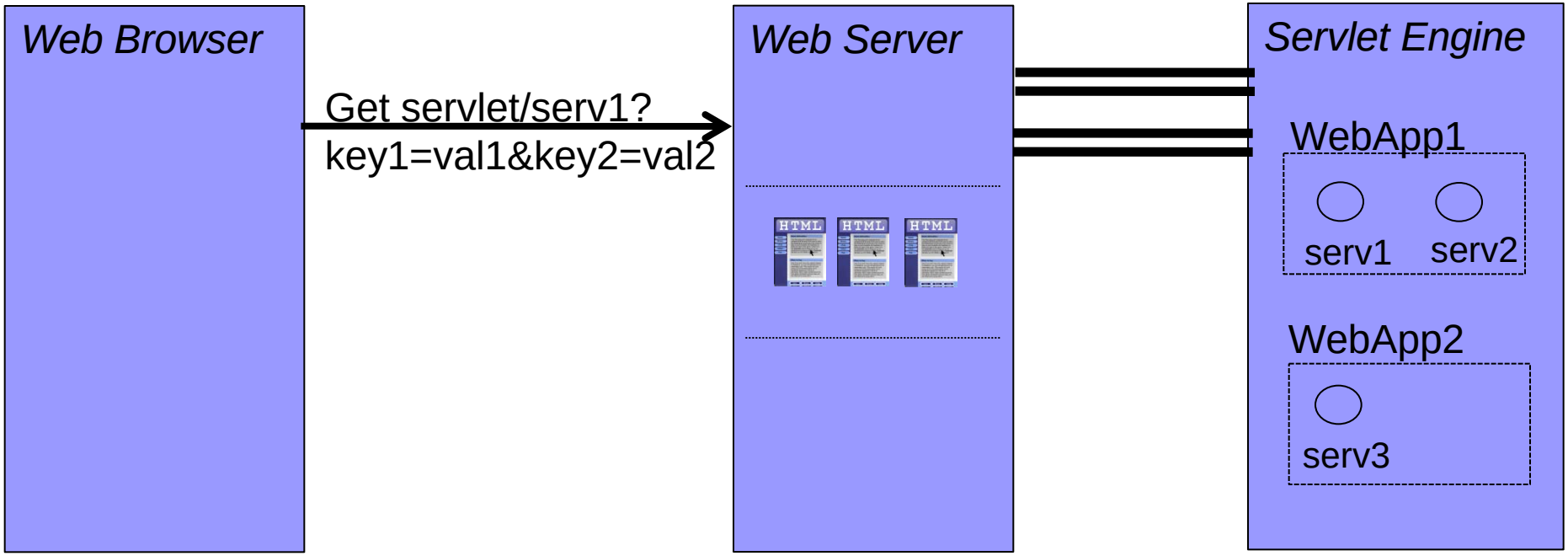
CSC309: Introduction to Web Programming

Lecture 10

Wael Aboulsaadat

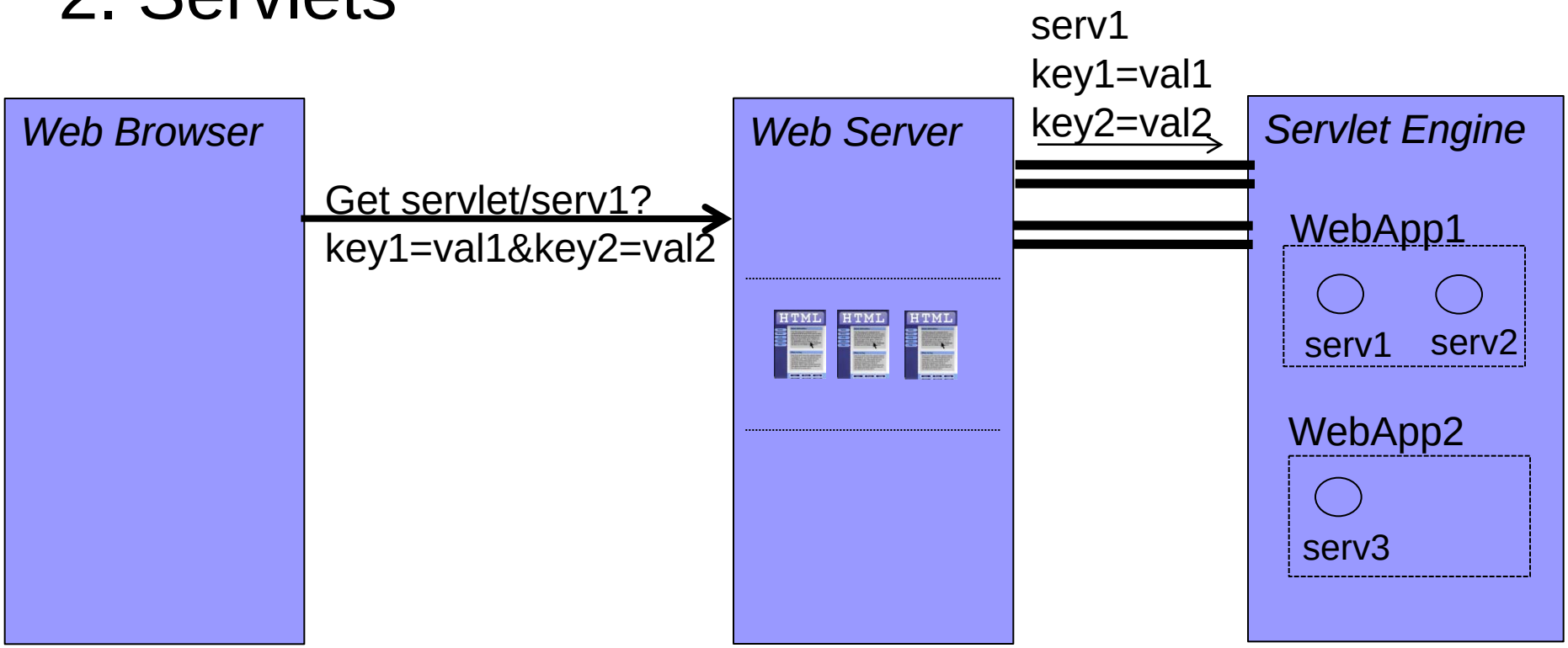
WebServer - WebApp Communication

2. Servlets



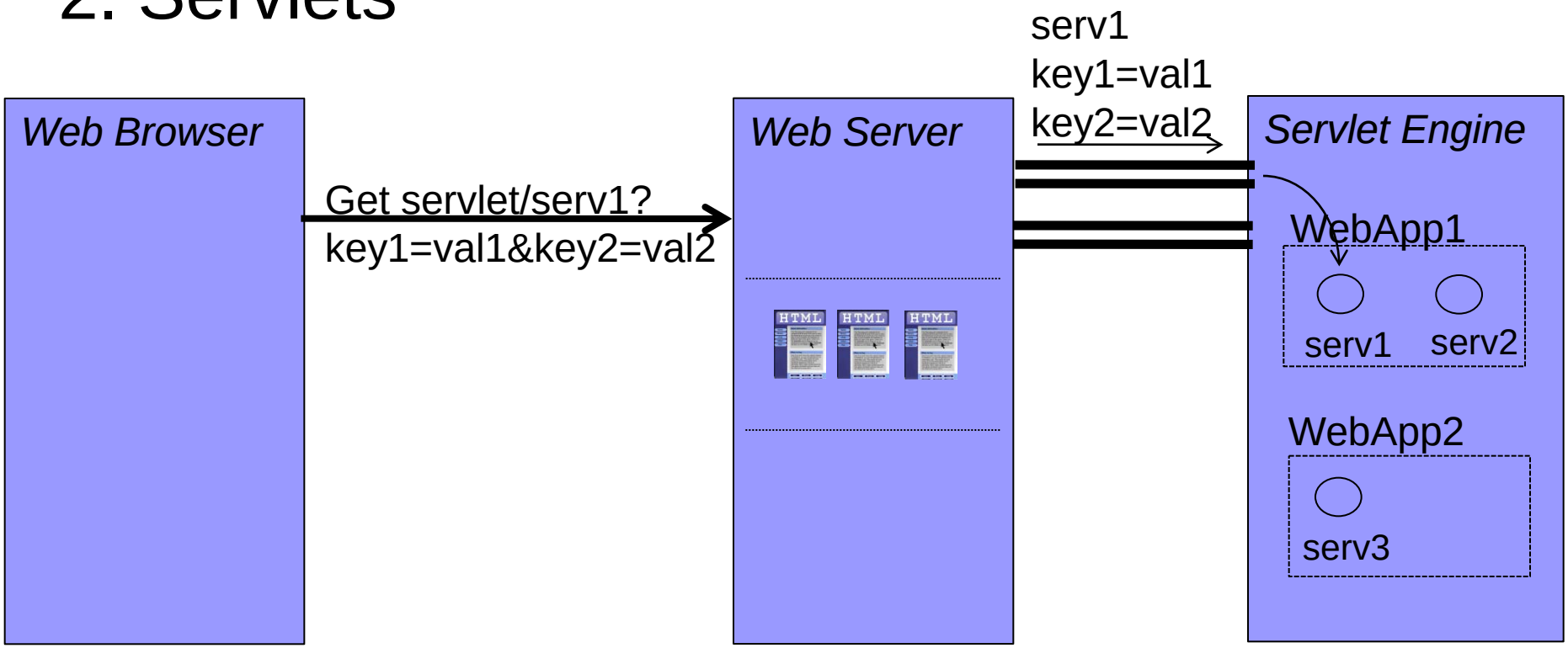
WebServer - WebApp Communication

2. Servlets



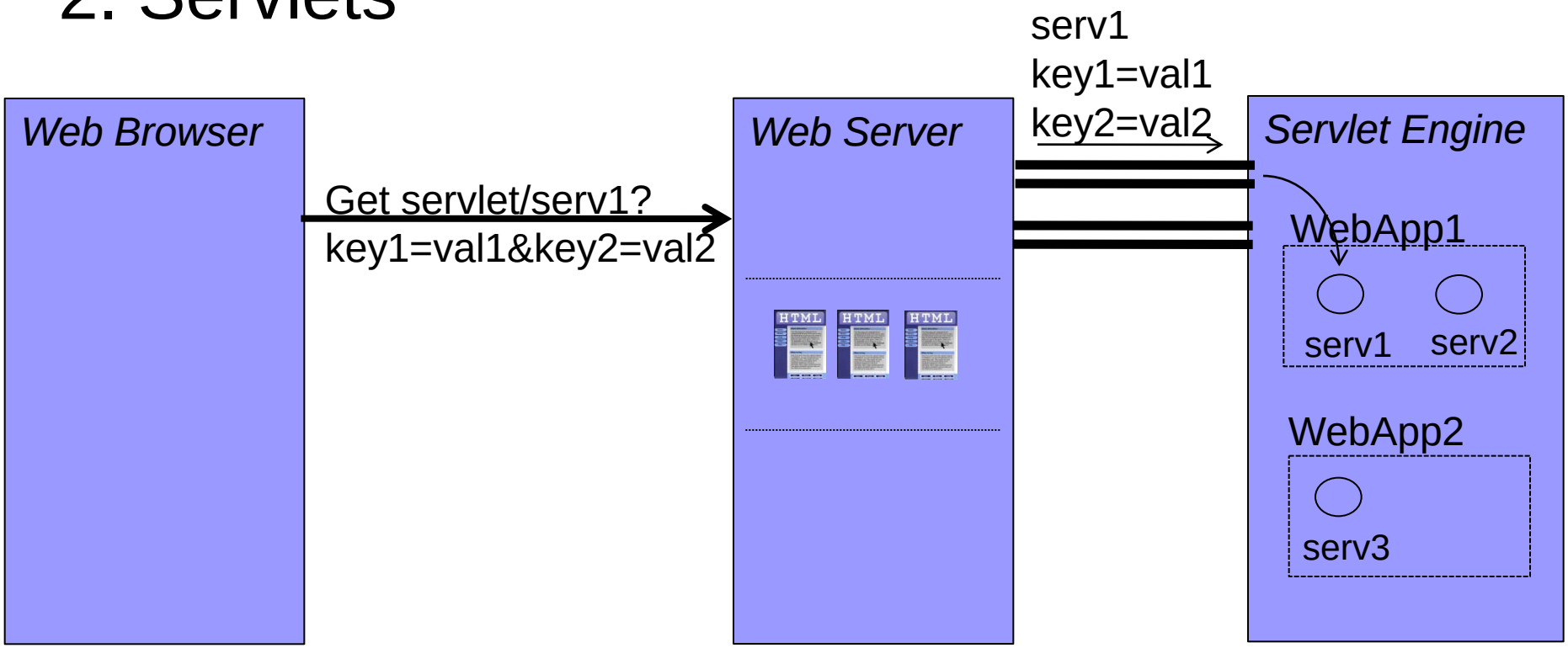
WebServer - WebApp Communication

2. Servlets



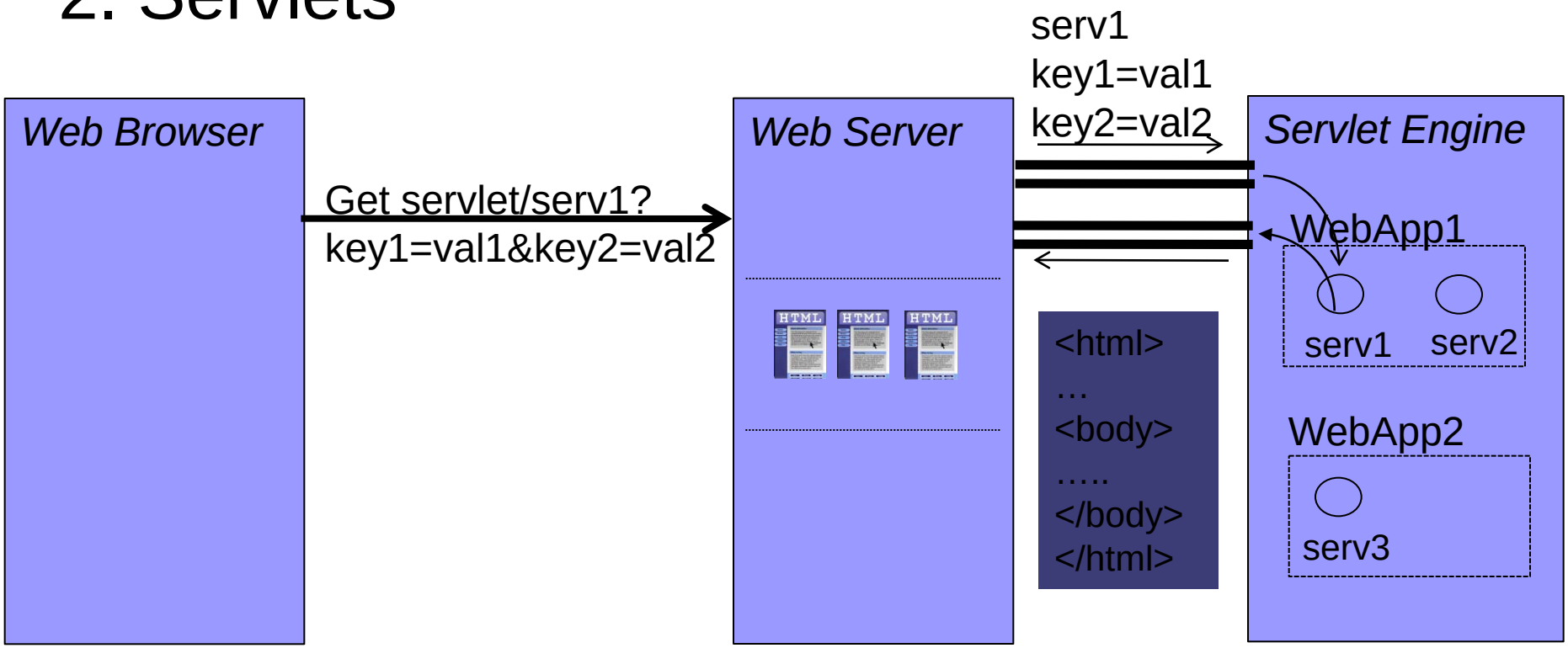
WebServer - WebApp Communication

2. Servlets



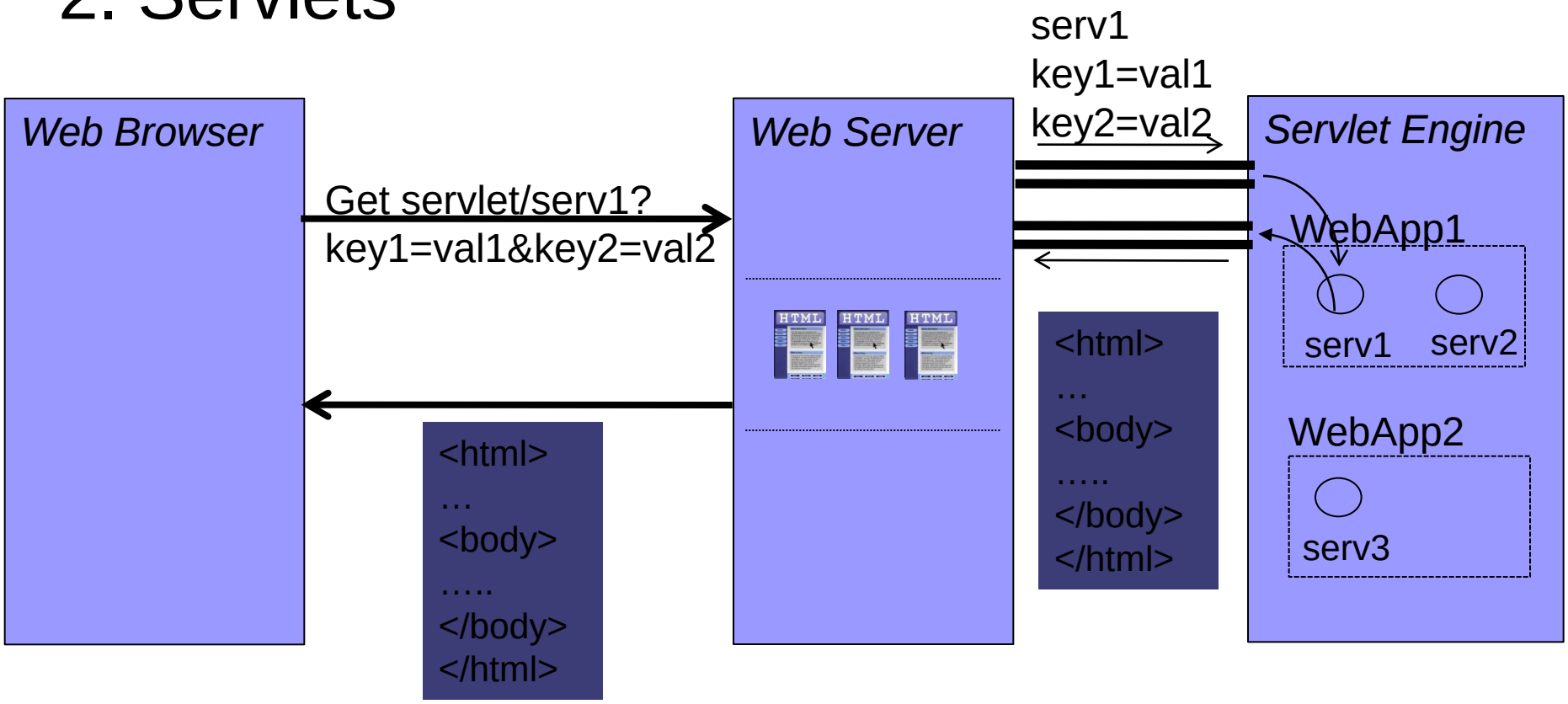
WebServer - WebApp Communication

2. Servlets



WebServer - WebApp Communication

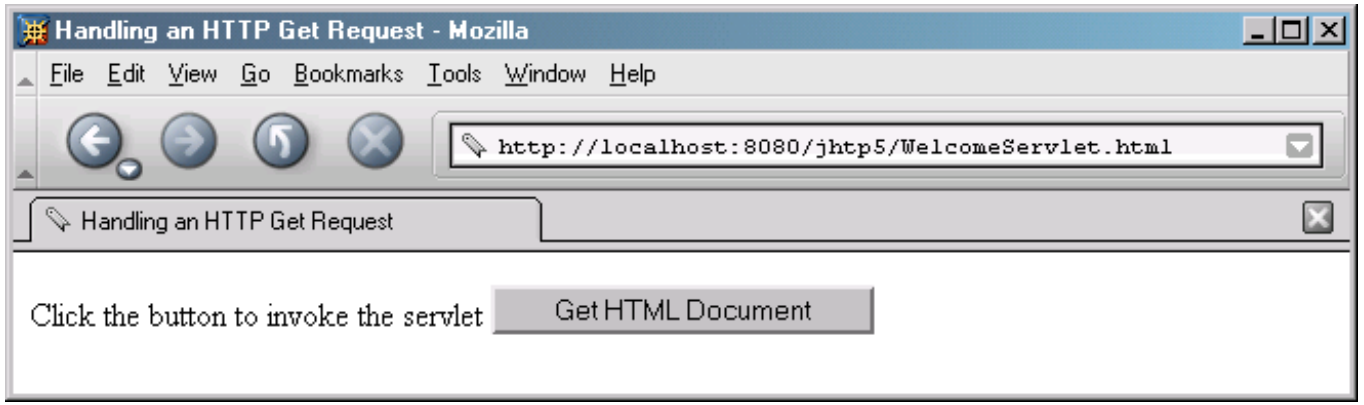
2. Servlets



```
4  import javax.servlet.*;
5  import javax.servlet.http.*;
6  import java.io.*;
7
8  public class WelcomeServlet extends HttpServlet {
9
10    // process "get" requests from clients
11    protected void doGet( HttpServletRequest request,
12       HttpServletResponse response )
13          throws ServletException, IOException
14    {
15       response.setContentType( "text/html" );
16       PrintWriter out = response.getWriter();
17
18       // send XHTML page to client
19
20       // start XHTML document
21       out.println( "<?xml version = \"1.0\"?>" );
22
23       out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
24          "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
25          "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
26
```

```
27          out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
28
29          // head section of document
30          out.println( "<head>" );
31          out.println( "<title>A Simple Servlet Example</title>" );
32          out.println( "</head>" );
33
34          // body section of document
35          out.println( "<body>" );
36          out.println( "<h1>Welcome to Servlets!</h1>" );
37          out.println( "</body>" );
38
39          // end XHTML document
40          out.println( "</html>" );
41          out.close();  // close stream to complete the page
42       }
43    }
```

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 24.6: WelcomeServlet.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Handling an HTTP Get Request</title>
10 </head>
11
12 <body>
13    <form action = "/jhtp5/welcome1" method = "get">
14
15       <p><label>Click the button to invoke the servlet
16          <input type = "submit" value = "Get HTML Document" />
17       </label></p>
18
19    </form>
20 </body>
21 </html>
```


Handling an HTTP Get Request - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/WelcomeServlet.html
Handling an HTTP Get Request
Click the button to invoke the servlet
Get HTML Document


A Simple Servlet Example - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/welcome1?
A Simple Servlet Example
Welcome to Servlets!

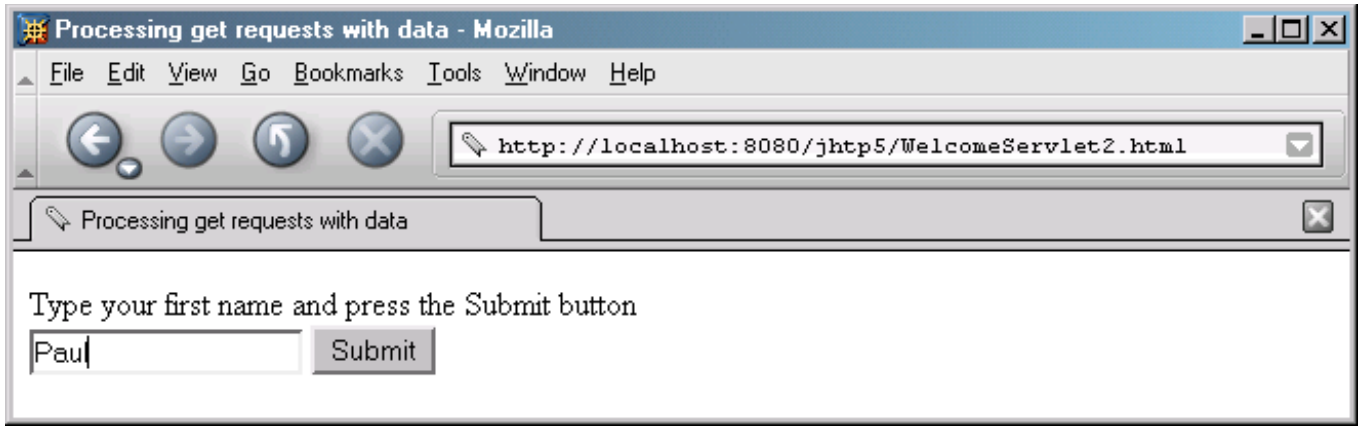
```
1   <!DOCTYPE web-app PUBLIC \
2      "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
3      "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
4
5   <web-app>
6
7      <!-- General description of your Web application -->
8      <display-name>
9         Java How to Program JSP
10        and Servlet Chapter Examples
11     </display-name>
12
13     <description>
14        This is the Web application in which we
15        demonstrate our JSP and Servlet examples.
16     </description>
17
18     <!-- Servlet definitions -->
19     <servlet>
20        <servlet-name>welcome1</servlet-name>
21
22        <description>
23           A simple servlet that handles an HTTP get request.
24        </description>
25
```

```
26          <servlet-class>
27             WelcomeServlet
28          </servlet-class>
29       </servlet>
30
31       <!-- Servlet mappings -->
32       <servlet-mapping>
33          <servlet-name>welcome1</servlet-name>
34          <url-pattern>/welcome1</url-pattern>
35       </servlet-mapping>
36
37    </web-app>
```

```
2  // Processing HTTP get requests containing data.
3
4  import javax.servlet.*;
5  import javax.servlet.http.*;
6  import java.io.*;
7
8  public class WelcomeServlet2 extends HttpServlet {
9
10    // process "get" request from client
11    protected void doGet( HttpServletRequest request,
12       HttpServletResponse response )
13          throws ServletException, IOException
14    {
15       String firstName = request.getParameter( "firstname" );
16
17       response.setContentType( "text/html" );
18       PrintWriter out = response.getWriter();
19
20       // send XHTML document to client
21
22       // start XHTML document
23       out.println( "<?xml version = \"1.0\"?>" );
24
```

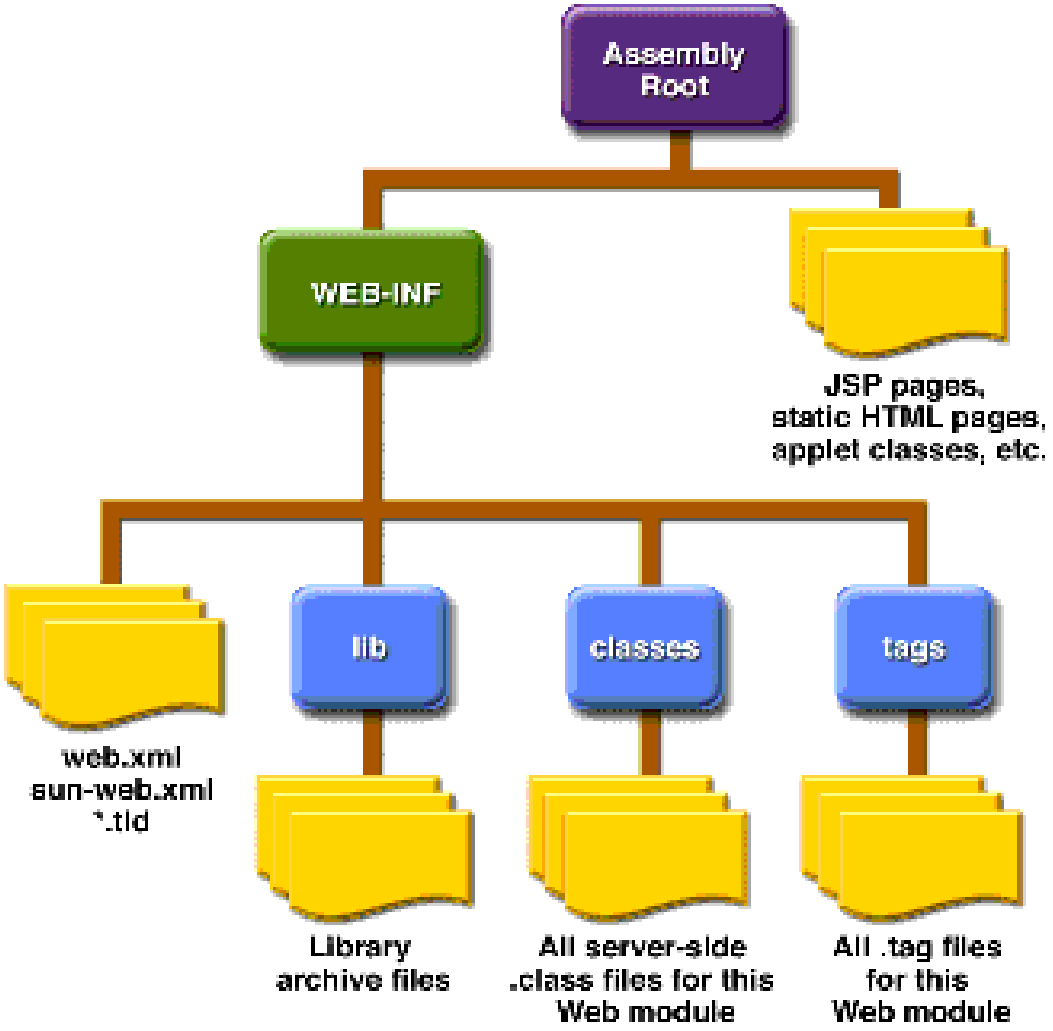
```
25          out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
26             "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
27             "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
28
29          out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
30
31          // head section of document
32          out.println( "<head>" );
33          out.println(
34             "<title>Processing get requests with data</title>" );
35          out.println( "</head>" );
36
37          // body section of document
38          out.println( "<body>" );
39          out.println( "<h1>Hello " + firstName + ",<br />" );
40          out.println( "Welcome to Servlets!</h1>" );
41          out.println( "</body>" );
42
43          // end XHTML document
44          out.println( "</html>" );
45          out.close();  // close stream to complete the page
46       }
47    }
```

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 24.12: WelcomeServlet2.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Processing get requests with data</title>
10 </head>
11
12 <body>
13    <form action = "/jhtp5/welcome2" method = "get">
14
15       <p><label>
16          Type your first name and press the Submit button
17          <br /><input type = "text" name = "firstname" />
18          <input type = "submit" value = "Submit" />
19       </p></label>
20
21    </form>
22 </body>
23 </html>
```


Processing get requests with data - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/WelcomeServlet2.html
Processing get requests with data
Type your first name and press the Submit button
Paul
Submit


Processing get requests with data - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/welcome2?firstname=Paul
Processing get requests with data
Hello Paul,
Welcome to Servlets!

Servlets – WebApp structure



Servlet Lifecycle

Servlets Demo

Servlet Demo – cont'd

1. Forwarding via Servlets

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

public class ForwardServlet extends HttpServlet {
   public void doGet(HttpServletRequest request,
   HttpServletResponse response) throws IOException,
   ServletException {
      HttpSession mySession = request.getSession();
      Date myDate = (Date)mySession.getAttribute("firstvisit");
      if (myDate == null) {
         myDate = new Date();
         mySession.setAttribute("firstvisit", myDate);
      }
      RequestDispatcher rd = request.getRequestDispatcher("/Welcome.html");
      rd.forward(request, response);
   }
}
```

Servlet Demo – cont'd

1. Sharing Data between Servlets

```
1:  public class FooServlet extends HttpServlet
 2:  {
 3:    protected void doGet(HttpServletRequest req,
 4:                         HttpServletResponse res)
 5:              throws ServletException, IOException
 6:    {
 7:      ...
 8:      ServletContext context = getServletConfig().getServletContext();
 9:      context.setAttribute("name", "wael");
10:      String strCourse = context.getAttribute("course-key" );
11:       ..
12:     }
```

Servlet Demo – cont'd

3. Inter-servlets communication

```
1:  public class FooServlet extends HttpServlet
 2:  {
 3:    protected void doGet(HttpServletRequest req,
 4:                         HttpServletResponse res)
 5:              throws ServletException, IOException
 6:    {
 7:      ...
 8:      ServletContext context = getServletConfig().getServletContext();
 9:      BarInterface bar = (BarInterface)context.getServlet("BarServlet");
10:       bar.bar();
11:       ..
12:     }
```

```
1:  public interface
          BarInterface
 2:  {
 3:    public void bar();
 4:  }
```

```
1:    public class BarServlet extends HttpServlet implements BarInterface
 2:   {
 3:     public void bar() {
 5:      System.err.println(""bar() called"");
 6:    }
```

Servlet issues

1. Writing Thread-safe servlets
   - setHeader(…)

Servlets issue

2. Servlets Performance

>> Use init() method to cache static data

>> Use StringBuffer rather than using + operator when you concatenate multiple strings

>> Use print() method rather than println() method

>> Use ServletOutputStream rather than PrintWriter to send binary data

>> Initialize the PrintWriter with proper size

>> Flush the data partly

>> Minimize code in the synchronized block

>> Use thread pool for your servlet engine

Servlet issues

3. Servlet initialization parameters

- Where does a servlet get its initialization values?
  - From the web.xml file
- Inside <servlet> </servlet>

```
<init-param>
    <param-name>myName</param-name>
    <param-value>myValue</param-value>
</init-param>
```

- In the servlet code:
  - `String myValue = getServletConfig().getInitParameter("myName");`

Servlet issues

4. Servlets context parameters

- Where does a servlet get its context?
  - From the web.xml file
- Not inside <servlet> </servlet>

```
<context-param>
    <param-name>myName</param-name>
    <param-value>myValue</param-value>
</context-param>
```

- In the servlet code:
  - `String myValue =getServletContext().getInitParameter("myName");`

Servlet issues

5. Accessing Servlet Resources

```
InputStream confIn = getClass().getResourceAsStream("myservlet.cfg");
```

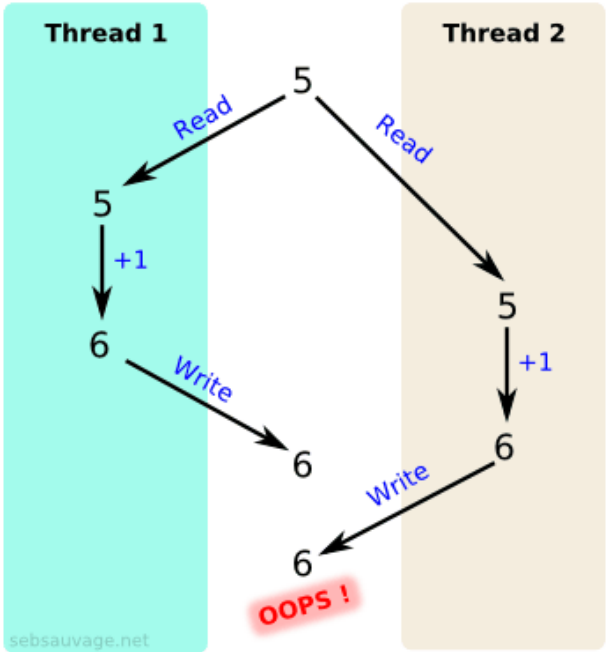
Servlets Implementation

1. Multi-threaded model
   a. Servlet containers create a new Java thread for each request.
   b. The new thread is given an object reference to the requested servlet, which issues the response through the same thread.
      - Each request thread for your servlet runs as if a single user were accessing it alone,

        you can use static variables to store and present information that is common to all threads, (e.g.counter)

Thread safety

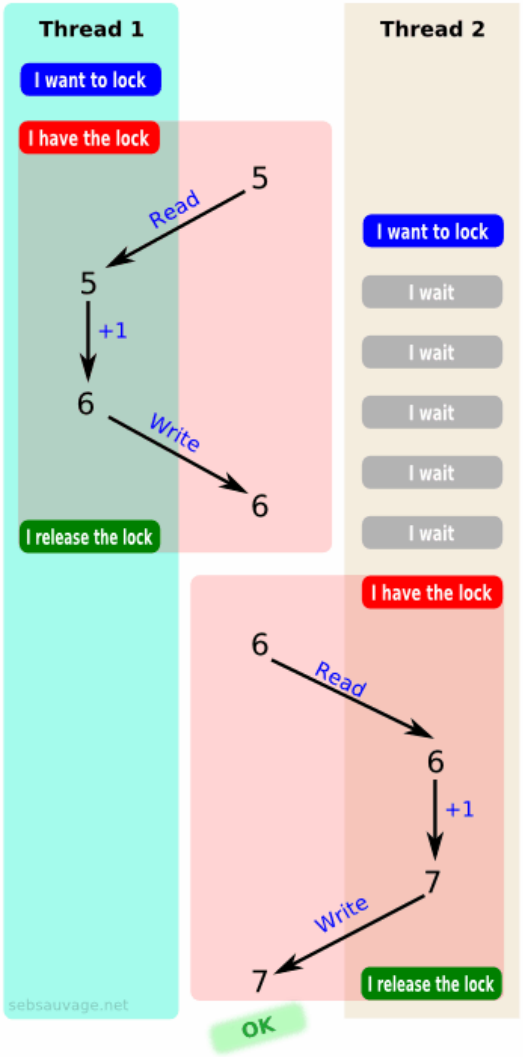
- Thread problems can occur when:
  - One Thread is writing to (modifying) an object at the same time another Thread is reading it
  - Two (or more) Threads are trying to write to the same object at the same time



Thread safety – cont'd

- Thread problems cannot (in general) be detected by the Java runtime system
  - Instead, thread problems cause random, mysterious, non-replicable corruption of data
- There are simple steps that you can take to avoid many threading problems
  - However, threading is very error-prone and can be extremely difficult to ensure that you have it right

Thread safety – cont'd



Thread safety in servlets

- Each request, and therefore each Thread, has its own request and response objects
  - Therefore, these are inherently Thread-safe
  - *Local* variables (including parameters) of your service methods are thread-safe
  - *Instance* variables are thread-unsafe
    - You don't have multiple servlet objects—you have multiple *Threads* using the *same* servlet object
- Application (context) scope is shared by all servlets
  - Therefore, context attributes are inherently Thread-*un*safe
- Session attributes are *not* completely Thread-safe
  - It is possible to have multiple simultaneous requests from the same session

Protecting context attributes

- To protect context attributes, synchronize on the ServletContext object
  - Example

```
synchronized(getServletContext()) {
    getServletContext().setAttribute("foo", "22");
    getServletContext().setAttribute("bar", "42");

    out.println(getServletContext().getAttribute("foo"));
    out.println(getServletContext().getAttribute("bar"));
}
```

- This will protect you from any other code that *also* synchronizes on the ServletContext
- It will *not* protect you from code that doesn't synchronize
  - But this is the best we can do

Protecting session attributes

- To protect session attributes, synchronize on the HttpSession object
  - Example:

```
HttpSession session = request.getSession();
synchronized(session) {
    session.setAttribute("foo", "22");
    session.setAttribute("bar", "42");

    out.println(session.getAttribute("foo"));
    out.println(session.getAttribute("bar"));
}
```

- This will protect you from any other code that *also* synchronizes on the HttpSession

Unsafe Servlet example

```
public class SomeServlet extends HTTPServlet
{
    private String someParam;

    protected void doGet(HttpServletRequest request,
                                HttpServletResponse response)
                                        throws ServletException, IOException {

          someParam = request.getParameter("someParam");
          processParam();
    }

 private void processParam() {
    // Do something with someParam
    }
```

Safe Servlet example

- A thread safe alternative is:

```
public class SomeServlet extends HTTPServlet
 {
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
                         throws ServletException, IOException {

        String someParam = request.getParameter("someParam");
        processParam(someParam);
     }

    private void processParam( String strInput ) {
        // Do something with strInput
     }
```

Thread safety in class assignments

- In reality, the servlets you write for this course are not going to service thousands of requests per second
- However...
- Bottom line: Try your best to make your servlets thread-safe, even though we can't test them for thread safety

Servlets Implementation

2. Single thread-model

```
public class SingleThreadServlet extends HttpServlet
    implements SingleThreadServlet
  {
// Standard HTTP servlet methods


  }
```

Servlets && Application Architecture

1. Mutliple-Servlets Architecture
   - One per functionality:
     - LoginServlet, LogoutServlet, RegisterUserServlet,…

Servlets && Application Architecture

2. Single-Servlet Architecture

- One façade servlet, action in parameter list

  http://www.xyz.com/go?action=register

  http://www.xyz.com/go?action=login

Servlets && Application Architecture

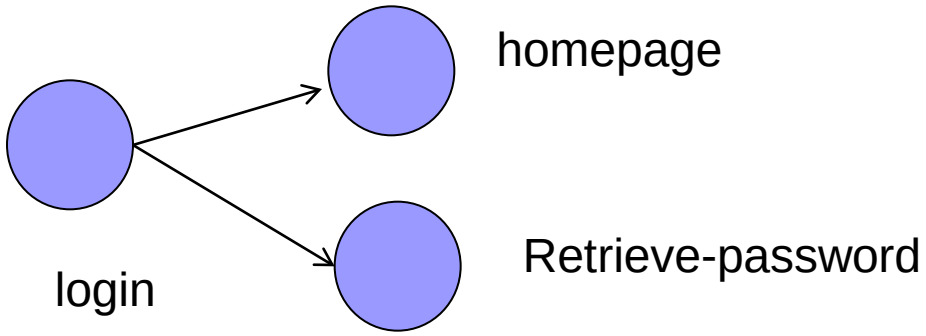
2. Single-Servlet Architecture

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

public class GoServlet extends HttpServlet {

    protected void doGet(HttpServletRequest req,
                         HttpServletResponse res)
                throws ServletException, IOException  {

        String strAction = req.getParameter("action");
        ……

    }

    protected void doPost(HttpServletRequest req,
                          HttpServletResponse res)
                                throws ServletException, IOException {

                                doGet( req, res );
        }
}
```

Servlets && Application Architecture

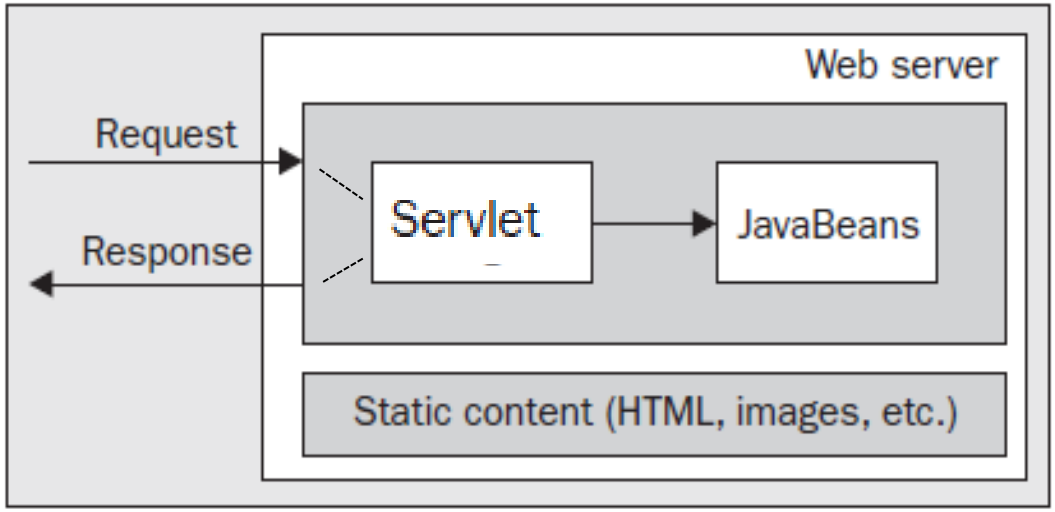
3. Routing-Servlet Architecture

- A mix model
  - One entry point, forward to one of several possible servlets
  - Each can forward to one of several possible servlets



Servlets && Application Architecture

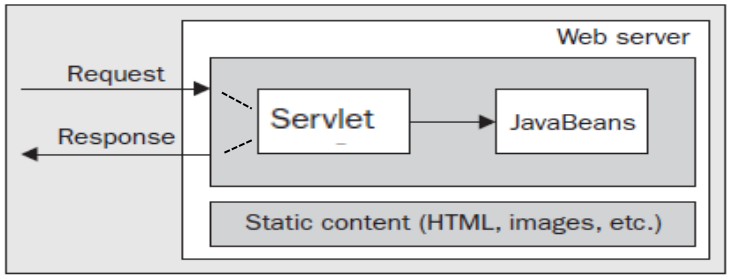
1. Mutliple-Servlets Architecture
2. Single-Servlet Architecture
3. Routing-Servlet Architecture



Model 1 Architecture

Servlets

Model 1 Architecture

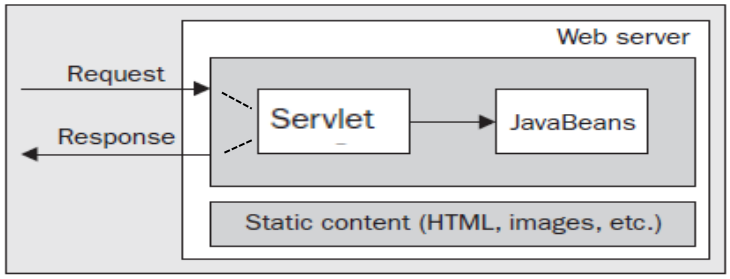


Maintainability Problems

- structure of the application will be embodied within the pages. The pages will include diverse functionality, such as:
  - Complex business logic
  - Links to other parts of the web application
  - The names of pages to which forms should be submitted

Servlets

Model 1 Architecture

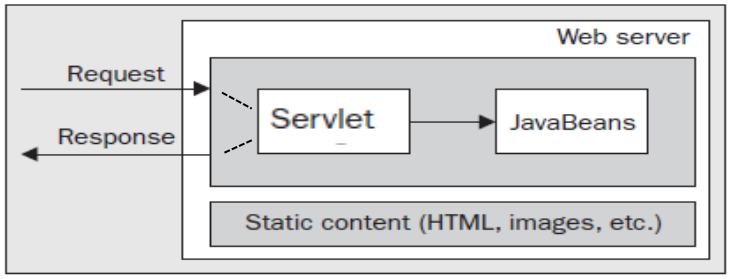


Extensibility Problems

- Pages contain diverse functionality ➔ tightly-coupled together
  - Changing some functionality could ripple causing bugs

Servlets

Model 1 Architecture



Security Problems

- Each page in a restricted area of a website would have to perform its own security checks to ensure that the user is logged in and that the user is permitted to see the contents of the page.

Servlets

```
25          out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
26             "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
27             "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
28
29          out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
30
31          // head section of document
32          out.println( "<head>" );
33          out.println(
34             "<title>Processing get requests with data</title>" );
35          out.println( "</head>" );
36
37          // body section of document
38          out.println( "<body>" );
39          out.println( "<h1>Hello " + firstName + ",<br />" );
40          out.println( "Welcome to Servlets!</h1>" );
41          out.println( "</body>" );
42
43          // end XHTML document
44          out.println( "</html>" );
45          out.close();  // close stream to complete the page
46       }
47    }
```

Servlets

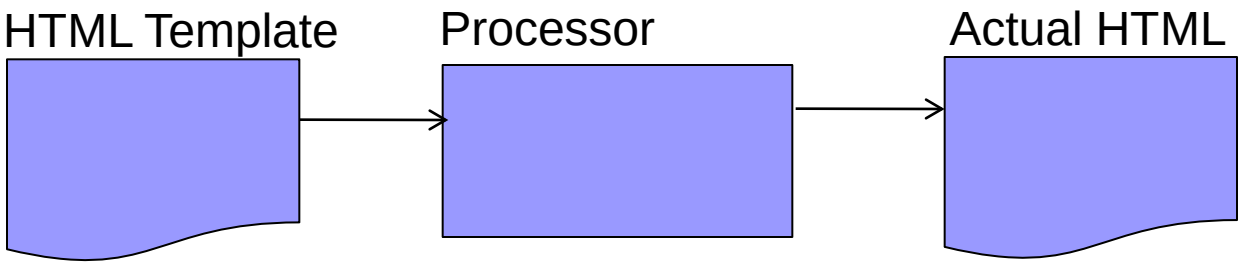

```
25          out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
26             "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
27             "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
28
29          out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
30
31          // head section of document
32          out.println( "<head>" );
33          out.println(
34             "<title>Processing get requests with data</title>" );
35          out.println( "</head>" );
36
37          // body section of document
38          out.println( "<body>" );
39          out.println( "<h1>Hello " + firstName + ",<br />" );
40          out.println( "Welcome to Servlets!</h1>" );
41          out.println( "</body>" );
42
43          // end XHTML document
44          out.println( "</html>" );
45          out.close();  // close stream to complete the page
46       }
47    }
```

Lots of HTML and Lots of Java…

JSP

JSP

- Java Server Pages
- .jsp page is really a template of what the actual html should be…



Components of a JSP Page

- A .jsp file contains
  - JSP Elements
    - Instructions to the JSP container about what code to generate and how it should operate
    - These elements have specific start and end tags that identify them to the JSP compiler
  - Fixed Template data (aka HTML)
    - Everything else that is not recognized by the JSP container
    - Usually HTML data, passed through unmodified
    - Results in HTML code that is sent to the client
  - Any combination of the two

JSP Elements

- 3 types of JSP Elements
- Directives
  - Instructions to the JSP container that describes what code should be generated
  - <%@ *directive-name* [*attribute*="value" *attribute*="value" ….] %>
  - Three standard directives
    - page directive
    - include directive
    - taglib directive
- Scripting Elements
    - Lets you specify Java code inside the .jsp page
- Actions
    - Specify existing components that should be used and otherwise control the behavior of JSP engine

JSP scripting elements

- <%= ***expression*** %>
  - The ***expression*** is evaluated and the result is inserted into the HTML page
- <% ***code*** %>
  - The ***code*** is inserted into the servlet's service method
  - If ***code*** contains declarations, they become *local* variables of the service method
  - This construction is called a scriptlet
- <%! ***declarations*** %>
  - The ***declarations*** are inserted into the servlet *class,* not into a method
  - Hence, declarations made here become *instance* variables

Example JSP

- <HTML>
  <BODY>
  Hello! The time is now <%= new java.util.Date() %>
  </BODY>
  </HTML>

- Notes:
  - The <%= … %> tag is used, because we are computing a *value* and inserting it into the HTML
  - The fully qualified name (java.util.Date) is used, instead of the short name (Date), because we haven't yet talked about how to do import declarations

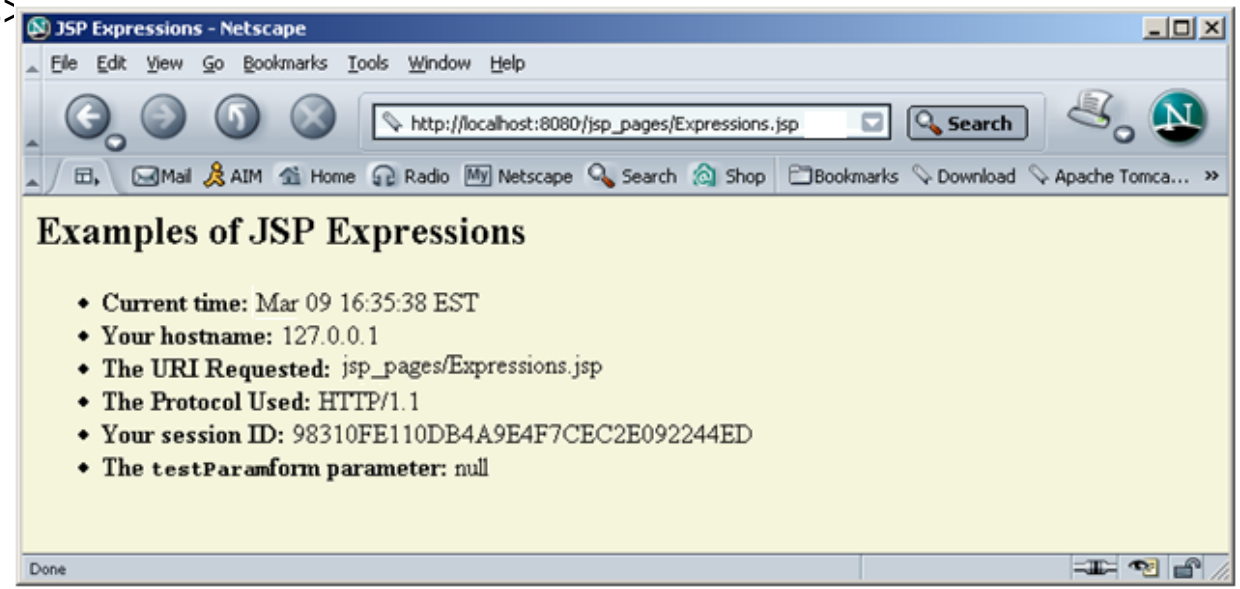
JSP Variables

- You can declare your own variables, as usual..

- JSP provides several predefined variables
  - request : The HttpServletRequest parameter
  - response : The HttpServletResponse parameter
  - session : The HttpSession associated with the request
  - out : A JspWriter (like a PrintWriter) used to send output to the client

Example Using JSP Expressions

```
<HTML> <BODY> <H2>JSP Expressions</H2>
<UL>
  <LI>Current time: <%= new java.util.Date() %>
  <LI>Your hostname: <%= request.getRemoteHost() %>
 <LI>The URI Requested: <%= request.getRequestURI() %>
 <LI>The Protocol Used: <%= request.getProtocol() %>
  <LI>Your session ID: <%= session.getId() %>
  <LI>The <CODE>testParam</CODE> form parameter:
      <%= request.getParameter("testParam") %>
</UL> </BODY> </HTML>
```



JSP Scriptlets

- Scriptlets are enclosed in <% … %> tags
  - Scriptlets are executable code and do not directly affect the HTML
  - Scriptlets *may* write into the HTML with out.print(***value***) and out.println(***value***)
  - Example:

```
<% String queryData = request.getQueryString();
   out.println("Attached GET data: " + queryData); %>
```

- Scriptlets are inserted into the servlet *exactly as written*, and are not compiled until the entire servlet is compiled
  - Example:

```
<% if (Math.random() < 0.5) { %>
        Have a <B>nice</B> day!
<% } else { %>
        Have a <B>lousy</B> day!
<% } %>
```

JSP/Servlet Correspondence

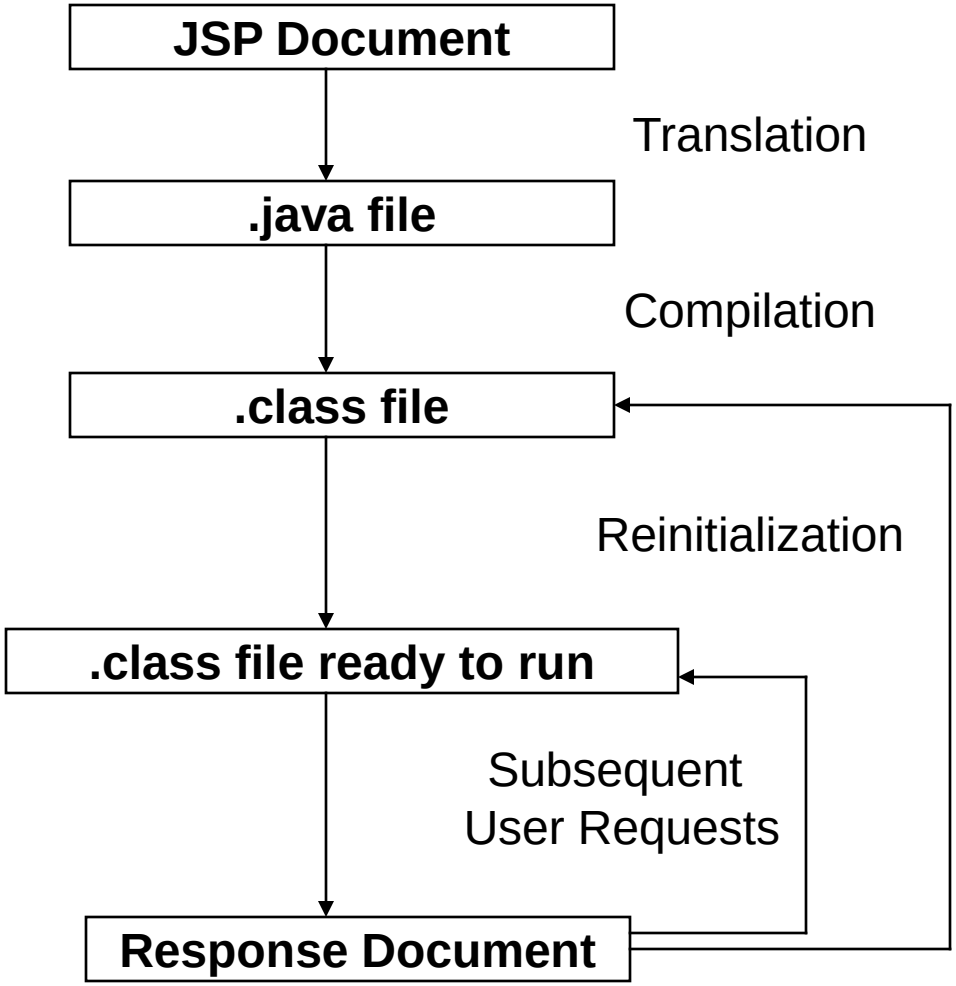
- Original .jsp

```
<H1>A Random Number</H1>
<%= Math.random() %>
```

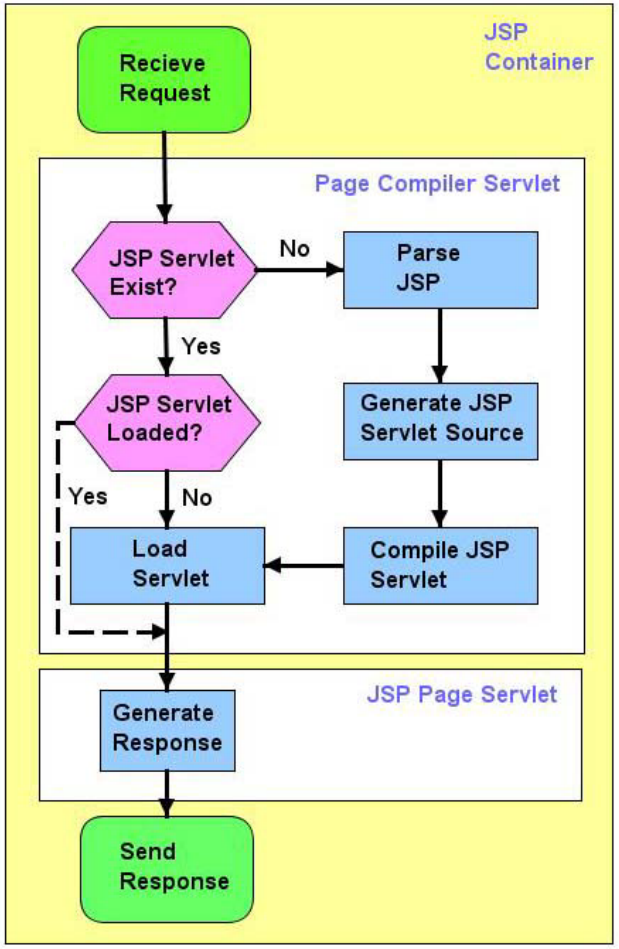
- Possible resulting servlet code

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
        throws ServletException, IOException {
          response.setContentType("text/html");
          HttpSession session = request.getSession(true);
          JspWriter out = response.getWriter();
           out.println("<H1>A Random Number</H1>");
          out.println(Math.random());
...
```

How does JSP work?

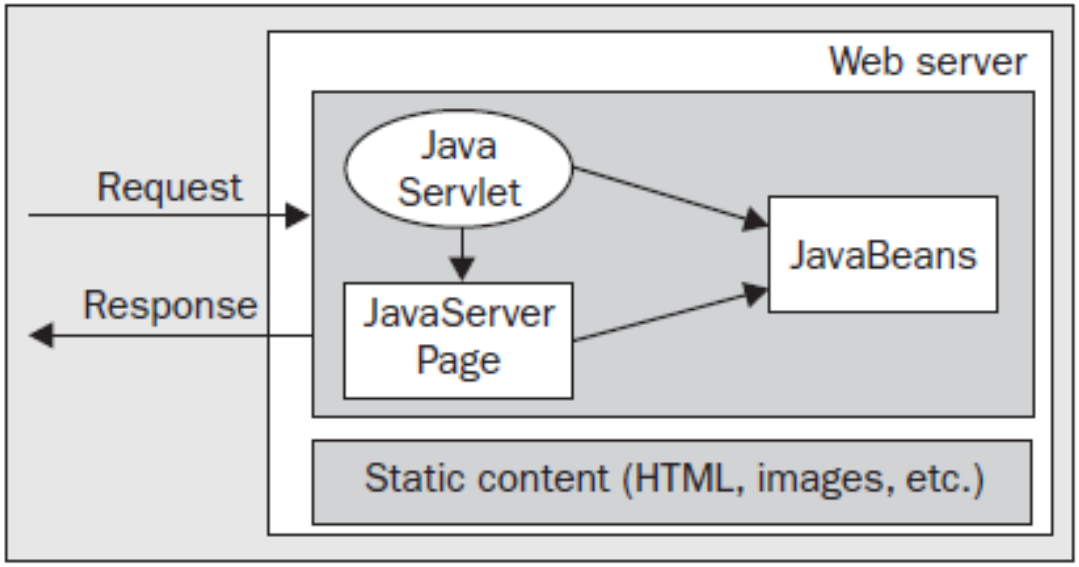


JSP


JSP
Container
Recieve
Request
Page Compiler Servlet
JSP Servlet
Exist?
No
Parse
JSP
Yes
JSP Servlet
Loaded?
Generate JSP
Servlet Source
Yes
No
Load
Servlet
Compile JSP
Servlet
JSP Page Servlet
Generate
Response
Send
Response

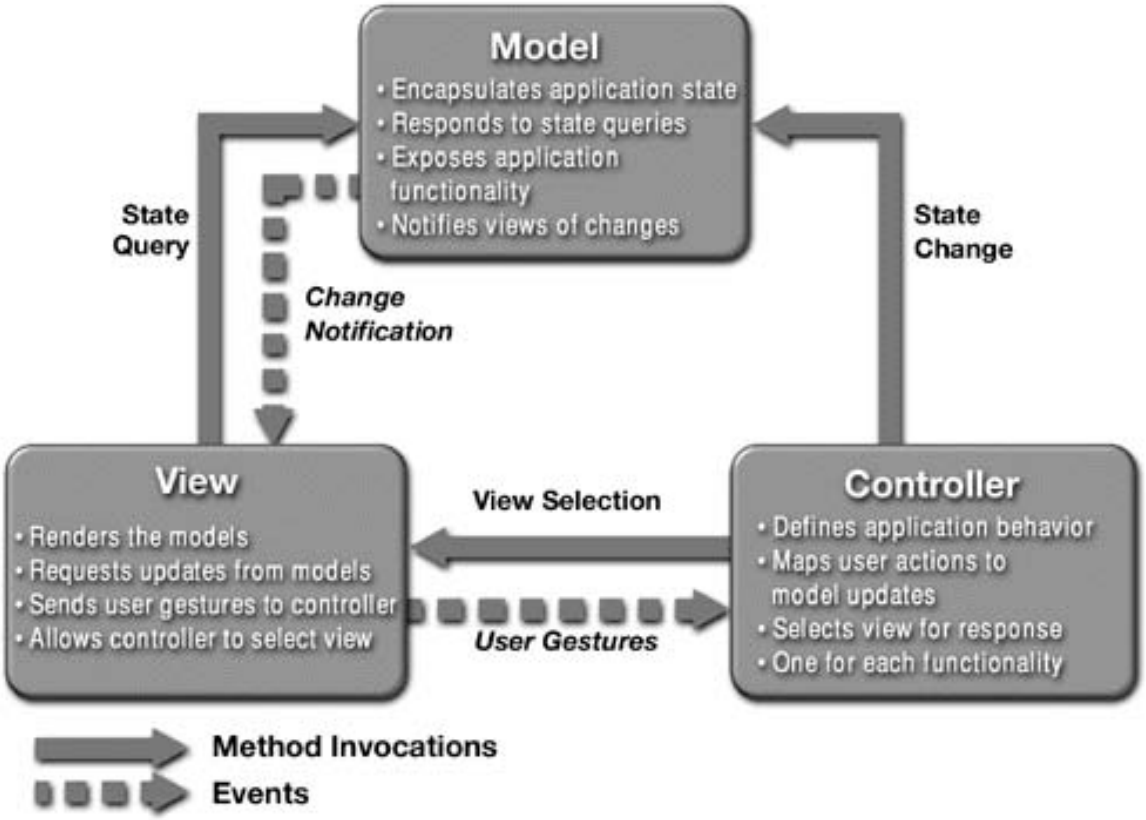
Servlets+JSP

Model 2 Architecture



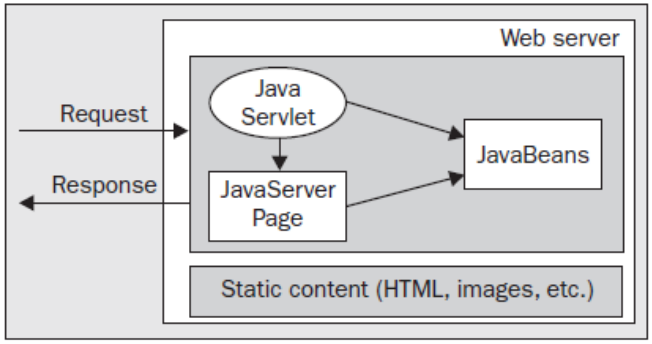
Servlets+JSP

Model 2 Architecture = MVC Design Pattern



Servlets+JSP

Model 2 Architecture

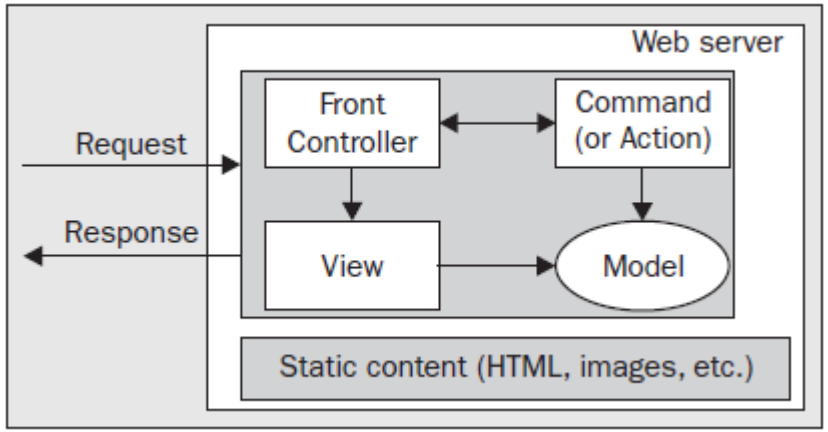


Controller Servlet

- Act as a single point of entry for requests
- Process the requests, accessing and modifying the underlying model
- Delegate the task of presenting information to a specific view component

Servlets+JSP

Model 2 Architecture + Command Pattern

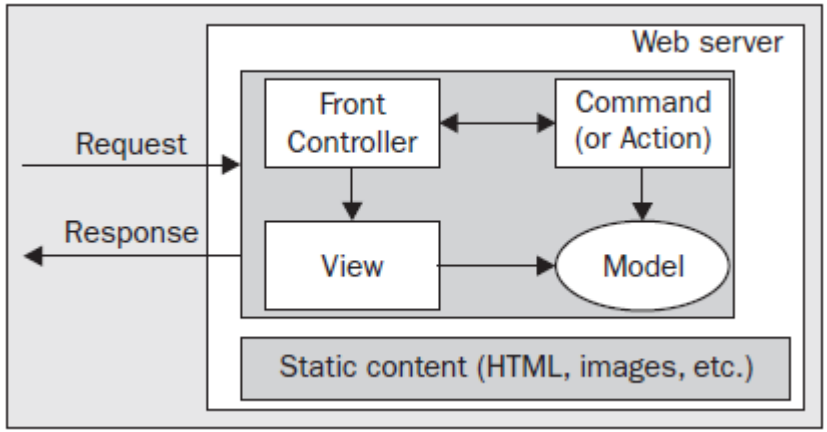


Controller Servlet

- Act as a single point of entry for requests
- Process the requests by dispatching it to a specific command
  - (command access and modify the underlying object model)
- Delegate the task of presenting information to a specific view component

Servlets+JSP

Model 2 Architecture + Command Pattern

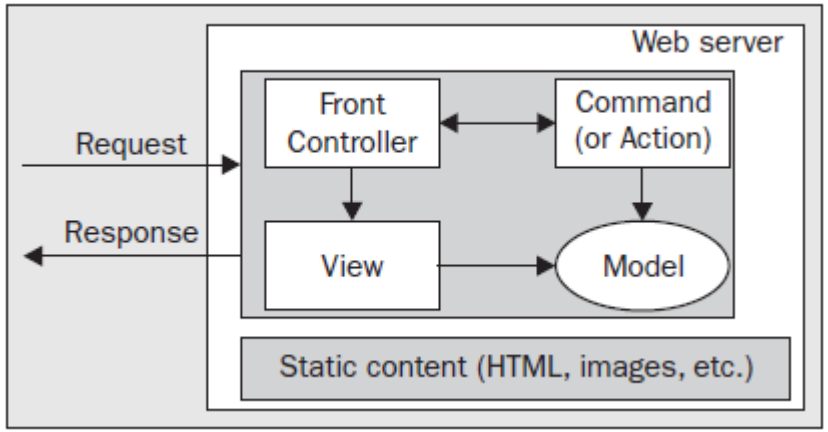



```
import java.util.HashMap;
public class ActionHelper {
private static HashMap actions = new HashMap();
static {
actions.put("ViewTopic", "forum.ViewTopicAction");
actions.put("Login", "forum.LoginAction");
actions.put("Logout", "forum.LogoutAction");
actions.put("NewResponse", "forum.NewResponseAction");
actions.put("ProcessNewResponse",
"forum.ProcessNewResponseAction");
actions.put("DeleteResponse", "forum.DeleteResponseAction");
}
```

```
public static Action getAction(String
name) {
Action action = null;
try {
    Class c =
      Class.forName((String)actions.
            get(name));
     action = (Action)c.newInstance();
}
catch (Exception e) {
e.printStackTrace();
}
return action;
} }
```

Servlets+JSP

Model 2 Architecture + Command Pattern




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

public class FrontController extends HttpServlet {

protected void processRequest(     HttpServletRequest req,
                                   HttpServletResponse res)
                        throws ServletException, IOException {

String actionName = req.getPathInfo().substring(1);

Action action = ActionHelper.getAction(actionName);

String nextView = action.process(req, res);

RequestDispatcher dispatcher =
    getServletContext().
        getRequestDispatcher(nextView);
    dispatcher.forward(req, res);
}

protected void doGet(HttpServletRequest req,
                                   HttpServletResponse res)
throws ServletException, IOException {
     processRequest(req, res);
}
```

Servlets+JSP

Model 2 Architecture + Command Pattern