
    }  
  
    public static String makeURL(String searchEngineName, String searchString) {  
  
        SearchSpec[] searchSpecs = getCommonSpecs();  
        String searchURL = null;  
        for (SearchSpec spec : searchSpecs) {  
            if (spec.getName().equalsIgnoreCase(searchEngineName)) {  
                searchURL = spec.makeURL(searchString);  
                break;  
            }  
        }  
        return (searchURL);  
    }  
}
```

Common Search
Engine, Search
String
Specifications



Creates URL String



Use Case

Front-End to Various Search Engines Continued ...

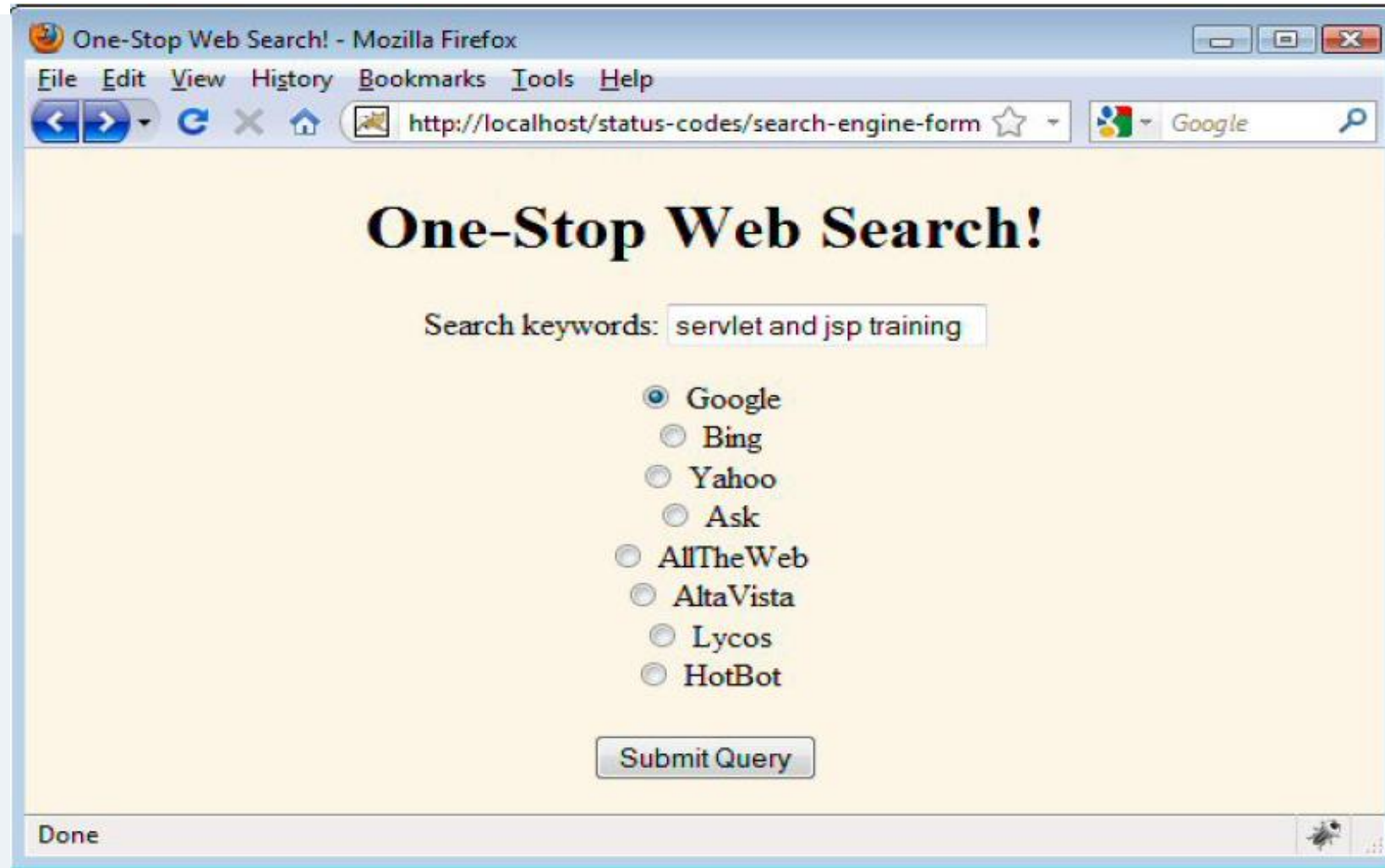
```
public class SearchSpec {  
    private String name, baseURL;  
  
    public SearchSpec(String name,  
                      String baseURL) {  
        this.name = name;  
        this.baseURL = baseURL;  
    }  
  
    public String makeURL(String searchString) {  
        return(baseURL + searchString);  
    }  
  
    public String getName() {  
        return(name);  
    }  
}
```



Make URL and
return

Use Case

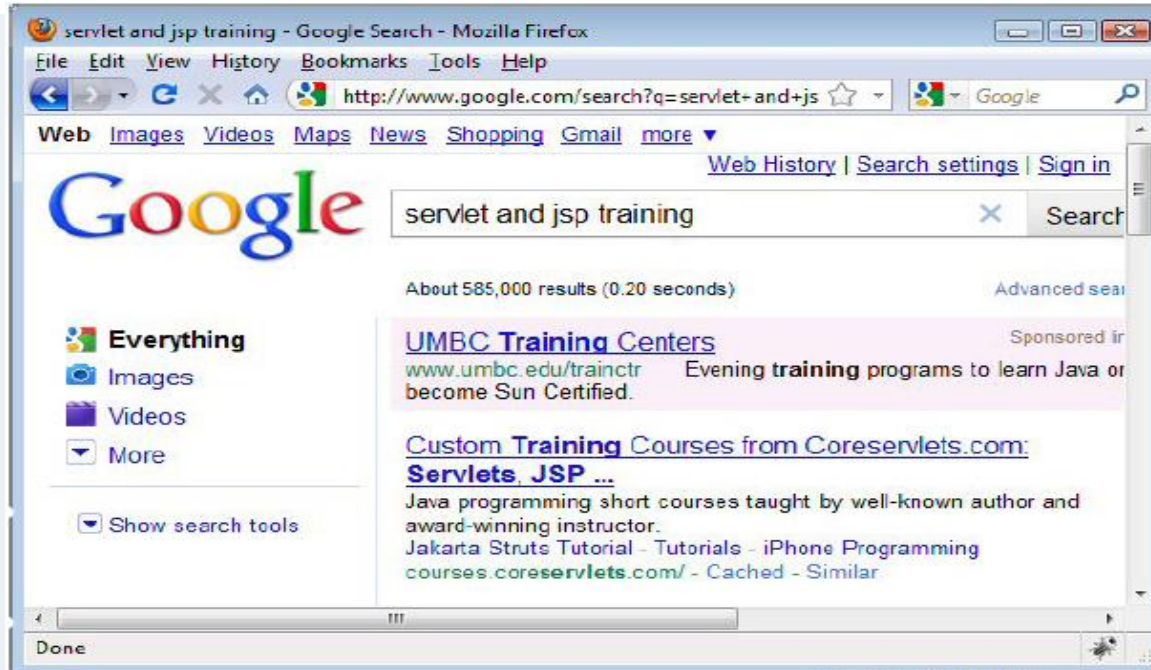
Front-End



Use Case

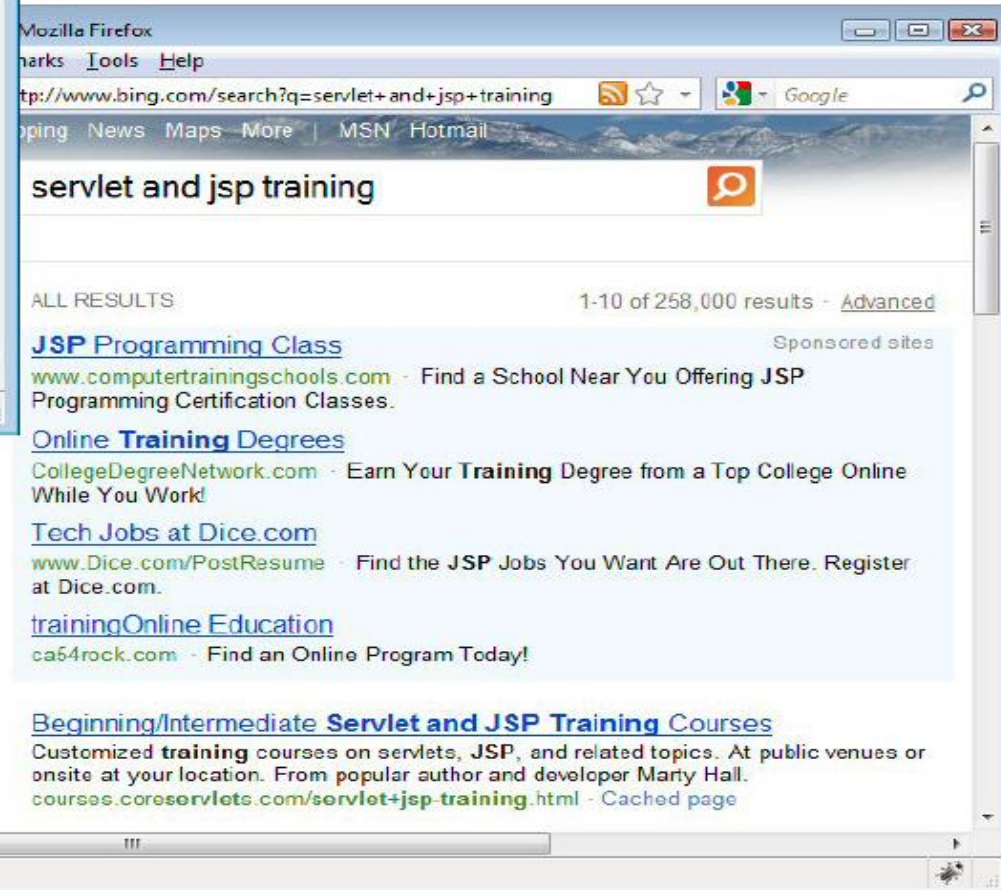
Result: Successful Search

Search Result
returned



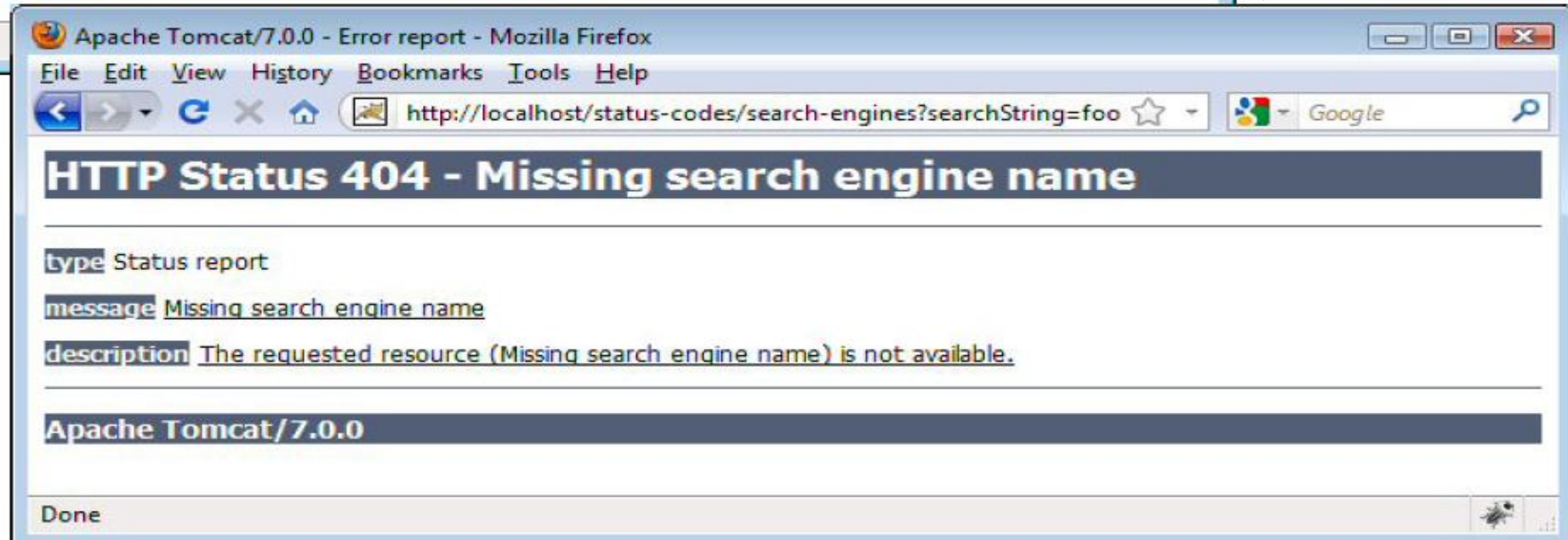
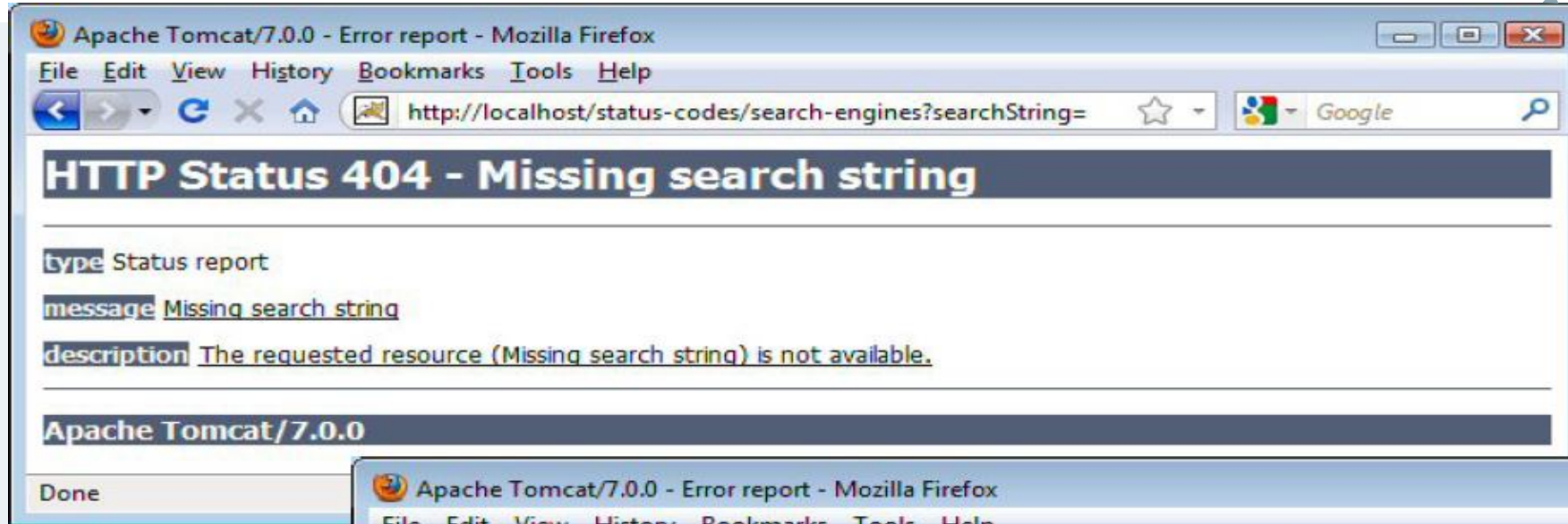
JSP and MySQL
JSP Database
JSP Code
Tomcat Web Hosting
Example of an
Overview.
Servlets and JSP
Books

SEARCH HISTORY
servlet and jsp training
jsp training
marty hall



Use Case

Result: Unsuccessful Search



Search failed. 404
Returned