Core Servlets and JavaServer Pages / 2e
Volume 1: Core Technologies
Marty Hall • Larry Brown

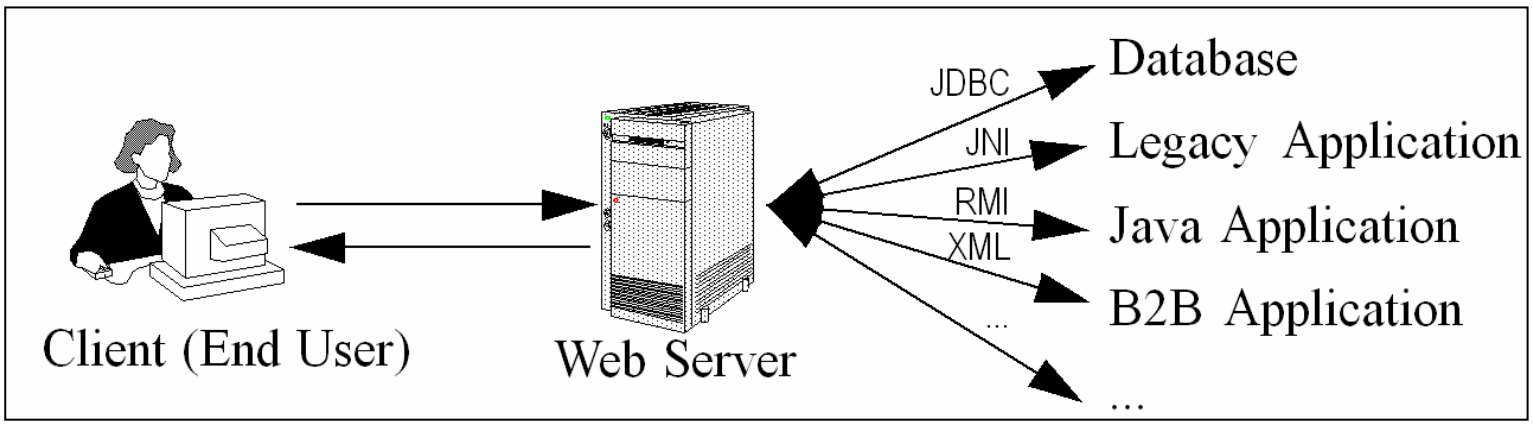
Servlet Basics

Agenda

- The basic structure of servlets
- A simple servlet that generates plain text
- A servlet that generates HTML
- Servlets and packages
- Some utilities that help build HTML
- The servlet life cycle
- Servlet debugging strategies

A Servlet's Job

- Read explicit data sent by client (form data)
- Read implicit data sent by client (request headers, cookies)
- Generate the results
- Send the explicit data back to client (HTML, XML, GIF images, etc.)
- Send the implicit data to client (status codes and response headers)



Basic servlet to handle GET requests

```
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 query data from HTML forms

// Use “response” to specify the HTTP response status
// code and headers (e.g., the content type and cookies).

    PrintWriter out = response.getWriter();

// Use “out” to send content to browser.
  }
}
```

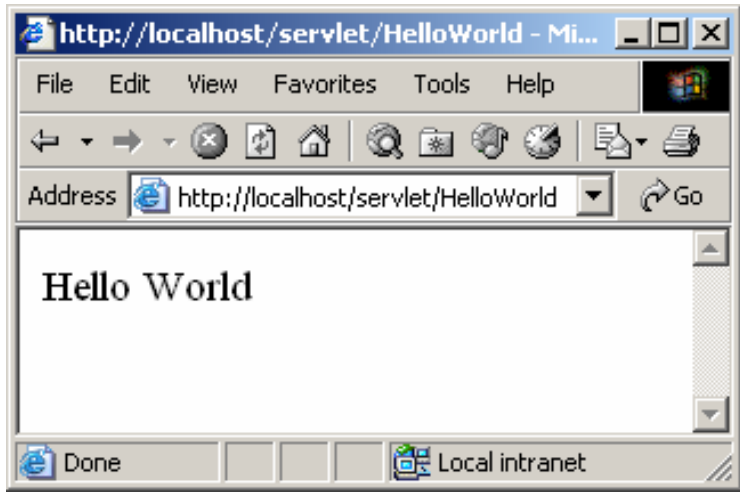
Basic servlet to handle GET requests

- **Servlets typically extend HttpServlet and override doGet or doPost**
- **HttpServletRequest has methods that enable you to find out information such as HTTP request headers, client's hostname and form/query data**
- **HttpServletResponse lets you specify outgoing information such as HTTP status codes and headers**
  - It also lets you obtain a PrintWriter that can be used to send document content back to the client

A Servlet That Generates Plain Text (HelloWorld.java)

```
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");
  }
}
```



A Servlet That Generates HTML

- **Tell the browser that you're sending it HTML**
  - `response.setContentType("text/html");`
  - **Other possibilites**
    - **"application/vnd.ms-excel" for Excel**
    - **"image/jpeg" for JPEG images**
    - **"text/xml" for XML documents**
- **The content type must be set *before* transmitting the actual document**

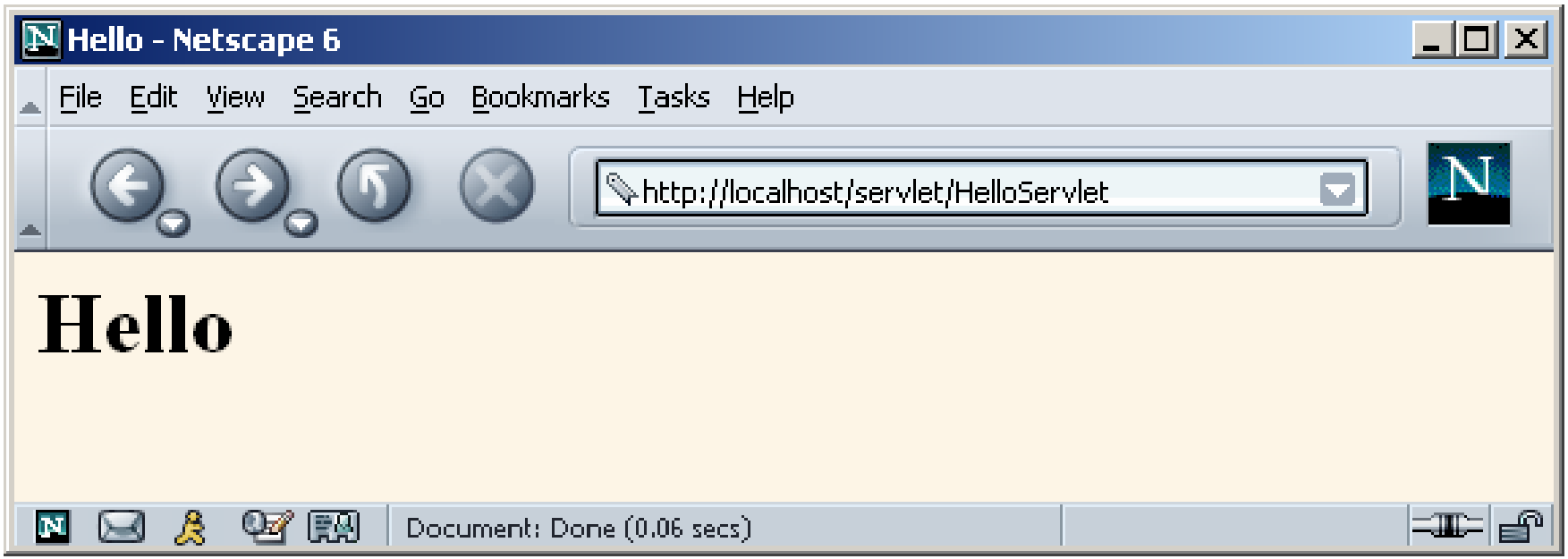
A Servlet That Generates HTML

- **Modify the println statements to build a legal Web page**
  - Print statements should output HTML tags, not plain text
- **Check your HTML with a formal syntax validator**
  - http://validator.w3.org/
  - http://www.htmlhelp.com/tools/validator/
- **Make sure to include the !DOCTYPE line as it is used by validators to detect the version of HTML you are using**

A Servlet That Generates HTML (Code)

```
public class HelloServlet 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</TITLE></HEAD>\n"+
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1>Hello</H1>\n" +
                "</BODY></HTML>");
  }
}
```

A Servlet That Generates HTML (Result)



Packaging Servlets

- **Move the files to a subdirectory that matches the intended package name**
  - For example, I'll use the coreservlets package for most of the rest of the servlets in this course. So, the class files need to go in a subdirectory called coreservlets.
- **Insert a package statement in the class file**
  - e.g., top of HelloServlet2.java:
    package coreservlets;
- **Keep CLASSPATH referring to top-level dir**
  - e.g., C:\Servlets+JSP. (No changes to CLASSPATH!)
- **Include package name in URL**
  - http://localhost/servlet/coreservlets.HelloServlet2

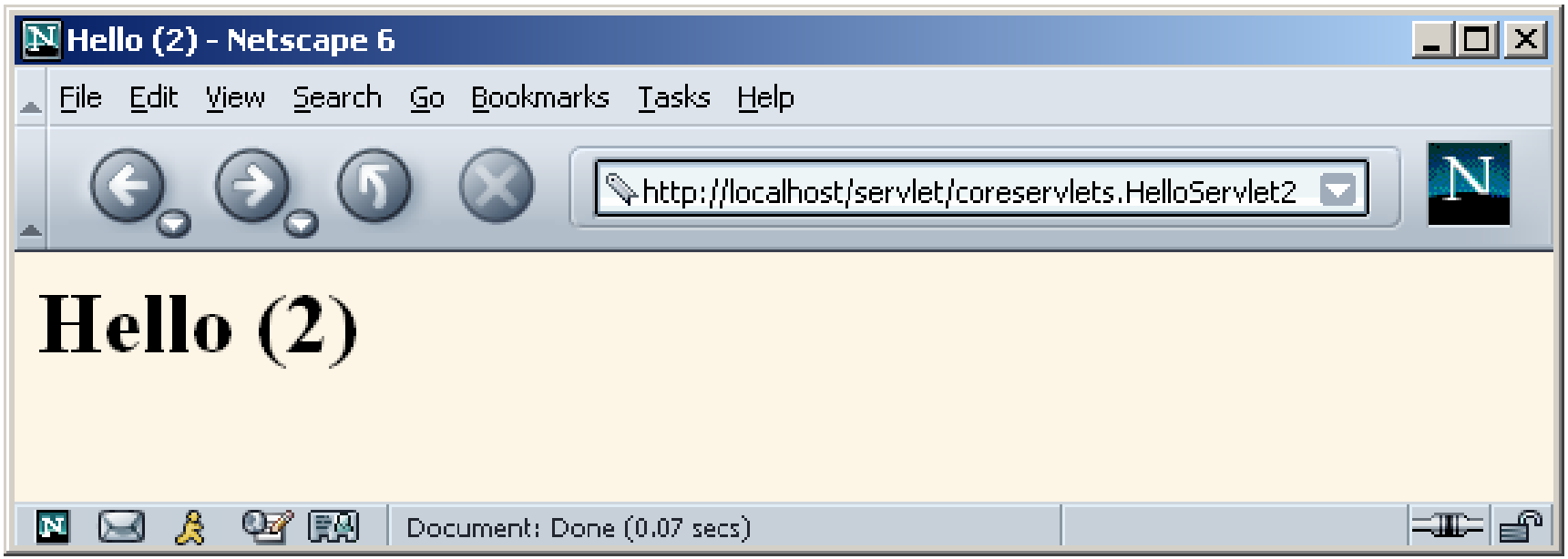
Packaging Servlets: HelloServlet2 (Code)

```
package coreservlets;

…

public class HelloServlet2 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 (2)</TITLE></HEAD>\n"+
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1>Hello (2)</H1>\n" +
                "</BODY></HTML>");
  }
}
```

Packaging Servlets: HelloServlet2 (Result)



Some Simple HTML-Building Utilities

```
public class ServletUtilities {
  public static final String DOCTYPE =
    "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML
  4.0 " +
    "Transitional//EN\">";

  public static String headWithTitle(String
  title) {
    return(DOCTYPE + "\n" +
           "<HTML>\n" +
           "<HEAD><TITLE>" + title +
           "</TITLE></HEAD>\n");
  }
  ...
}
```

- DOCTYPE and <HEAD> tag are unlikely to change, so this utility is useful

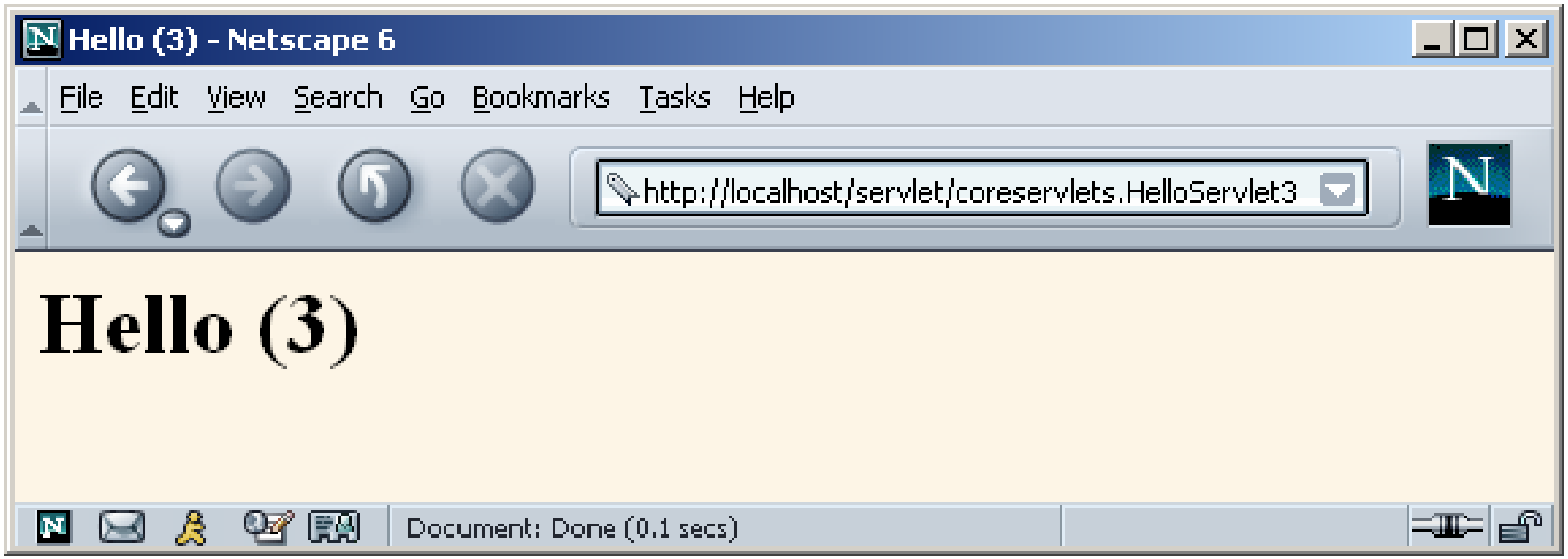
HelloServlet3: HelloServlet with Packages and Utilities

```
package coreservlets;

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

public class HelloServlet3 extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Hello (3)";
    out.println(ServletUtilities.headWithTitle(title)+
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1>" + title + "</H1>\n" +
                "</BODY></HTML>");
  }
}
```

HelloServlet3: Result



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. based on HTTP request type
  - Do not override this method!
- **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 service method

- **Each time the server receives a request for a servlet, the server spawns a new thread and calls service**
- **If a servlet handles a POST and GET in identical fashion, do not override service but implement doGet and doPost and have doPost call doGet(request, response)**
  - You can add support for other services later by adding doPut, doTrace, etc.
  - You can add support for modification dates by adding a getLastModified method which is invoked by the default service method
  - The service method gives you automatic support for:
    - HEAD requests
    - OPTIONS requests
    - TRACE requests

init method

- **Called when the servlet is first loaded, not repeated for each request**

```
public void init()throws ServletException{
   // initialization code…
}
```

- **General initializations**
  - Creates or loads data that will be used throughout the life of the servlet
  - Performs some one-time computation
  - In LotteryNumbers.java the init() method
    - Stores a page modification date that is used by the getLastModified method
    - Initializes an array with 10 random numbers

init method

LotteryNumbers servlet

```
public void init() throws ServletException {

 // Round to nearest second (i.e, 1000 milliseconds)
 modTime = System.currentTimeMillis()/1000*1000;

// builds array of 10 random numbers
 for(int i=0; i<numbers.length; i++) {
   numbers[i] = randomNum();
  }
}
```

init method

- **Since, the output of the servlet does not change except when the server is rebooted, init stores a page modification date that is used by the getLastModified method**
  - This method should return a modification time expressed in milliseconds and is converted into GMT for comparison with the Last-Modified header
  - If the server receives a conditional GET request (specifying that the client only wants pages marked If-Modified-Since a particular date), the system compares the specified date to that returned by getLastModified, returning the page only if it has been changed after the specified date
  - Browsers make these conditional requests for pages stored in their caches to get faster results and reduce server load
  - Run WebClient program

init method

- **Run WebClient program with Request Line of**

```
GET /servlet/coreservlets.LotteryNumbers HTTP/1.0
```

- **and then with same request line but with a Request Header of**

```
If-Modified-Since: Fri, 13 Sep 2020 17:00:00 GMT
```

destroy method

- **The server may decide to remove a previously loaded servlet instance, but before doing so it calls the destroy method**
  - Close database connections
  - Halt background threads
  - Write cookie lists to disk, etc.
- **If system crashes, destroy is not invoked**

Debugging Servlets

- **Use print statements; run server on desktop**
- **Use Apache Log4J**
- **Integrated debugger in IDE**
- **Look at the HTML source**
- **Return error pages to the client**
  - Plan ahead for missing or malformed data
- **HttpServlet has a log method for writing to a log file on the server**
  - log("message") or log("message", Throwable)
- **Separate the request and response data .**
  - Request: see EchoServer at www.coreservlets.com
  - Response: see WebClient at www.coreservlets.com
- **Stop and restart the server**

Summary

- **Main servlet code goes in doGet or doPost:**
  - The HttpServletRequest contains the incoming information
  - The HttpServletResponse lets you set outgoing information
    - Call setContentType to specify MIME type
    - Call getWriter to obtain a Writer pointing to client
- **One-time setup code goes in init**
  - Servlet gets initialized and loaded once
  - Servlet gets invoked multiple times
  - Initialization parameters set in web.xml (covered in detail in *More Servlets & JavaServer Pages* Chapter 5)