Java Server Pages - JSP

Dr. Michael Whitney

** Slides influenced by both textbooks

The Need for JSP

With servlets, it is easy to:

- Read form data.
- Read HTTP request headers.
- Set HTTP status codes and response headers.
- Use cookies and session tracking.
- Share data among servlets.
- Remember data between requests.

But, it sure is a pain to

- Use those println statements to generate HTML.
- Maintain that HTML.

The JSP Framework

Idea:

- Use regular HTML for most of page.
- Mark servlet code with special tags.
- Entire JSP page gets translated into a servlet (once), and servlet is what actually gets invoked (for each request).

Example:

```
<!DOCTYPE html><html>
<head><title>Order Confirmation</title></head>
<body>
    <h1>Order Confirmation</h1>
    Thanks for ordering
    <em><%= request.getParameter("title") %></em>!
</body>
</html>
```

Benefits of JSP

Although JSP technically can't do anything servlets can't do, JSP makes it easier to:

- Write HTML.
- Read and maintain the HTML.

JSP makes it possible to:

- Use standard HTML tools such as Macromedia DreamWeaver or Adobe GoLive.
- Have different members of your team do the HTML layout than do the Java programming.

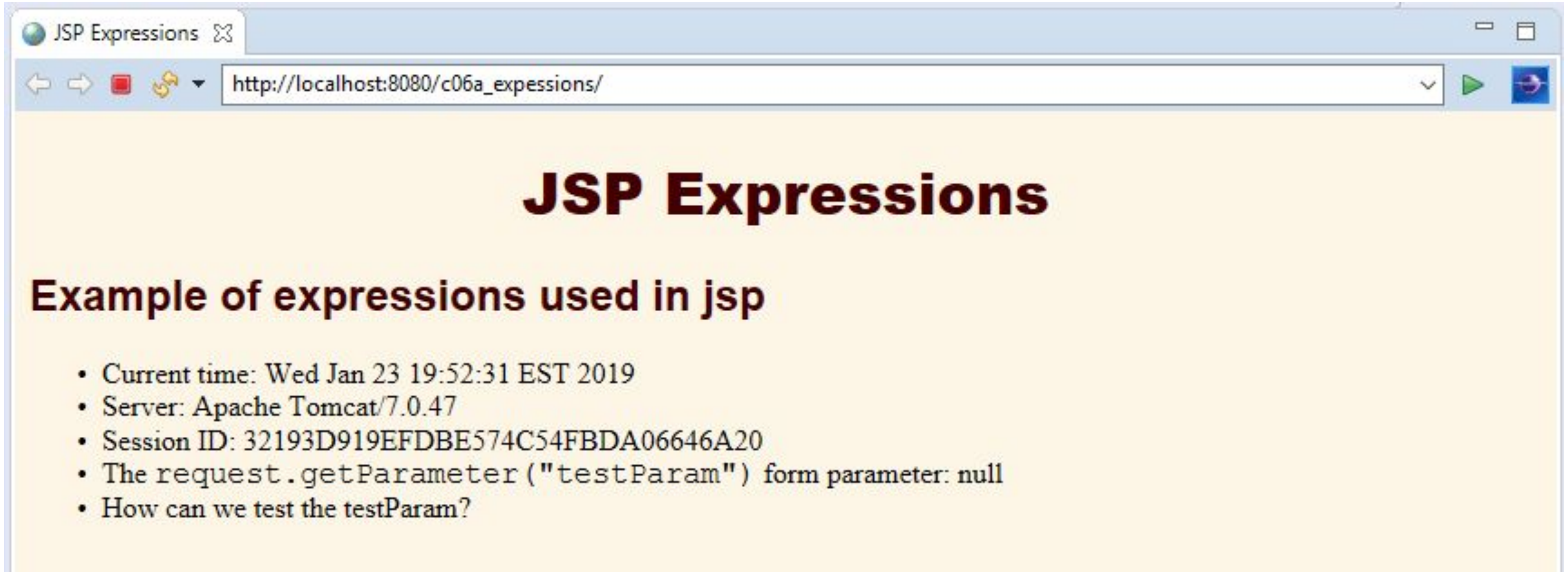
JSP encourages you to:

- Separate the (Java) code that creates the content from the(HTML) code that presents it.

JSP Example

```
<html>
<head>
        <title>jsp expressions</title>
        <link rel="stylesheet" href="jsp-styles.css" type="text/css">
</head>
<body>
        <h1>jsp expressions</h1>
        <ul>
                <li>current time: <%= new java.util.date() %>
                <li>server: <%= application.getserverinfo() %>
                <li>session id: <%= session.getid() %>
                <li>the <code>testparam</code> form parameter: <%= request.getparameter("testparam") %></li>
        </ul>
</body></html>
```

Result



http://localhost:8080/c06a_expessions/

JSP Expressions

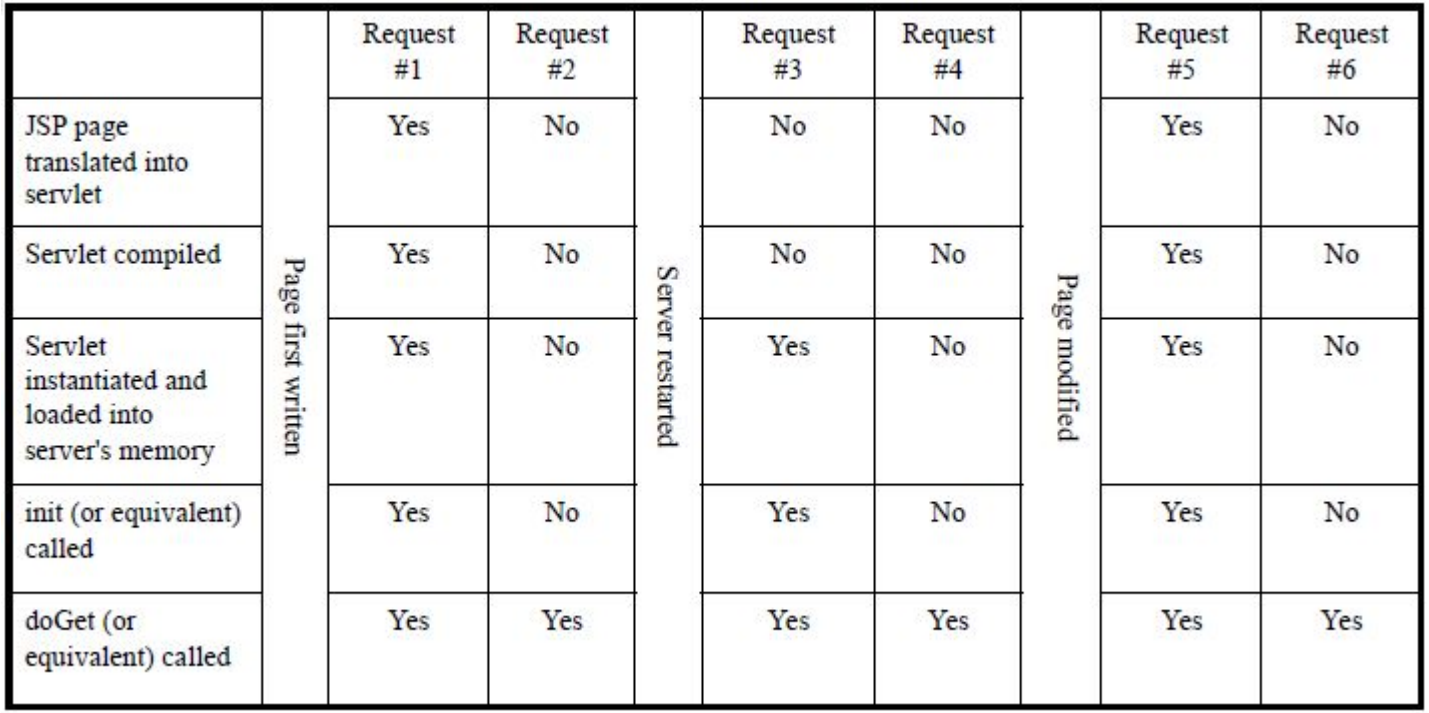
Example of expressions used in jsp

- Current time: Wed Jan 23 19:52:31 EST 2019
- Server: Apache Tomcat/7.0.47
- Session ID: 32193D919EFDBE574C54FBDA06646A20
- The `request.getParameter("testParam")` form parameter: null
- How can we test the testParam?

Quick Summary

- JSP makes it easier to create and maintain HTML, while still providing full access to servlet code.
- JSP pages get translated into servlets:
    - It is the servlets that run at request time.
    - Client does not see anything JSP-related.
- You still need to understand servlets:
    - Understanding how JSP really works.
    - Servlet code called from JSP.
    - Knowing when servlets are better than JSP.
    - Mixing servlets and JSP.

The JSP Lifecycle



		Request #1	Request #2		Request #3	Request #4		Request #5	Request #6
JSP page translated into servlet	Page first written	Yes	No	Server restarted	No	No	Page modified	Yes	No
Servlet compiled		Yes	No		No	No		Yes	No
Servlet instantiated and loaded into server's memory		Yes	No		Yes	No		Yes	No
init (or equivalent) called		Yes	No		Yes	No		Yes	No
doGet (or equivalent) called		Yes	Yes		Yes	Yes		Yes	Yes

Invoking Java Code with JSP Scripting Elements

Uses of JSP Constructs

Simple Application

$\downarrow$

Complex Application

- Scripting elements calling servlet code directly
- Scripting elements calling servlet code indirectly (by means of utility classes)
- Beans
- Servlet/JSP combo (MVC)
- MVC with JSP expression language
- Custom tags
- MVC with beans, custom tags, and a framework like Struts or JSF

Strategy: Limit Java Code in JSP

You have two options

- Put 25 lines of Java code directly in the JSP page
- Put those 25 lines in a separate Java class and put 1 line in the JSP page that invokes it

Why is the second option *much* better?

- **Development**. You write the separate class in a Java environment (editor or IDE), not an HTML environment
- **Debugging**. If you have syntax errors, you see them immediately at compile time. Simple print statements can be seen.
- **Testing**. You can write a test routine with a loop that does 10,000 tests and reapply it after each change.
- **Reuse**. You can use the same class from multiple pages.

Basic Syntax

html text

- <h1>blah</h1>
- Passed through to client. Really turned into servlet code that looks like
  - out.print("<h1>blah</h1>");

html comments

- <!-- comment -->
- same as other html: passed through to client

jsp comments

- <%-- comment --%>
- not sent to client

To get <% to show up on the page, use <\%

Types of Scripting Elements

Expressions:

- Format: <%= expression %>.
- Evaluated and inserted into the servlet's output and sent to the client each time the page is requested - results in something like out.print(expression).

Scriptlets:

- Format: <% code %>.
- Statement or statements that are executed each time the page is requested.

Declarations:

- Format: <%! code %>.
- Inserted verbatim into the body of the servlet class, outside of any existing methods. Becomes part of the class definition when page is translated into a servlet.

Directive:

- Format: <%@ code %>.
- To set conditions that apply to the entire JSP.

JSP Expressions

Format

- <%= Java Expression %>

Result

- Expression evaluated, converted to String, and placed into HTML page at the place it occurred in JSP page

Examples

- Current time: <%= new java.util.Date() %>
- Your hostname: <%= request.getRemoteHost() %>

XML-compatible syntax

- <jsp:expression>Java Expression</jsp:expression>
- You cannot mix versions within a single page. You must use XML for *entire* page if you use jsp:expression.

JSP/Servlet Correspondence: Expression

Original JSP

```
<h1>A Random Number</h1>
<%= Math.random() %>
```

Representative resulting servlet code

```
    public void _jspService(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html");
        HttpSession session = request.getSession();
        JspWriter out = response.getWriter();
        out.println("<h1>A Random Number</h1>");
        out.println(Math.random());
    ...}
```

Predefined Variables

request

- The HttpServletRequest (1st argument to service/doGet)

response

- The HttpServletResponse (2nd arg to service/doGet)

out

- The Writer (a buffered version of type JspWriter) used to send output to the client

session

- The HttpSession associated with the request (unless disabled with the session attribute of the page directive)

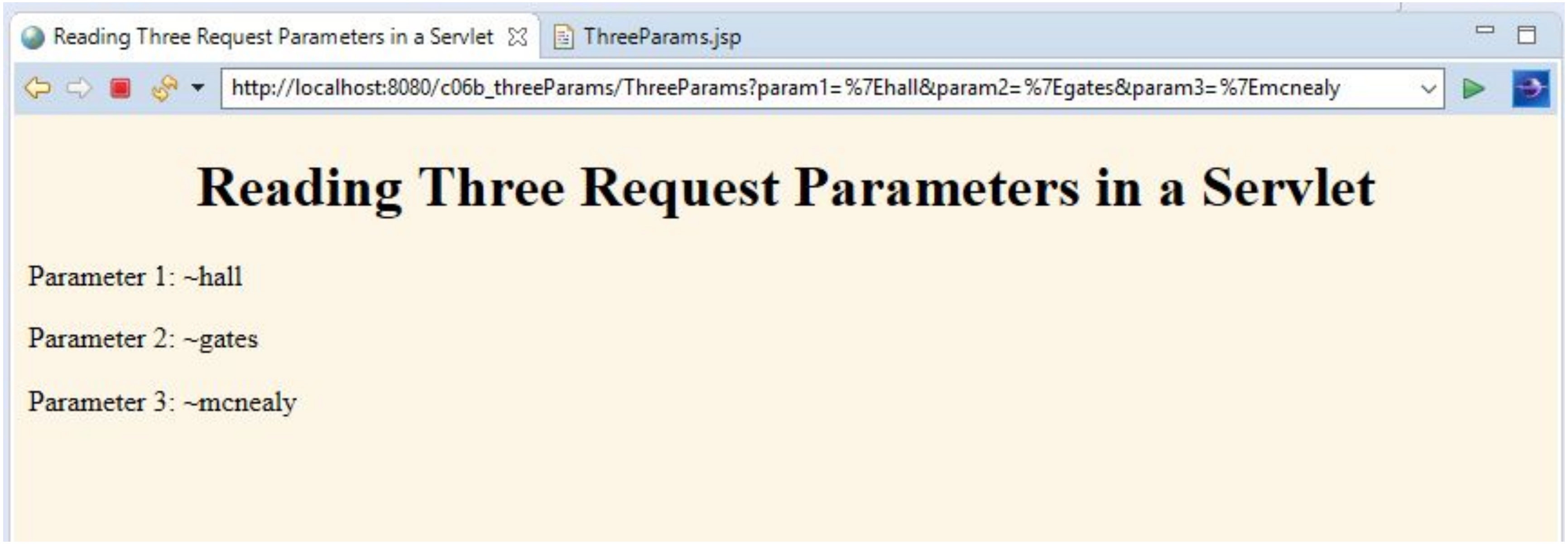
application

- The ServletContext (for sharing data) as obtained via getServletContext()

Comparing Servlets to JSP: Reading Three Params (Servlet)

```
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">\n" +
                        "<h1>" + title + "</h1>\n" +
                        "<ul>\n" +
                                " <li>param1: " + request.getparameter("param1") + "</li>\n" +
                                " <li>param2: " + request.getparameter("param2") + "</li>\n" +
                                " <li>param3: " + request.getparameter("param3") + "</li>\n" +
                        "</ul>\n" +
                        "</body></html>");
        }
}
```

Reading Three Params (Servlet): Result



http://localhost:8080/c06b_threeParams/ThreeParams?param1=%7Ehall&param2=%7Egates&param3=%7Emcnealy

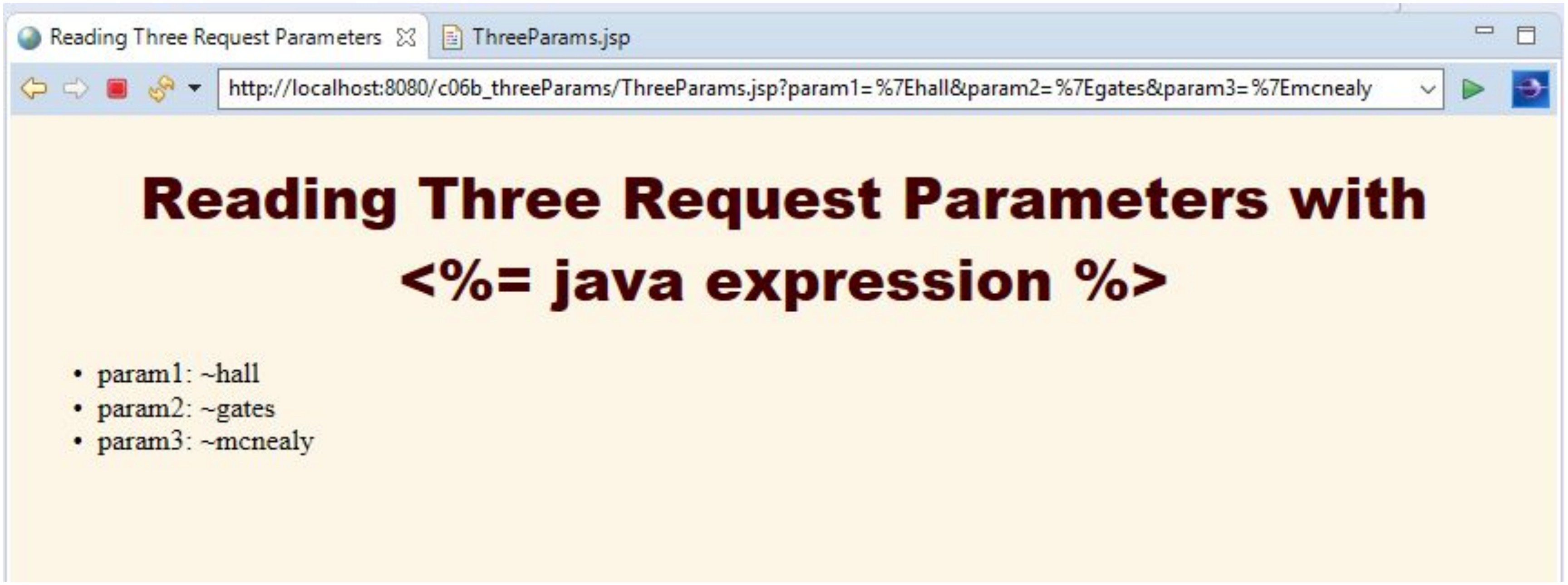
Reading Three Request Parameters in a Servlet

```
Parameter 1: ~hall

Parameter 2: ~gates

Parameter 3: ~mcnealy
```

Comparing Servlets to JSP: Reading Three Params (JSP)

```
<html>
        <head>
                <title>reading three request parameters</title>
        </head>
        <body>
                <h1>Reading Three Request Parameters with<br> &lt;%= java expression %&gt;</h1>
                <ul>
                        <li><b>param1</b>: <%= request.getparameter("param1") %> </li>
                        <li><b>param2</b>: <%= request.getparameter("param2") %> </li>
                        <li><b>param3</b>: <%= request.getparameter("param3") %> </li>
                </ul>
                <!-- Expressions are evaluated and inserted into the servlet's output -->
</body></html>
```

Reading Three Params (Servlet): Result



http://localhost:8080/c06b_threeParams/ThreeParams.jsp?param1=%7Ehall&param2=%7Egates&param3=%7Emcnealy

Reading Three Request Parameters with <%= java expression %>

- param1: ~hall
- param2: ~gates
- param3: ~mcnealy

JSP Scriptlets

Format

```
<% Java Code %>
```

Result

Code is inserted verbatim into servlet's _jspService.

Example

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

XML-compatible syntax

```
<jsp:scriptlet>Java Code</jsp:scriptlet>
```

JSP/Servlet Correspondence: Scriptlet

Original JSP

```
    <h2>foo</h2>
    <%= bar() %>        <!-- this is an expression that places its contentent directly into the html -->
    <% baz(); %>        <!-- this is a scriptlet that will be run in the _jspService -
```

Representative resulting servlet code

```
    public void _jspService(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
            response.setContentType("text/html");
            HttpSession session = request.getSession();
            JspWriter out = response.getWriter();
            out.println("<h2>foo</h2>");
            out.println(bar());
            baz();
            …
    }
```

JSP Scriptlets: Example

Suppose you want to let end users customize the background color of a page

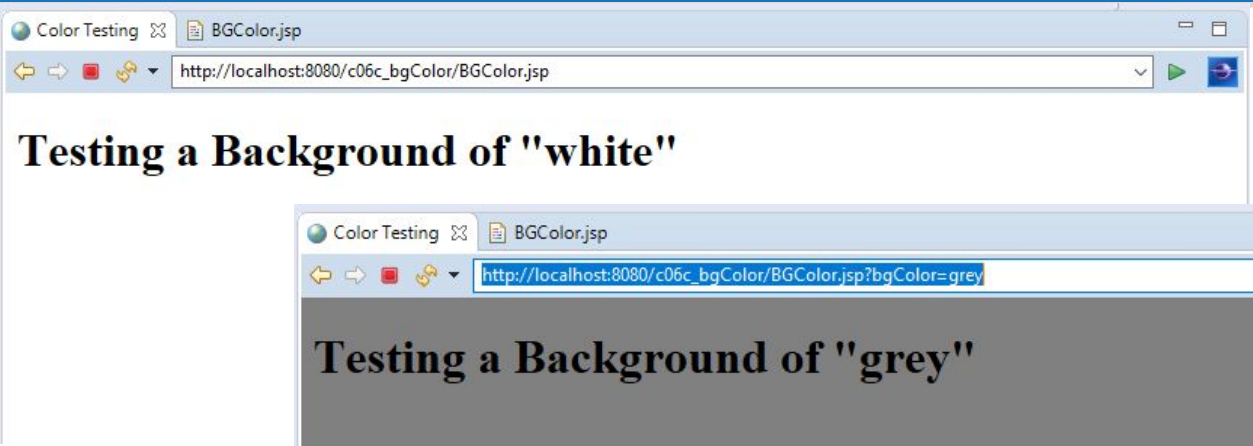
What is wrong with the following code (think bigger picture vs syntax)?

```
<body style="background-color:<%= request.getParameter("bgColor") %>">
```

JSP Scriptlets: Example

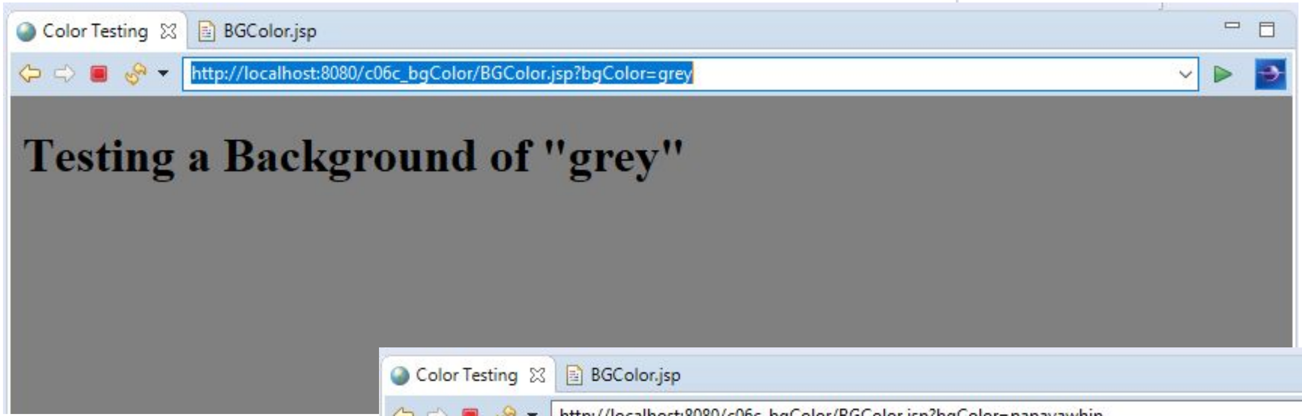
```
<html>
        <head>
                <title>color testing</title>
        </head>
        <%      string bgColor = request.getparameter("bgColor");
                if ((bgColor == null)||(bgColor.trim().equals(""))){
                        bgColor = "white";
                }
        %>
        <body style="background-color:<%= bgColor %>">
                <h2 align="center">testing a background of <%= bgcolor %></h2>
        </body>
</html>
```

Predefined Variables



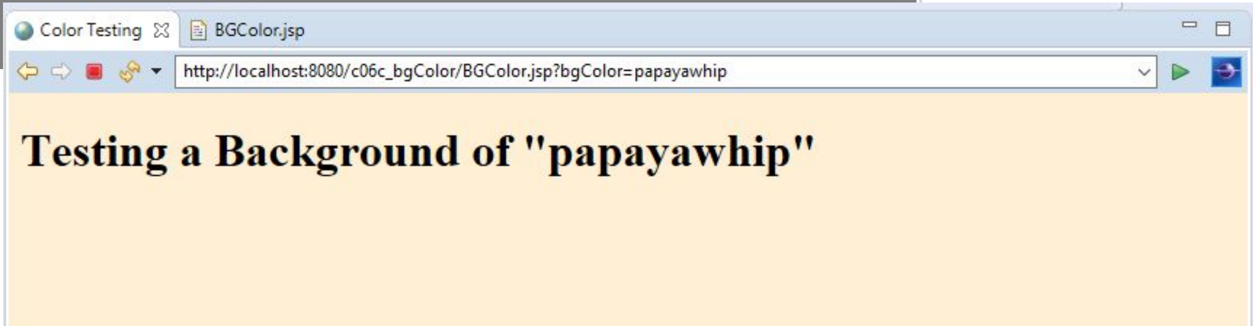
http://localhost:8080/c06c_bgColor/BGColor.jsp

Testing a Background of "white"



http://localhost:8080/c06c_bgColor/BGColor.jsp?bgColor=grey

Testing a Background of "grey"



http://localhost:8080/c06c_bgColor/BGColor.jsp?bgColor=papayawhip

Testing a Background of "papayawhip"

Using Scriptlets to Make Parts of the JSP File Conditional

Point

- Scriplets are inserted into servlet exactly as written.

Example

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

Representative result

```
if (Math.random() < 0.5) {
    out.println("Have a <strong>nice</strong> day!");
} else {
    out.println("Have a <strong>lousy</strong> day!");
}
```

JSP Declarations

Format

```
<%! Java Code %>
```

Result

Code is inserted verbatim into servlet's class definition, outside of any existing methods

Examples

```
<%! private int someField = 5; %>
<%! private void someMethod(...) {...} %>
```

Design consideration

Fields are clearly useful. For methods, it is usually better to define the method in a separate Java class.

XML-compatible syntax

```
<jsp:declaration>Java Code</jsp:declaration>
```

JSP Declarations

Original JSP

```
<h1>Some Heading</h1>
<%!
private String randomHeading() {
return("<H2>" + Math.random() + "</H2>");
}
%>
<%= randomHeading() %>
```

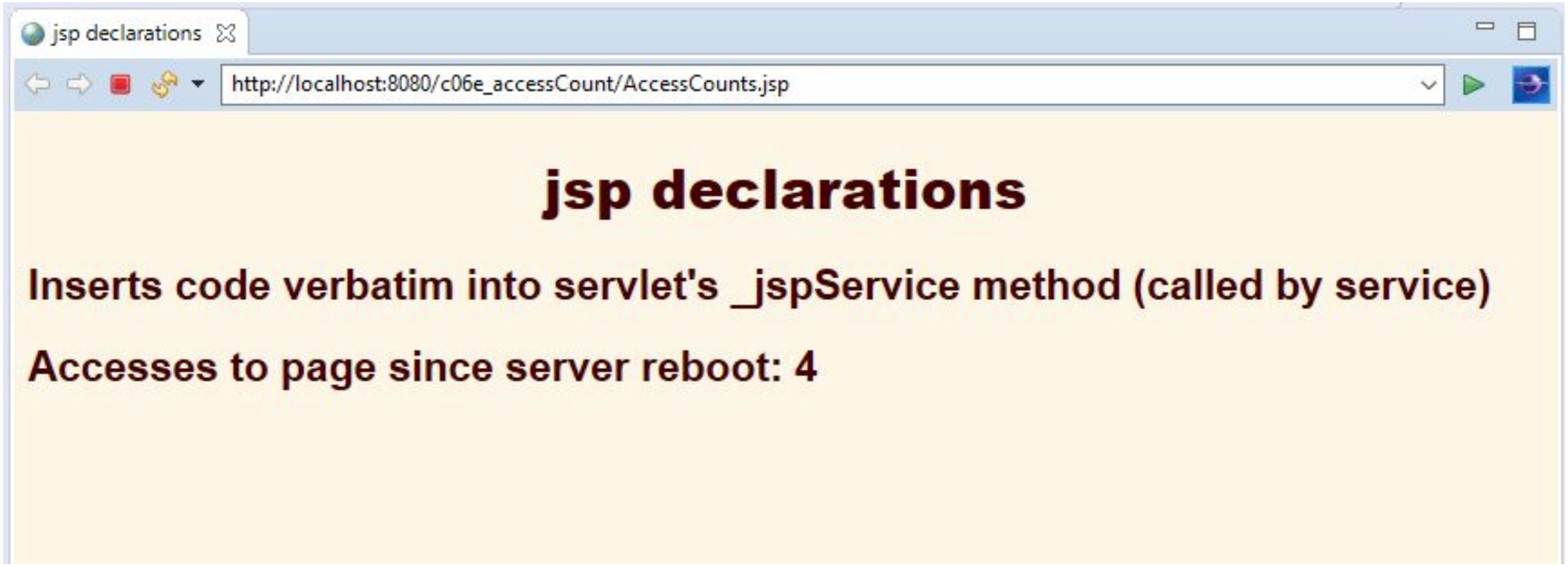
Possible resulting servlet code

```
public class xxxx implements HttpJspPage {
        private String randomHeading() {
                return("<h2>" + Math.random() + "</h2>");
        }
public void _jspService(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        HttpSession session = request.getSession();
        JspWriter out = response.getWriter();
        out.println("<h1>Some Heading</h1>");
        out.println(randomHeading());
        ...
} ...}
```

JSP Declarations: Example

```
<html>
        <head>
        <title>jsp declarations</title>
        </head>
<body>
        <h1>jsp declarations</h1>
        <%! private int accesscount = 0; %>
        <h2>accesses to page since server reboot: <%= ++accesscount %></h2>
</body>
</html>
```

Predefined Variables



jsp declarations

Inserts code verbatim into servlet's _jspService method (called by service)

Accesses to page since server reboot: 4

Comparing Expressions, Scriptlets and Declarations

Task 1

- Output a bulleted list of five random ints from 1 to 10.
    - Since the structure of this page is fixed and we use a separate helper class for the randomInt method, JSP expressions are all that is needed.

Task 2

- Generate a list of between 1 and 10 entries (selected at random), each of which is a number between 1 and 10.
    - Because the number of entries in the list is dynamic, a JSP scriptlet is needed.

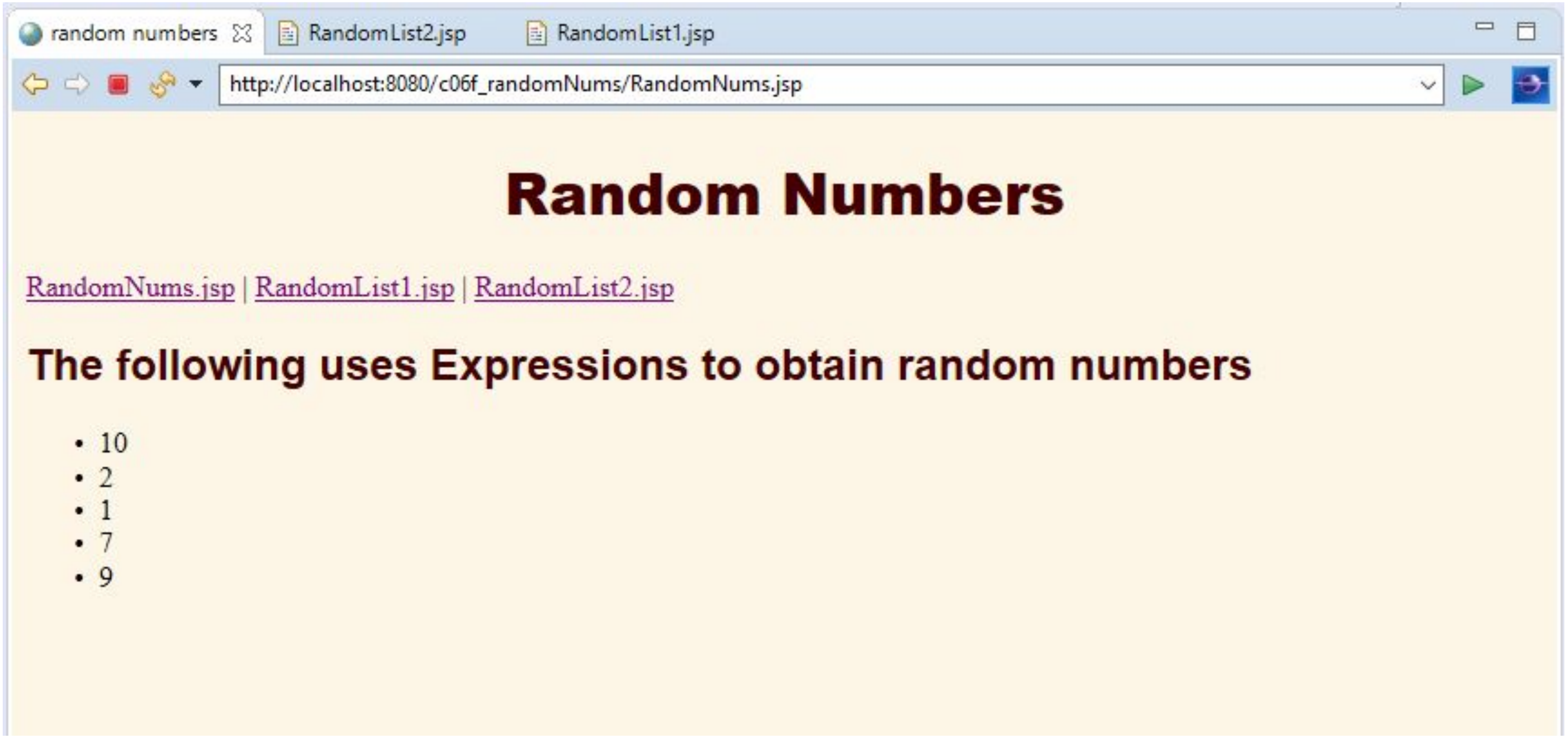
Task 3

- Generate a random number on the first request, then show the same number to all users until the server is restarted.
    - Instance variables (fields) are the natural way to accomplish this persistence. Use JSP declarations for this.

Task 1: JSP Expressions

```
<html>
        <head>
                <title>random numbers</title>
        </head>
<body>
        <h1>random numbers</h1>
        <ul>
                <li><%= coreservlets.RanUtilities.randomInt(10) %></li>
                <li><%= coreservlets.RanUtilities.randomInt(10) %></li>
                <li><%= coreservlets.RanUtilities.randomInt(10) %></li>
                <li><%= coreservlets.RanUtilities.randomInt(10) %></li>
                <li><%= coreservlets.RanUtilities.randomInt(10) %></li>
        </ul>
</body></html>
```

Task 1: JSP Expressions (Result)



Random Numbers

RandomNums.jsp | RandomList1.jsp | RandomList2.jsp

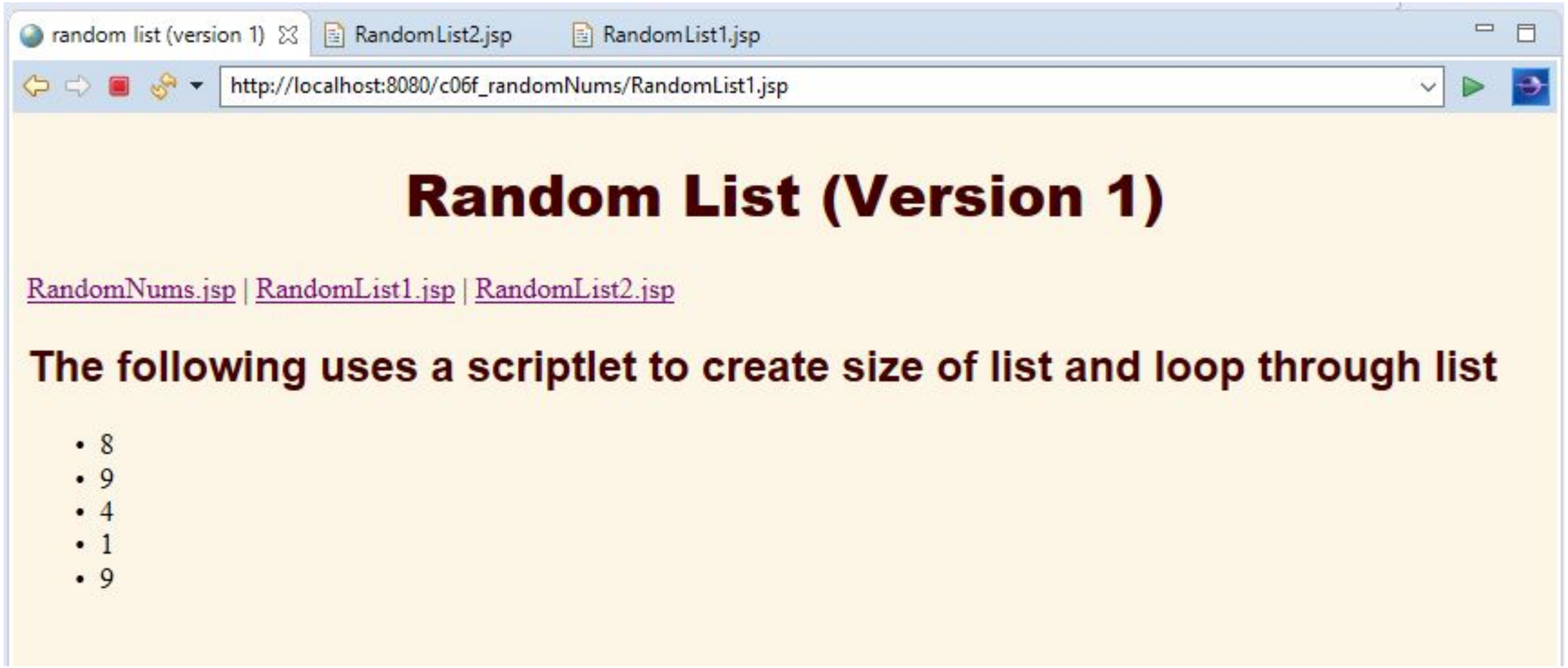
The following uses Expressions to obtain random numbers

- 10
- 2
- 1
- 7
- 9

Task 2: JSP Scriptlets (Code: Version 1)

```
<body>
        <h1>Random List (Version 1)</h1>
        <nav><a href="RandomNums.jsp">RandomNums.jsp</a> |
                <a href="RandomList1.jsp">RandomList1.jsp</a> |
                <a href="RandomList2.jsp">RandomList2.jsp</a></nav>
        <h2>The following uses a scriptlet to create size of list and loop through list</h2>
        <ul>
                <%
                int numEntries = coreservlets.RanUtilities.randomInt(10);
                for(int i=0; i<numEntries; i++) {
                        out.println("<li>" + coreservlets.RanUtilities.randomInt(10) + "</li>");
                }
                %>
        </ul>
</body>
```

Task 2: JSP Scriptlets (Code: Version 1)



http://localhost:8080/c06f_randomNums/RandomList1.jsp

Random List (Version 1)

RandomNums.jsp | RandomList1.jsp | RandomList2.jsp

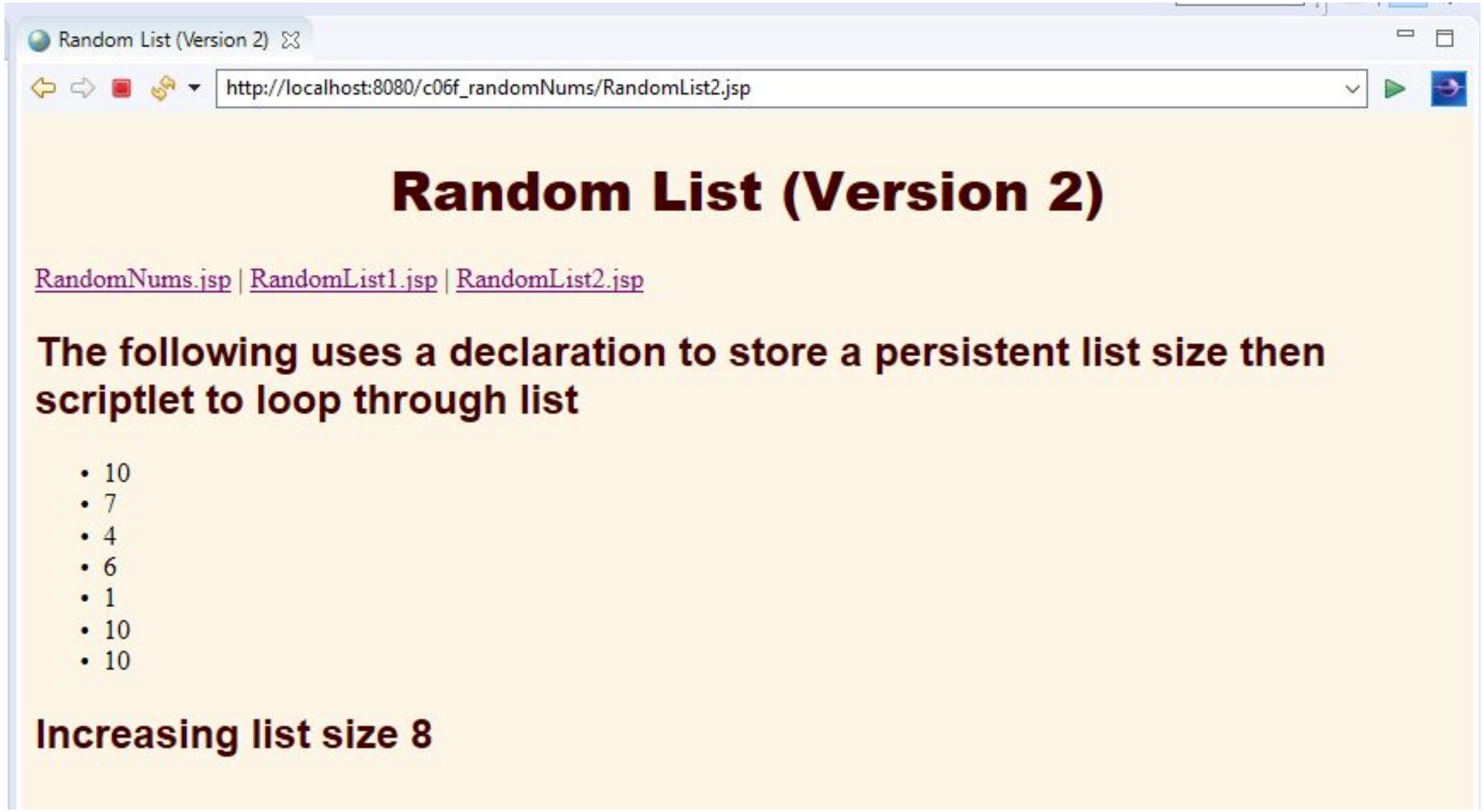
The following uses a scriptlet to create size of list and loop through list

- 8
- 9
- 4
- 1
- 9

Task 2: JSP Scriptlets & Declaration (Version 2 – Dangling Brace)

```
<body>
        <h1>random list (version 2)</h1>
        <nav><a href="RandomNums.jsp">RandomNums.jsp</a> |
                <a href="RandomList1.jsp">RandomList1.jsp</a> |
                <a href="RandomList2.jsp">RandomList2.jsp</a></nav>
        <h2>The following uses a declaration to store a persistent list size then scriptlet to loop through list</h2>
        <ul>
                <%! private int numEntries = coreservlets.RanUtilities.randomInt(5); %> <!-- declaration -->
                <%
                        for(int i=0; i < numEntries; i++) {
                %>
                                <li><%= coreservlets.RanUtilities.randomInt(5) %> </li>
                <%      }       %>
        </ul>
        <h2>Increasing list size <%= ++numEntries %></h2>  <!-- updates declaration which is persistent -->
</body>                                                    <!-- refreshing the page increases list by 1 -->
```

Task 2: JSP Scriptlets (Result: Version 1)


Random List (Version 2)

http://localhost:8080/c06f_randomNums/RandomList2.jsp

Random List (Version 2)

RandomNums.jsp | RandomList1.jsp | RandomList2.jsp

The following uses a declaration to store a persistent list size then scriptlet to loop through list

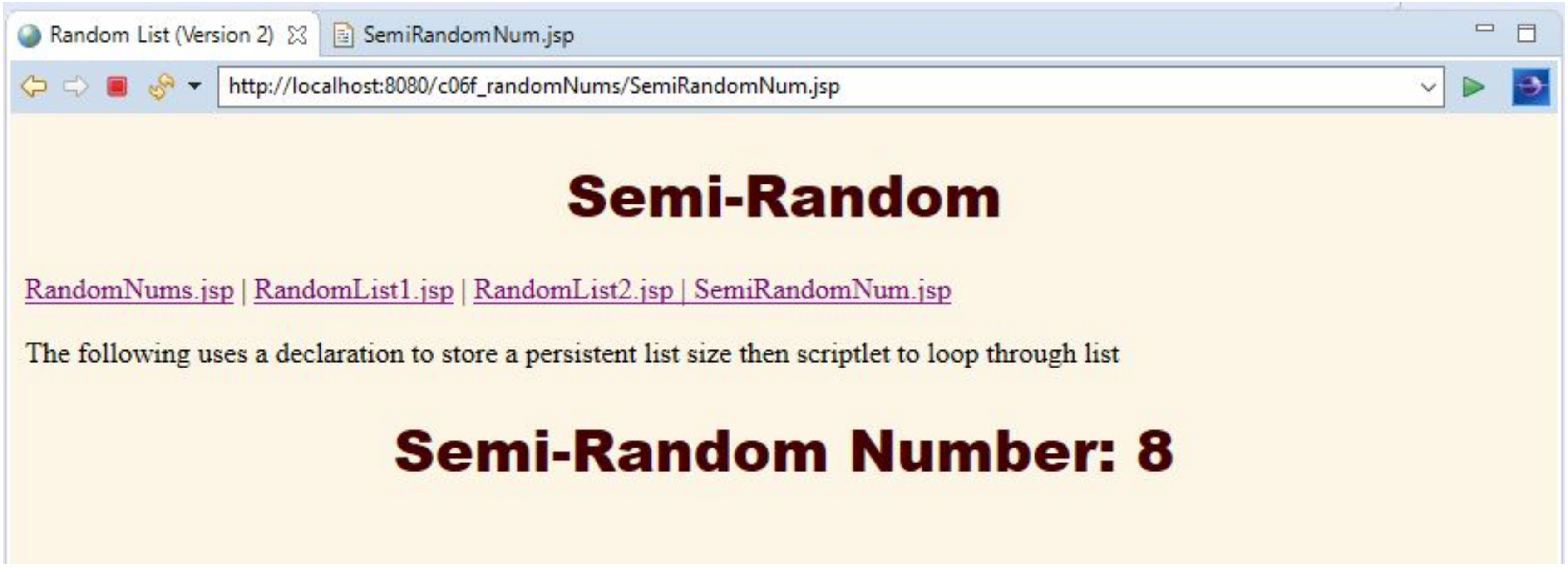
- 10
- 7
- 4
- 6
- 1
- 10
- 10

Increasing list size 8

Task 3: JSP Declarations

```
<body>
        <h1>Semi-Random</h1>
        <nav><a href="RandomNums.jsp">RandomNums.jsp</a> |
                <a href="RandomList1.jsp">RandomList1.jsp</a> |
                <a href="RandomList2.jsp">RandomList2.jsp</a></nav>
        <p>The following uses a declaration to store a persistent list size then scriptlet to loop through list</p>

        <%!
        private int randomNum = coreservlets.RanUtilities.randomInt(10);
        %>
        <H1>Semi-Random Number:<%= randomNum %></h1>
</body>
```

Task 3: JSP Scriptlets (Result)



Semi-Random

RandomNums.jsp | RandomList1.jsp | RandomList2.jsp | SemiRandomNum.jsp

The following uses a declaration to store a persistent list size then scriptlet to loop through list

Semi-Random Number: 8

JSP Summary

JSP Expressions

- Format: <%= expression %>
- Wrapped in out.print and inserted into _jspService

JSP Scriptlets

- Format: <% code %>
- Inserted verbatim into the servlet's _jspService method

JSP Declarations

- Format: <%! code %>
- Inserted verbatim into the body of the servlet class

Predefined variables

- request, response, out, session, application

Limit the Java code that is directly in page

- Use helper classes,beans, servlet/JSP combo (MVC), JSP expression language