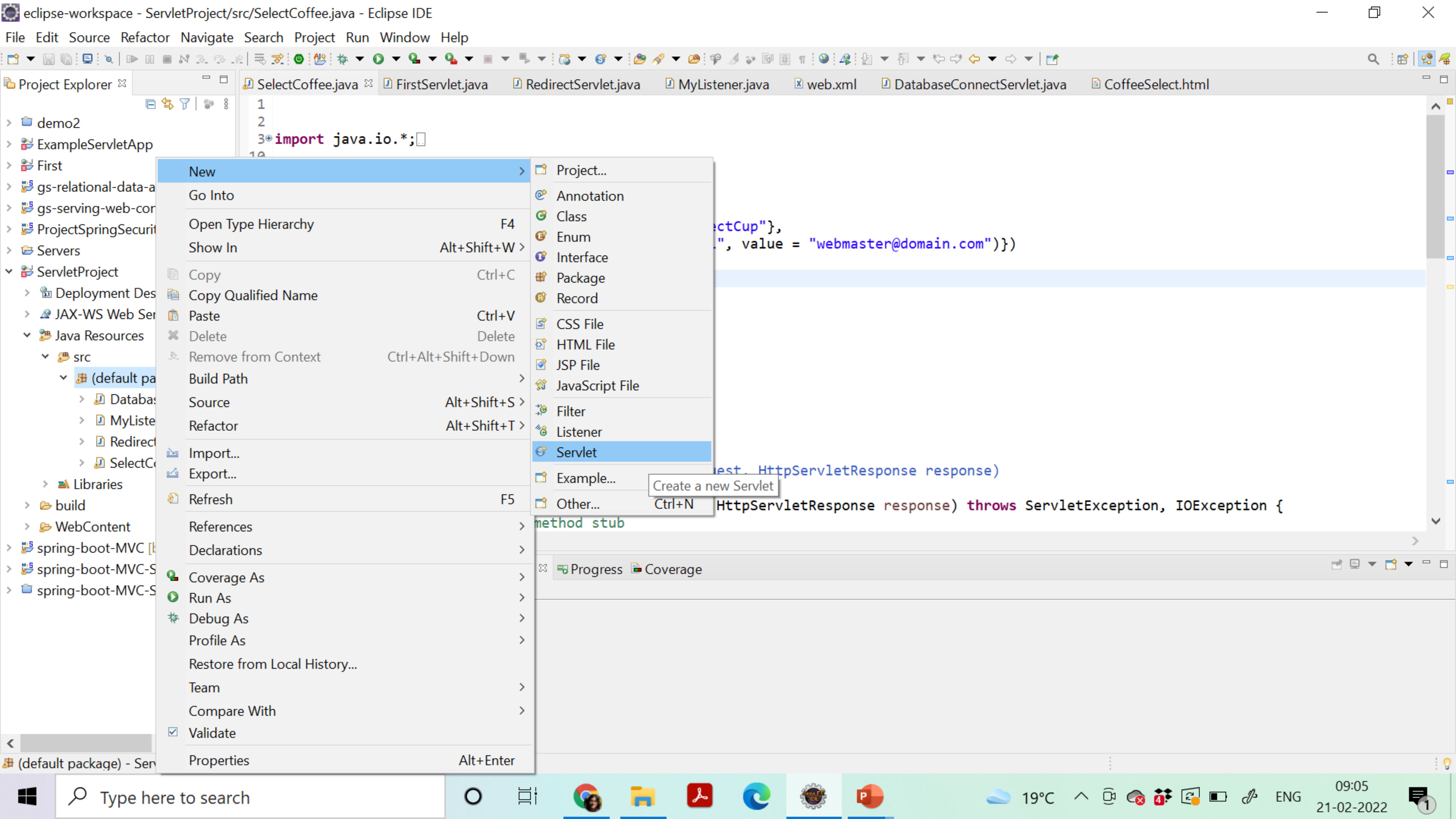
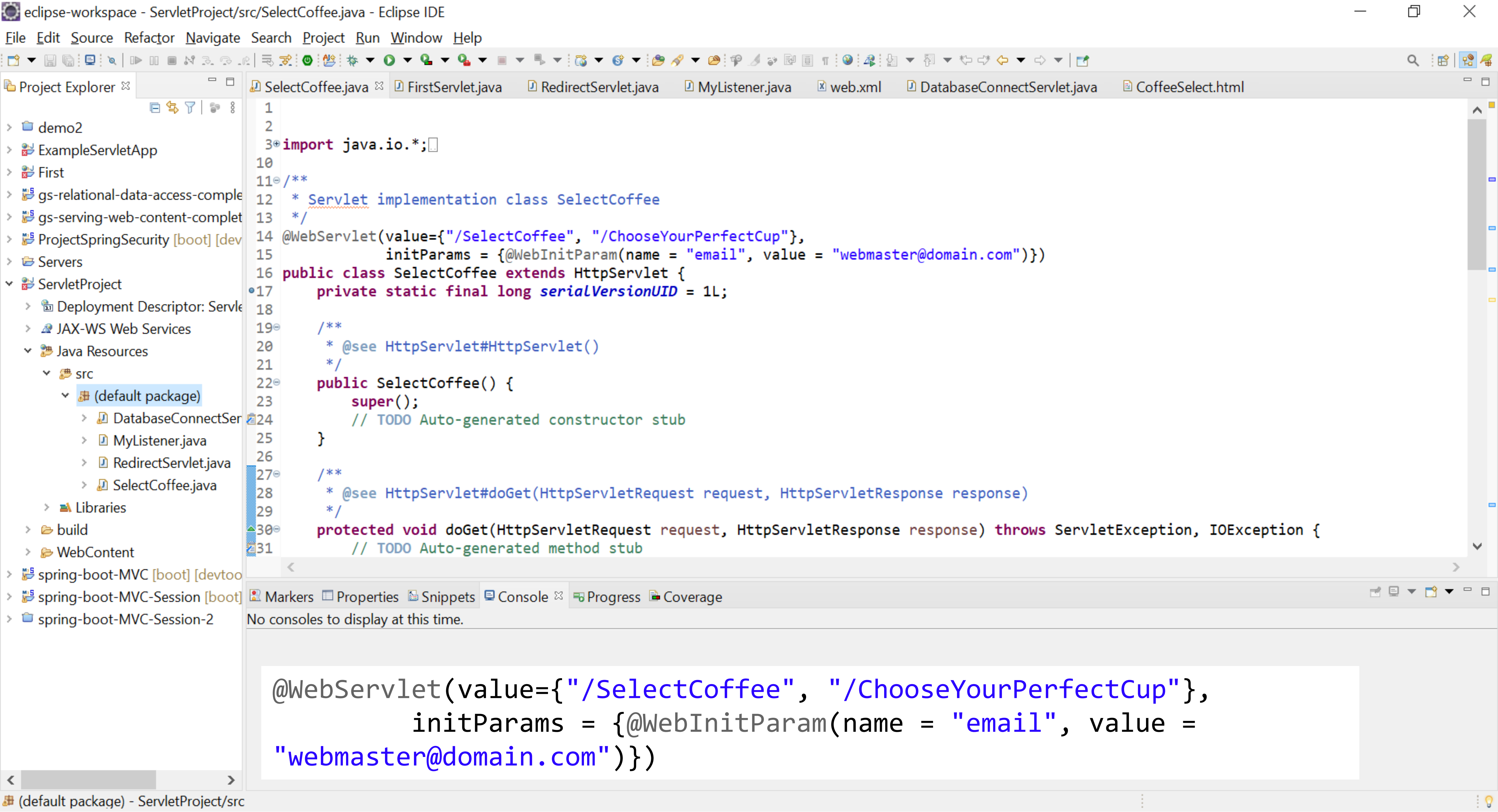
SERVLETS AND JSP



```
import java.io.*;
ectCup"},
", value = "webmaster@domain.com")})
est, HttpServletResponse response)
HttpServletResponse response) throws ServletException, IOException {
method stub
```

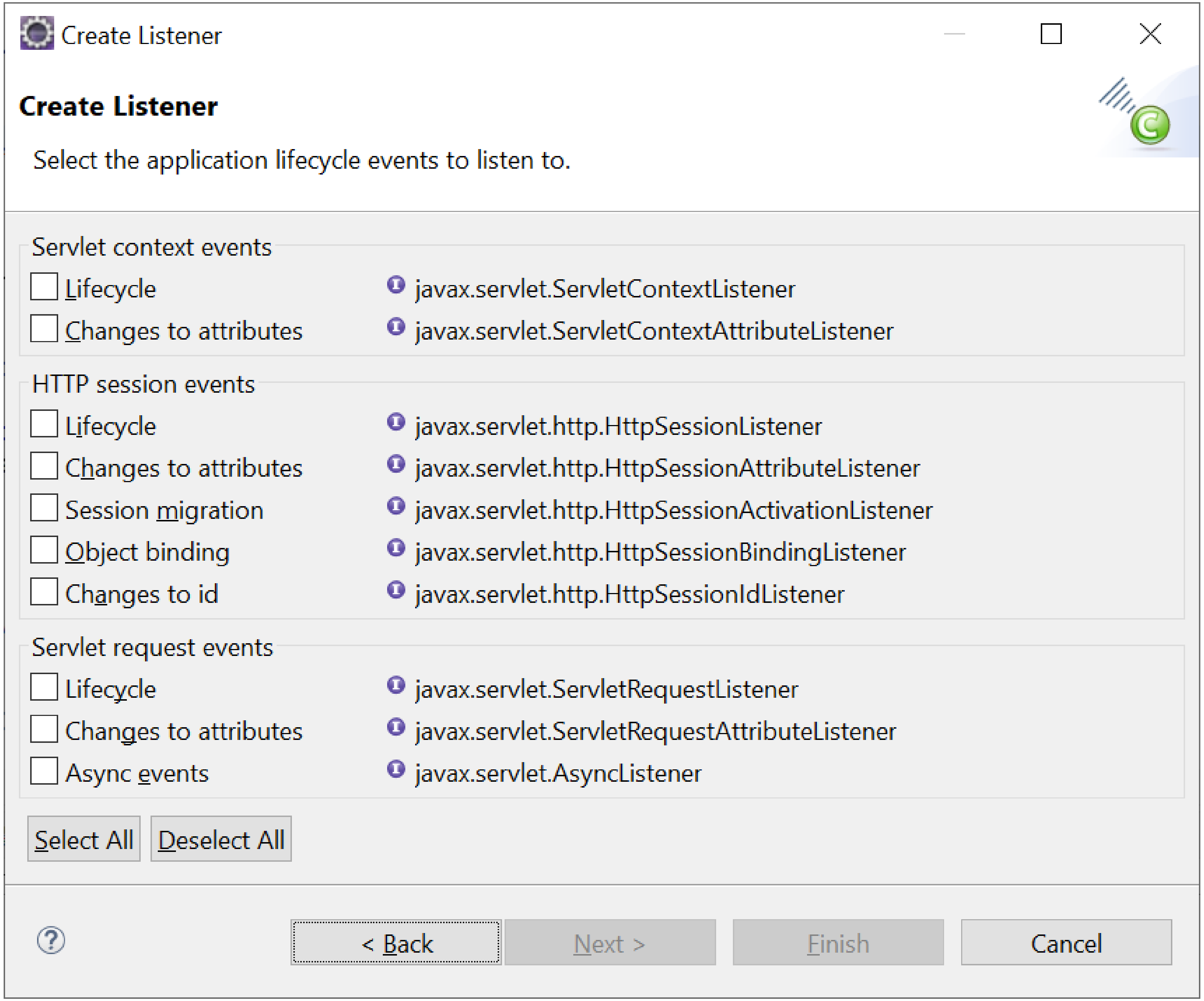

eclipse-workspace - ServletProject/src/SelectCoffee.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
demo2
ExampleServletApp
First
gs-relational-data-access-comple
gs-serving-web-content-complet
ProjectSpringSecurity [boot] [dev
Servers
ServletProject
Deployment Descriptor: Servle
JAX-WS Web Services
Java Resources
src
(default package)
DatabaseConnectSer
MyListener.java
RedirectServlet.java
SelectCoffee.java
Libraries
build
WebContent
spring-boot-MVC [boot] [devtoo
spring-boot-MVC-Session [boot]
spring-boot-MVC-Session-2
SelectCoffee.java
FirstServlet.java
RedirectServlet.java
MyListener.java
web.xml
DatabaseConnectServlet.java
CoffeeSelect.html
Markers
Properties
Snippets
Console
Progress
Coverage
(default package) - ServletProject/src

```


import java.io.*;

/**
 * Servlet implementation class SelectCoffee
 */
@WebServlet(value={"/SelectCoffee", "/ChooseYourPerfectCup"},
            initParams = {@WebInitParam(name = "email", value = "webmaster@domain.com")})
public class SelectCoffee extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public SelectCoffee() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
```

No consoles to display at this time.

```
@WebServlet(value={"/SelectCoffee", "/ChooseYourPerfectCup"},
            initParams = {@WebInitParam(name = "email", value =
"webmaster@domain.com")})
```

LISTENER



Create Listener

Create Listener

Select the application lifecycle events to listen to.

Servlet context events

□ Lifecycle — javax.servlet.ServletContextListener

□ Changes to attributes — javax.servlet.ServletContextAttributeListener

HTTP session events

□ Lifecycle — javax.servlet.http.HttpSessionListener

□ Changes to attributes — javax.servlet.http.HttpSessionAttributeListener

□ Session migration — javax.servlet.http.HttpSessionActivationListener

□ Object binding — javax.servlet.http.HttpSessionBindingListener

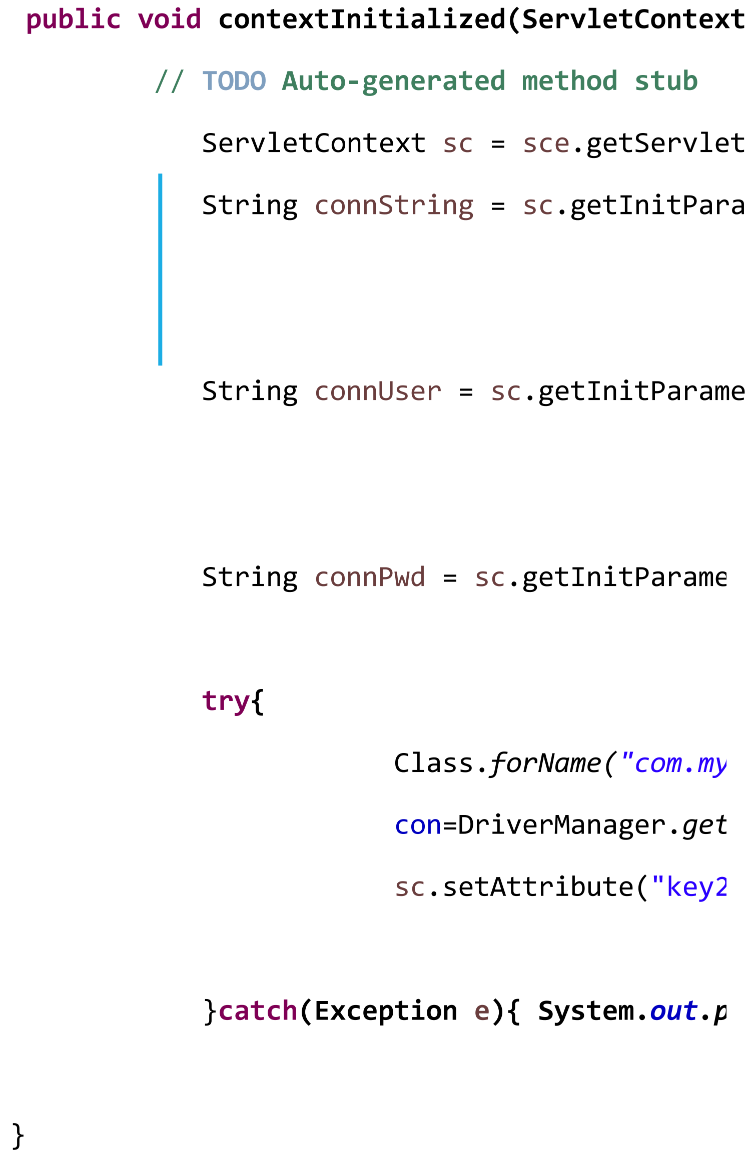
□ Changes to id — javax.servlet.http.HttpSessionIdListener

Servlet request events

□ Lifecycle — javax.servlet.ServletRequestListener

□ Changes to attributes — javax.servlet.ServletRequestAttributeListener

□ Async events — javax.servlet.AsyncListener



```
public void contextInitialized(ServletContextEvent sce)  {
        // TODO Auto-generated method stub
          ServletContext sc = sce.getServletContext();
          String connString = sc.getInitParameter("key1");

          String connUser = sc.getInitParameter("key1User");

          String connPwd = sc.getInitParame

          try{
                  Class.forName("com.my
                  con=DriverManager.get
                  sc.setAttribute("key2

          }catch(Exception e){ System.out.p
}
```

```
String c = request.getParameter("color");
try{
Connection con=(Connection)getServletContext().getAttribute("key2");
        Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select brand from CoffeeTypes where color
= '" + c + "'");
String result="";
while(rs.next())
result=result+rs.getString(1) ;
if(!result.equals("")) {
out.println…;
}
else
   out.println…;

}catch(Exception e){ System.out.println(e);}
```

```java
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet({ "/FirstServlet", "/Hello" })
public class FirstServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		// TODO Auto-generated method stub
		response.setContentType("text/html");
		response.getWriter().append("<br>Hello World");
	}

}
```

```jsp
<%@ page language="java"
contentType="text/html"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%="Hello World!" %>

</body>
</html>
```

A CONVERTED SERVLET

```
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class index_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

    public void _jspService(HttpServletRequest request, HttpServletResponse response)  throws java.io.IOException, ServletException {

      out.write("<title>A JSP Example</title>\r\n");
      out.write("</head>\r\n");
     out.write("Example</font></h1>\r\n");
     out.write("<p align=\"Center\">");
      out.print(new java.util.Date());
      out.write("&nbsp;</p>\r\n");
      out.write("</body>\r\n");
      out.write("</html>");
    }
}
```

JSP SCRIPTING ELEMENTS

There is more than one type of JSP “tag,” depending on what you want done with the Java

<%= **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
- This construction is called a scriptlet

<%! ***declarations*** %>

- The ***declarations*** are inserted into the servlet *class,* not into a method

OTHER JSP ELEMENTS

<%@ directive%>

- The ***directive*** is a way to give special instructions to the container at page translation time

<%--comment --%>

- The ***comment*** is just like java comments

DIRECTIVES

The container looks for directives for information it might need for translation

A directive has the form:

```
    <%@ directive attribute="value" %>
or
    <%@ directive attribute1="value1,value1.1"
                  attribute2="value2"
                  ...
                  attributeN="valueN" %>
```

The most useful directive is page, which lets you import packages

- Example: `<%@ page import="java.util.*" %>`

Three flavors of directives: page, include, taglib

<%@ DIRECTIVE %>

```
<%@ page import=“java.util.*”%>
<html>
  <head>
    <title>JSP is Easy</title>
  </head>
  <body bgcolor="white">
    <h1>JSP is as easy as ...</h1>
    <%-- Calculate the sum of 1 + 2 + 3 dynamically --%>
    1 + 2 + 3 = <c:out value="${1 + 2 + 3}" />
    <%=new Date()%>
  </body>
</html>
```

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, or null if there is none
- out : A JspWriter (like a PrintWriter) used to send output to the client

Example:

- Your hostname: <%= request.getRemoteHost() %>

IMPLICIT OBJECTS

API	Implicit Object
JspWriter	Out
HttpServletRequest	Request
HttpServletResponse	Response
HttpSession	Session
ServletContext	Application
ServletConfig	Config
Throwable	Exception
PageContext	pageContext
Object	page

PAGECONTEXT

PageContext encapsulates other implicit objects, so if any helper object gets this ref. then it can easily get other implicit objects and attributes from all scopes

Keeps track of attributes at four levels:

- Application
- Session
- Request
- Page

javax.servlet.jsp.PageContext pageContext

- getAttribute(name, scope)
- removeAttribute(name, scope)
- setAttribute(name, value, scope)
- findAttribute(name)
    - Finds in page scope → request scope → session scope → application scope

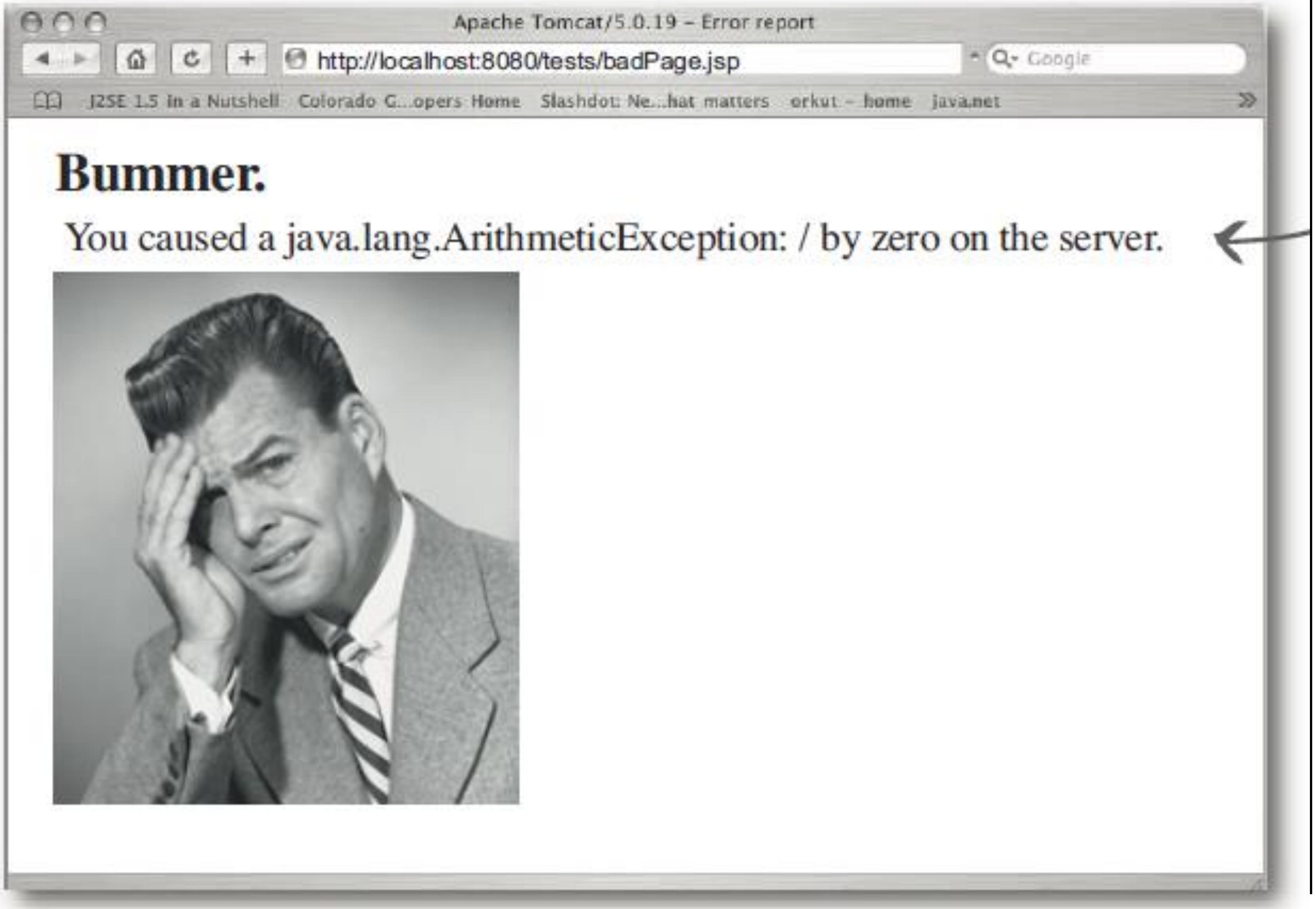
CLOSURE LOOK AT PAGE

Page

- <%@ page errorPage="Relative URL" %>
  - The exception thrown will be automatically available to the designated error page by means of the "exception“ variable
  - The web.xml file lets you specify *application-wide* error pages that apply whenever certain exceptions or certain HTTP status codes result
- <%@ page isErrorPage="false" %> <%-- Default --%>
  - Indicates whether or not the current page can act as the error page for another JSP page

<html>

errorPage.jsp

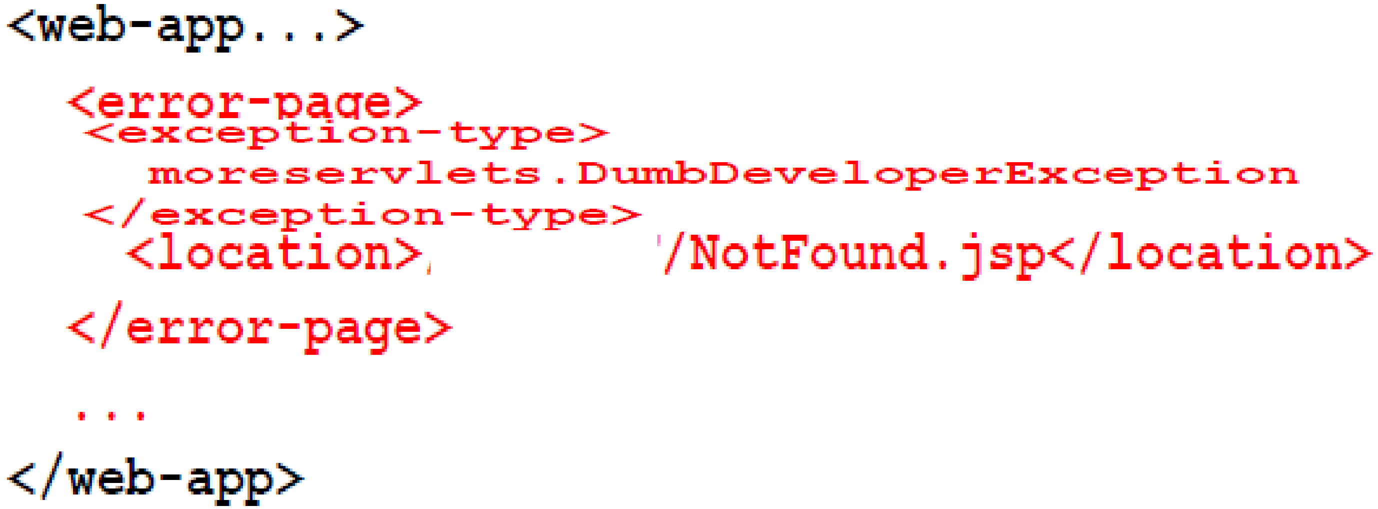


Apache Tomcat/5.0.19 – Error report
http://localhost:8080/tests/badPage.jsp
Google
J2SE 1.5 in a Nutshell
Colorado G...opers Home
Slashdot: Ne...hat matters
orkut – home
java.net

Bummer.

You caused a java.lang.ArithmeticException: / by zero on the server.

CONFIGURING ERROR PAGE IN THE DEPLOYMENT DESCRIPTOR



```
<web-app...>

  <error-page>
   <exception-type>
     moreservlets.DumbDeveloperException
   </exception-type>
    <location>          /NotFound.jsp</location>

  </error-page>

  ...

</web-app>
```

Pages to use for specific HTTP status codes

- Use the error-code element within error-page

Pages to use when specific uncaught exceptions are thrown

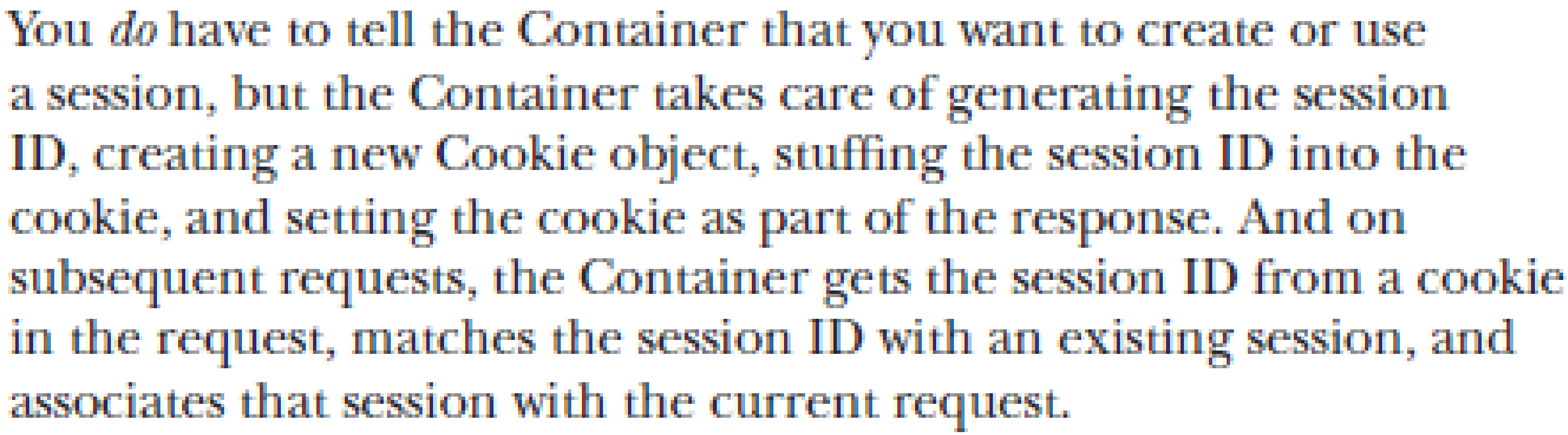
- Use the exception-type element within error-page

SCRIPTING IS HARMFUL

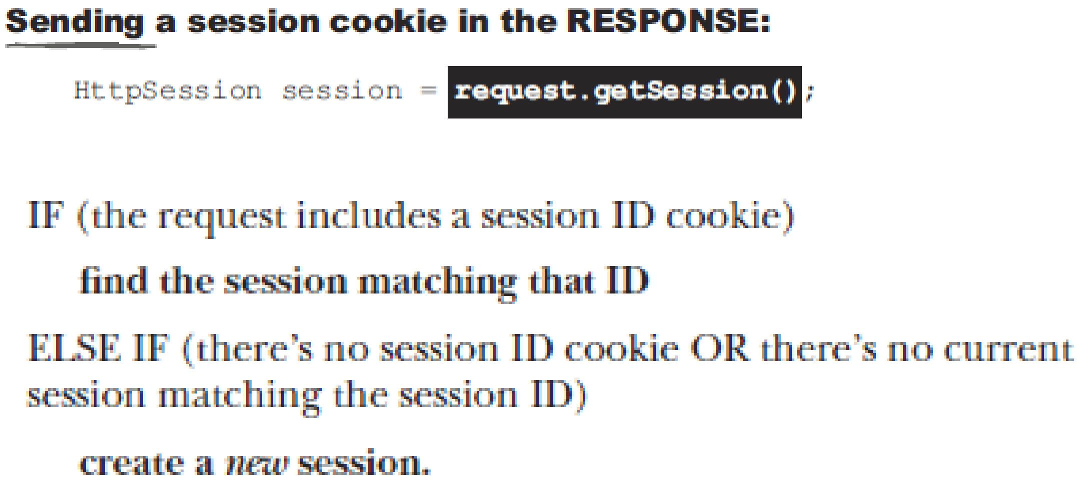
- Reasons
  - Web page designers need not know java
  - Java code in a jsp is hard to change and maintain
- Solution
  - Expression Language
  - Standard action <jsp:include…/>
  - Custom action <c:set…/>

SESSION HANDLING

ROLE OF THE CONTAINER



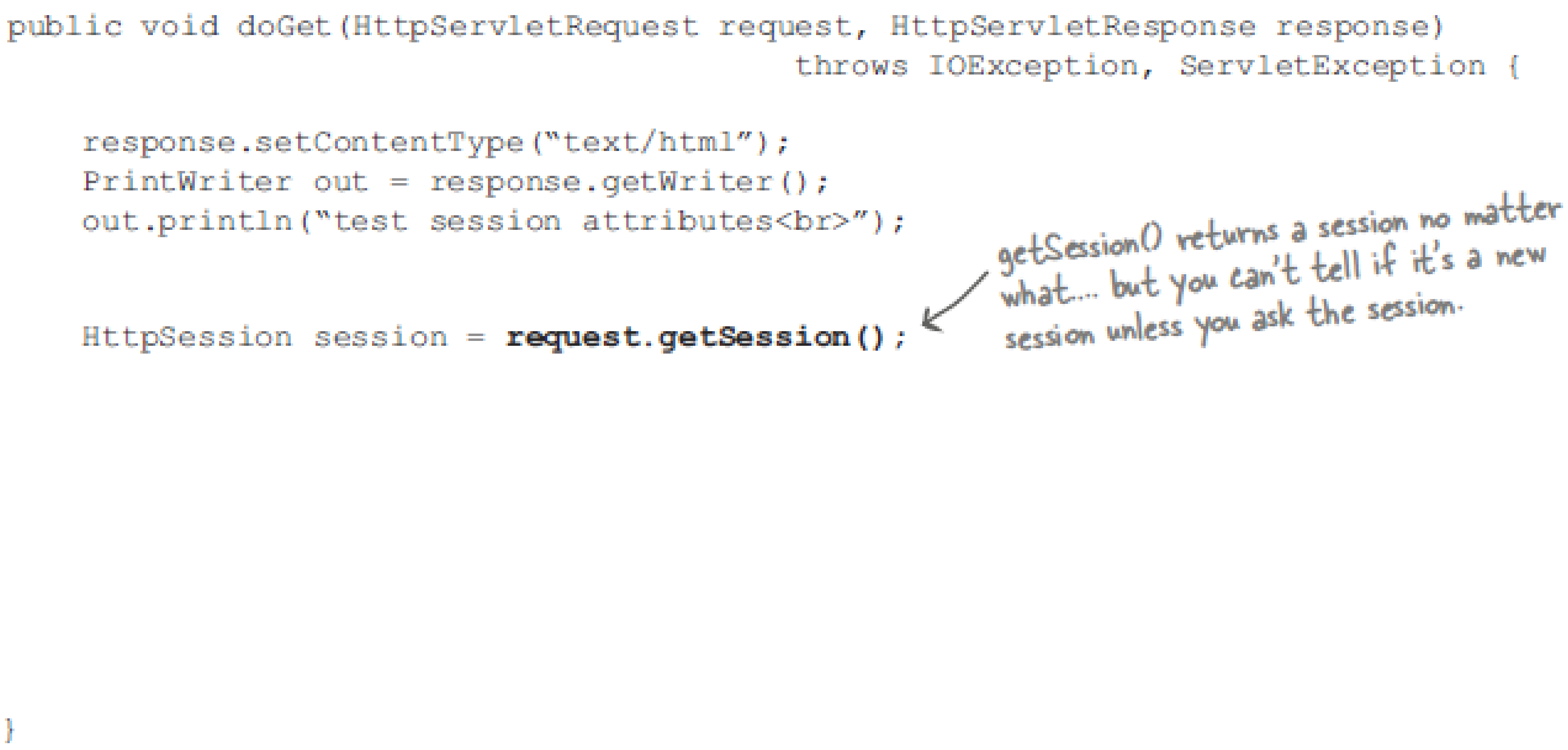
You *do* have to tell the Container that you want to create or use a session, but the Container takes care of generating the session ID, creating a new Cookie object, stuffing the session ID into the cookie, and setting the cookie as part of the response. And on subsequent requests, the Container gets the session ID from a cookie in the request, matches the session ID with an existing session, and associates that session with the current request.



Sending a session cookie in the RESPONSE:

```
HttpSession session = request.getSession();
```

```
IF (the request includes a session ID cookie)
    find the session matching that ID
ELSE IF (there's no session ID cookie OR there's no current
session matching the session ID)
    create a new session.
```



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

    response.setContentType(“text/html”);
    PrintWriter out = response.getWriter();
    out.println(“test session attributes<br>”);


    HttpSession session = request.getSession();







}
```

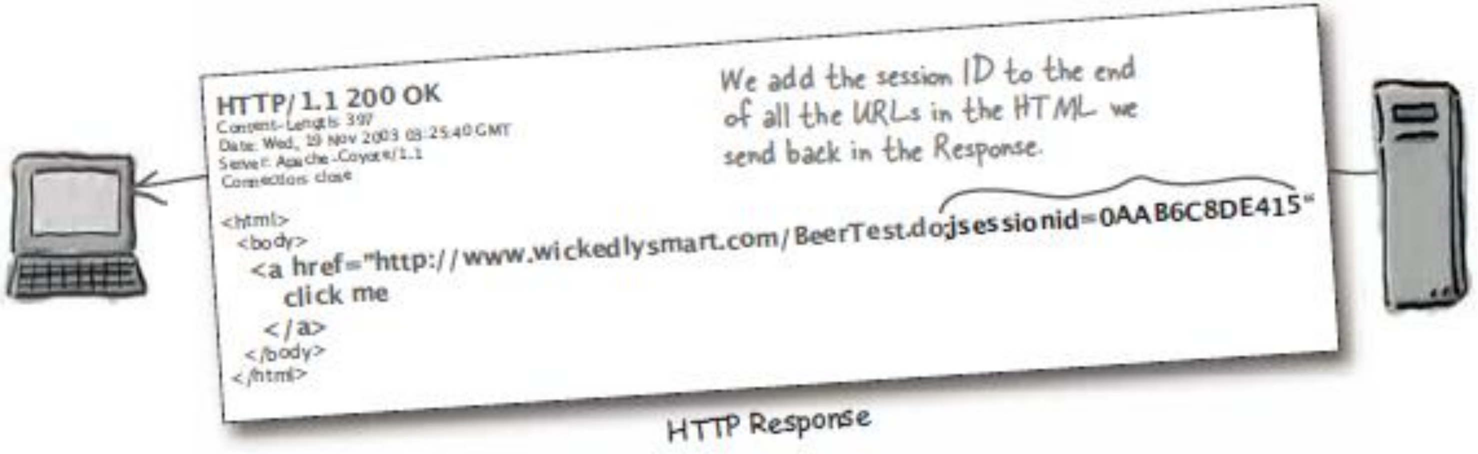
getSession() returns a session no matter what... but you can't tell if it's a new session unless you ask the session.

Request.getSession(false);---if an existing session is not found it returns null

URL REWRITING

response.encodeURL("…")

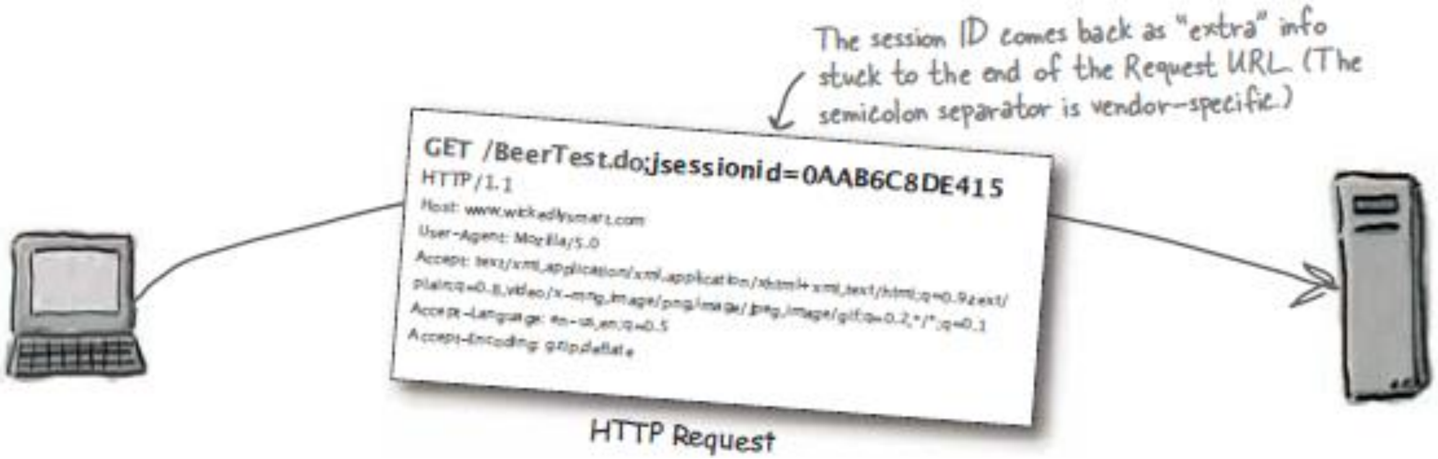
Response.encodeRedirectURL("   ")



```
HTTP/1.1 200 OK
Content-Length: 397
Date: Wed, 19 Nov 2003 03:25:40 GMT
Server: Apache-Coyote/1.1
Connection: close

<html>
  <body>
    <a href="http://www.wickedlysmart.com/BeerTest.do;jsessionid=0AAB6C8DE415"
      click me
    </a>
  </body>
</html>
```

We add the session ID to the end of all the URLs in the HTML we send back in the Response.

HTTP Response



```
GET /BeerTest.do;jsessionid=0AAB6C8DE415
HTTP/1.1
Host: www.wickedlysmart.com
User-Agent: Mozilla/5.0
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
```

The session ID comes back as "extra" info stuck to the end of the Request URL. (The semicolon separator is vendor-specific.)

HTTP Request

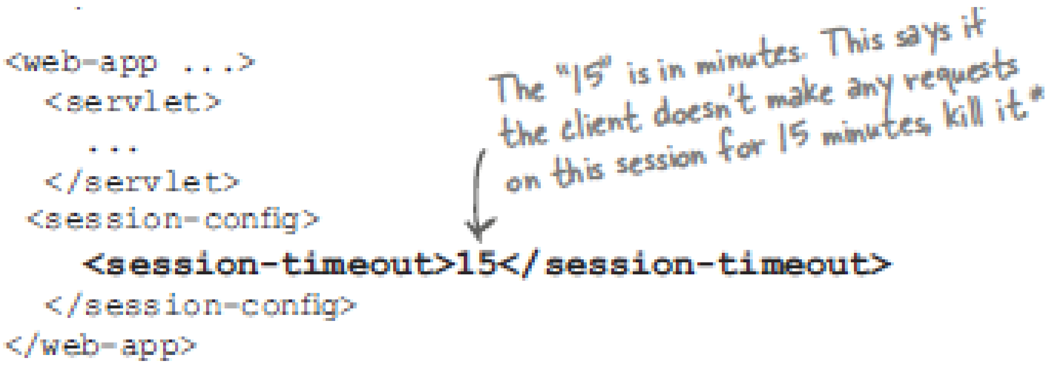
TERMINATING SESSIONS

Sessions are mostly used to get and set session scoped attributes

Ways to terminate a session

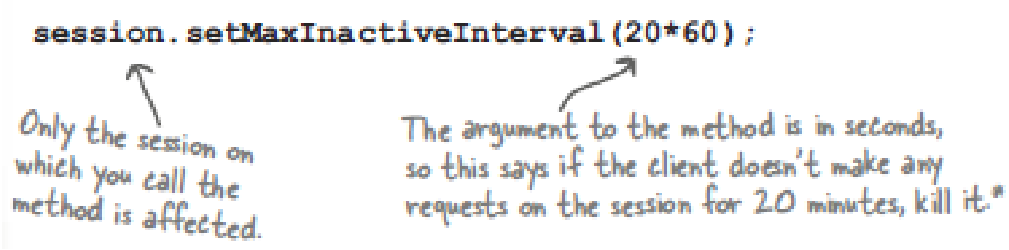
- It times out
- Calling session.invalidate()
- The application goes down (crashes or is undeployed)

SESSION TIMES OUT



```
<web-app ...>
  <servlet>
    ...
  </servlet>
 <session-config>
    <session-timeout>15</session-timeout>
  </session-config>
</web-app>
```

The "15" is in minutes. This says if the client doesn't make any requests on this session for 15 minutes, kill it*

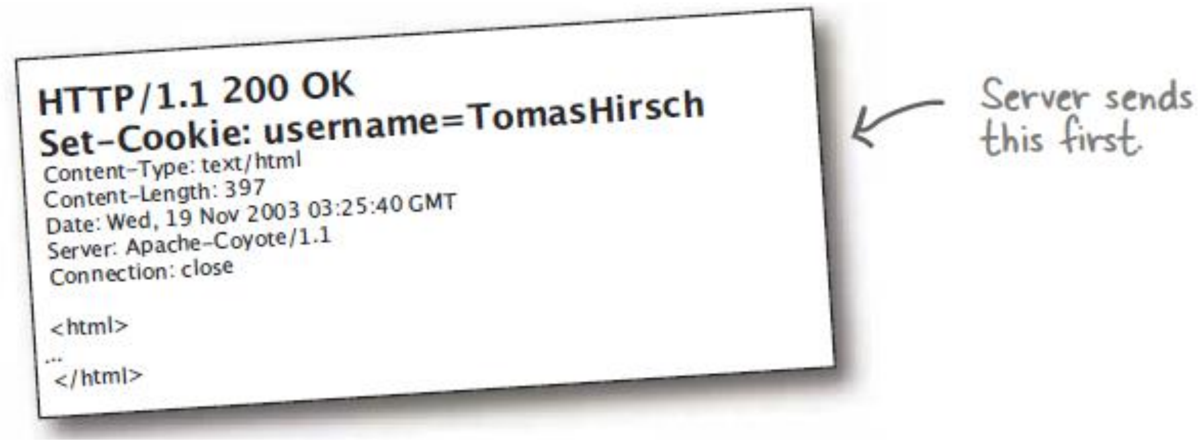


```
session.setMaxInactiveInterval(20*60);
```

Only the session on which you call the method is affected.

The argument to the method is in seconds, so this says if the client doesn't make any requests on the session for 20 minutes, kill it*

COOKIES

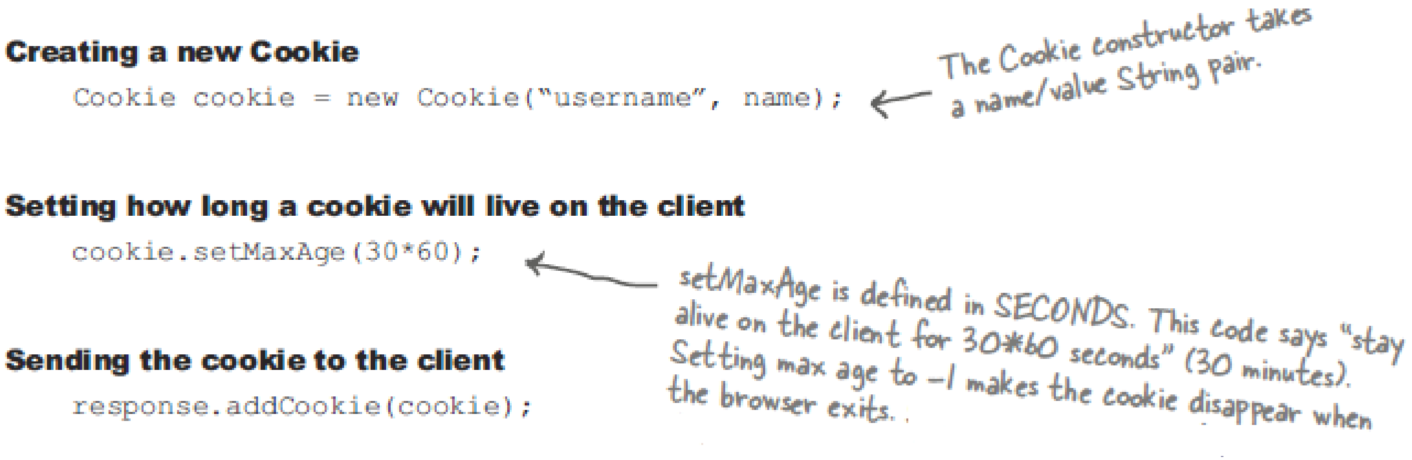


```
HTTP/1.1 200 OK
Set-Cookie: username=TomasHirsch
Content-Type: text/html
Content-Length: 397
Date: Wed, 19 Nov 2003 03:25:40 GMT
Server: Apache-Coyote/1.1
Connection: close

<html>
...
</html>
```

Server sends this first.

Session cookies vanish when the client's browser quits, but you CAN tell a cookie to persist on the client even after the browser shuts down.

ADDING AND GETTING COOKIES



Creating a new Cookie

```
    Cookie cookie = new Cookie(“username”, name);
```

The Cookie constructor takes a name/value String pair.

Setting how long a cookie will live on the client

```
    cookie.setMaxAge(30*60);
```

setMaxAge is defined in SECONDS. This code says “stay alive on the client for 30*60 seconds” (30 minutes). Setting max age to -1 makes the cookie disappear when the browser exits.

Sending the cookie to the client

```
    response.addCookie(cookie);
```

ATTRIBUTES

Scope	In a servlet	In a jsp Implicit obj
Application	getServletContext.setAttribute(“name”,obj)	application.setAttribute(“name”, obj)
Request	request. .setAttribute(“name”,obj)	request. .setAttribute(“name”,obj)

Session	Request.getSession()
	.setAttribute(“name”,obj)
		session.setAttribute(“name”,obj)

Page	NA	pageContext.setAttribute(“name”,obj)