

B.Sc. In Software Development. Year 4.  
Semester I. Enterprise Development.  
Sessions and Cookies.



**LIMERICK INSTITUTE  
OF TECHNOLOGY**  
**SCHOOL OF SCIENCE,  
ENGINEERING & I.T.**

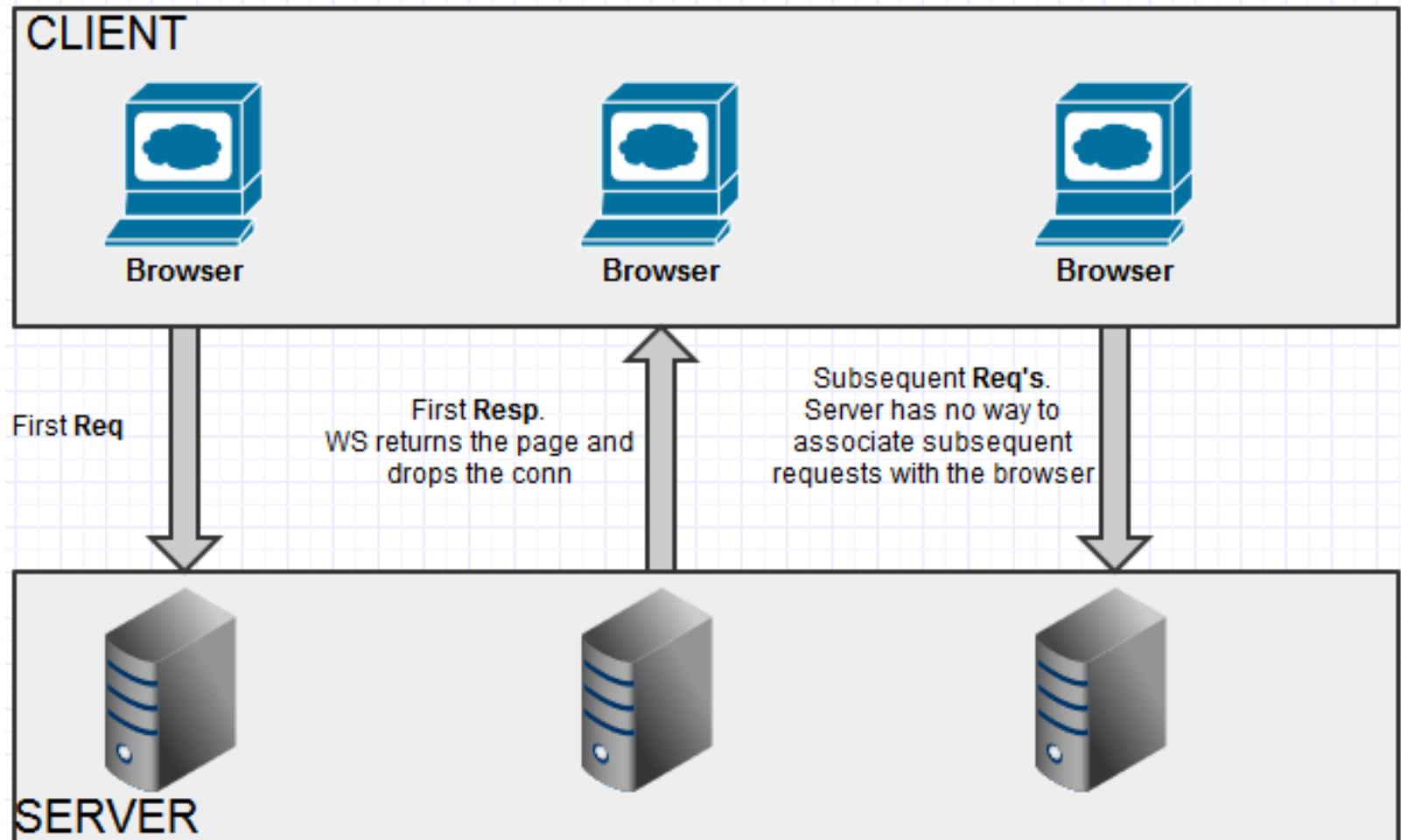
*Department of Information Technology*

# Introduction

- In all but the simplest of web applications you will need to track a user as they move through your site.
- Using the Servlet API tracking users using either Sessions or Cookies.
- Session tracking in Web Applications is more difficult than most other types of applications.
- You have two choices.
  1. Use a session.
  2. Use a cookie.

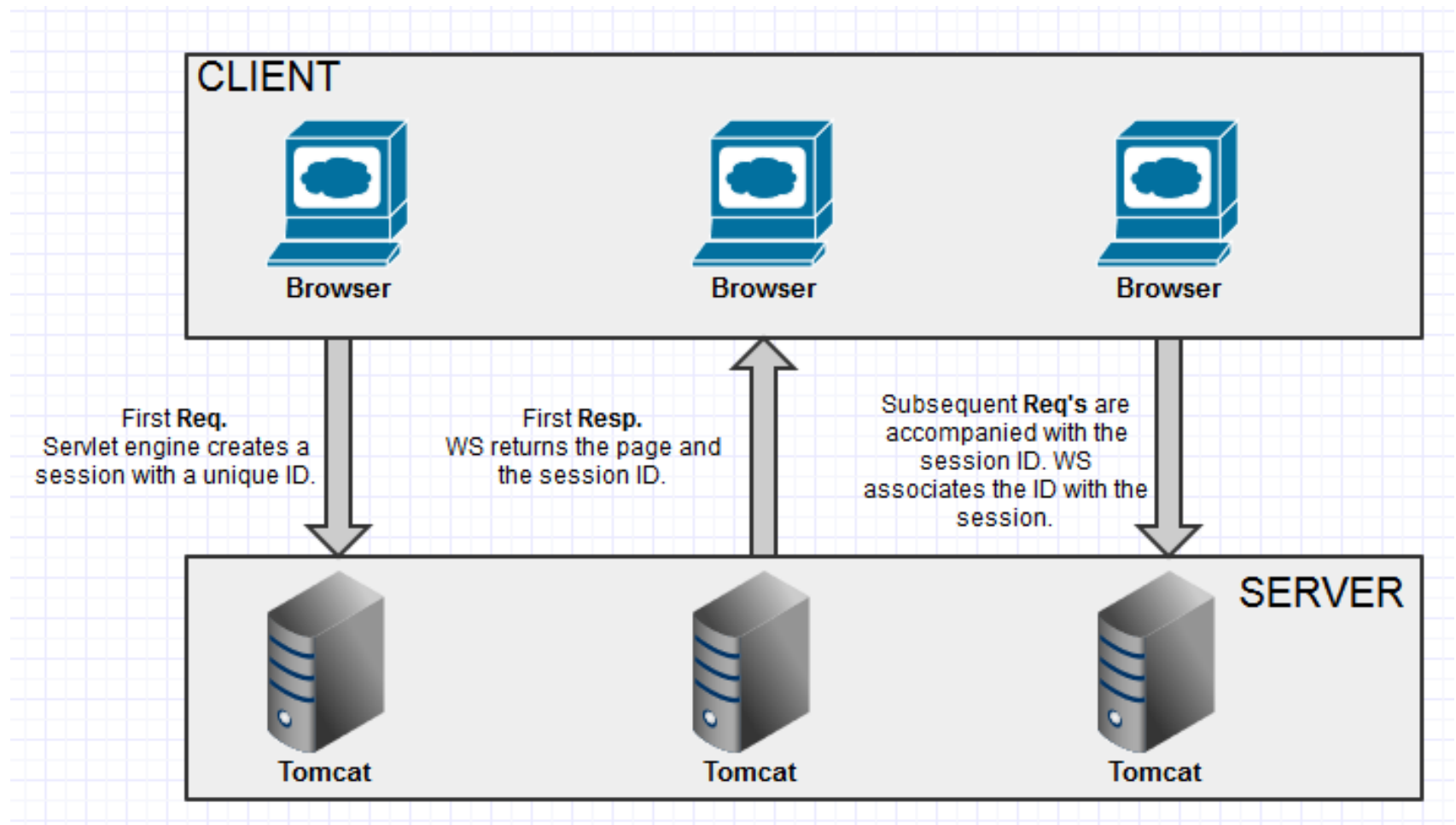
# Introduction

## Without Session Management



# Introduction

## Using Session Management



# How It Works

- To start, a browser requests a JSP/servlet from the server, which passes the request to the servlet engine.
- The engine checks if the request includes an ID for the Java session.
- If it doesn't, the servlet engine creates a unique ID for the session plus a session object that can be used to store data for the session.
- From that point onwards, the server uses the session ID (stored as a cookie) to relate each browser request to the session object. Even though the server still drops the HTTP connection after returning each page.

## How It Works

- When the next request is made, this cookie is added to the request.
- What if cookies have been disabled?
  - You will have to rewrite the URL so that it includes the session ID.
  - URL encoding.
- Once session handling has been implemented you can implement a wide variety of tasks.

# Sessions

- The data within a session is stored on the server so they are more secure than a cookie.
- The session persists for a specified period of time across more than one connection or page request from the user.
- Usually corresponds to one user who may visit a site many times.
- The session enables tracking of a large set of data.
- Typically, in the Servlet, you will either need to create a session or get your hands on the current session. The code to do so is as follows:

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

# Sessions

**Useful method of the [HttpServletRequest](#) object**

Return Type	Method Name	Description
HttpSession	getSession()	Returns the current session associated with this request, or if the request does not have a session, creates one.



# Sessions

## Useful methods of the session object

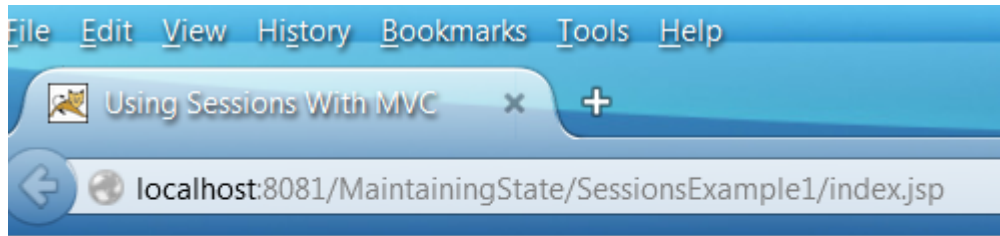
Return Type	Method Name	Description
Object	getAttribute(String name)	Returns the object bound with the specified name in this session, or null if no object is bound under the name.
void	setAttribute(String name, Object value)	Binds an object to this session, using the name specified.
String	getId()	Returns a string containing the unique identifier assigned to this session.

# Sessions

## Useful method of the session object

Return Type	Method Name	Description
void	<code>removeAttribute(String name)</code>	Removes the object bound with the specified name from this session.
void	<code>invalidate()</code>	Invalidates this session then unbinds any objects bound to it.
boolean	<code>isNew()</code>	Returns true if the client does not yet know about the session or if the client chooses not to join the session.
void	<code>setMaxInactiveInterval(int interval)</code>	Specifies the time, in seconds, between client requests before the servlet container will invalidate this session.

# Sessions - Example1 (putting simple values into session)



[Click Here](#)



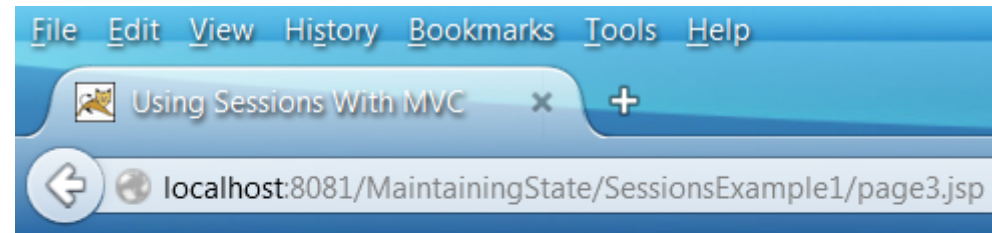
My Three wise men are....

Alan

Tom

Brendan

[Click Here](#)



My Four wise men are....

Alan

Tom

Brendan

Gerry

# Sessions - Example1 (putting simple values into session)

## Partial code-listing for index.jsp

```
8  <body>
9      <a href="../SessionServlet"> Click Here </a>
10
11
12  </body>
```

## Partial code-listing for SessionServlet

```
23  HttpSession sess = request.getSession();
24
25  sess.setAttribute("WiseMan1", "Alan");
26  sess.setAttribute("WiseMan2", "Tom");
27  sess.setAttribute("WiseMan3", "Brendan");
28
29  RequestDispatcher dispatcher = request.getRequestDispatcher("/SessionsExample1/page2.jsp");
30  dispatcher.forward(request, response);
```

# Sessions - Example1 (putting simple values into session)

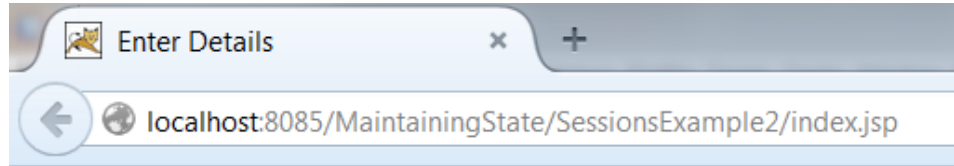
## Partial code-listing for page2.jsp

```
11      My Three wise men are....
12
13      <br> ${sessionScope.WiseMan1} <br> ${sessionScope.WiseMan2} <br> ${sessionScope.WiseMan3} <br>
14
15
16      <c:set var="WiseMan4" value="Gerry" scope="session" />
17
18
19      <a href="SessionsExample1/page3.jsp"> Click Here </a>
```

## Partial code-listing for page3.jsp

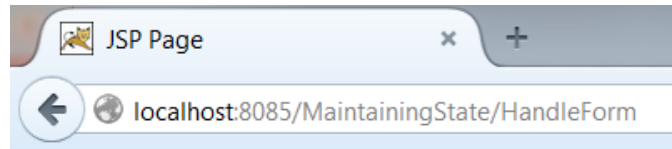
```
10      My Four wise men are....
11
12      <br> ${sessionScope.WiseMan1} <br> ${sessionScope.WiseMan2} <br> ${sessionScope.WiseMan3} <br> ${sessionScope.WiseMan4}
13
```

# Sessions - Example2 (putting objects into session)



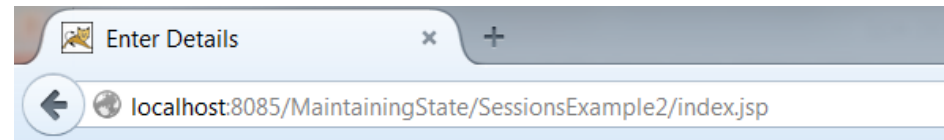
A screenshot of a web browser window. The title bar shows 'Enter Details'. The address bar shows 'localhost:8085/MaintainingState/SessionsExample2/index.jsp'. The page contains three text input fields with labels 'Enter First Name', 'Enter Last Name', and 'Enter Email Address'. The first field contains 'Dom', the second contains 'Jones', and the third contains 'dj@yahoo.ie'. Below the fields is a 'Submit' button.

Enter First Name Dom  
Enter Last Name Jones  
Enter Email Address dj@yahoo.ie



A screenshot of a web browser window. The title bar shows 'JSP Page'. The address bar shows 'localhost:8085/MaintainingState/HandleForm'. The page displays the submitted details and a link to edit them.

Submitted Details are:  
First Name: Dom  
Last Name: Jones  
Email Address: dj@yahoo.ie  
[Edit These Details](#)



A screenshot of a web browser window. The title bar shows 'Enter Details'. The address bar shows 'localhost:8085/MaintainingState/SessionsExample2/index.jsp'. The page contains three text input fields with labels 'Enter First Name', 'Enter Last Name', and 'Enter Email Address'. The first field contains 'Dom', the second contains 'Jones', and the third contains 'dj@yahoo.ie'. Below the fields is a 'Submit' button.

Enter First Name Dom  
Enter Last Name Jones  
Enter Email Address dj@yahoo.ie

# Sessions - Example2 (putting objects into session)

## Partial code-listing for index.jsp

```
<form action="../HandleForm" method="POST">

    Enter First Name <input type="text" name="firstName" value="${sessionScope.aUser.firstName}" /> <br>
    Enter Last Name <input type="text" name="lastName" value="${sessionScope.aUser.lastName}" /> <br>
    Enter Email Address<input type="text" name="emailAddress" value="${sessionScope.aUser.emailAddress}" />
    <input type="submit" value="Submit" />
</form>
```

## Partial code-listing for HandleForm (a Servlet)

```
22 String fname = request.getParameter("firstName");
23 String lname = request.getParameter("lastName");
24 String email = request.getParameter("emailAddress");
25
26 User user = new User(fname, lname, email);
27
28 HttpSession sess = request.getSession();
29
30 sess.setAttribute("aUser", user);
31
32 RequestDispatcher dispatcher = request.getRequestDispatcher("SessionsExample2/page2.jsp");
33
34 dispatcher.forward(request, response);
```

# Sessions - Example2 (putting objects into session)

## Partial code-listing for page2.jsp

```
<body>
    Submitted Details are: <br>
    First Name: ${sessionScope.aUser.firstName} <br>
    Last Name: ${sessionScope.aUser.lastName} <br>
    Email Address: ${sessionScope.aUser.emailAddress} <br>

    <a href="SessionsExample2/index.jsp">Edit These Details</a>

</body>
```



# Sessions

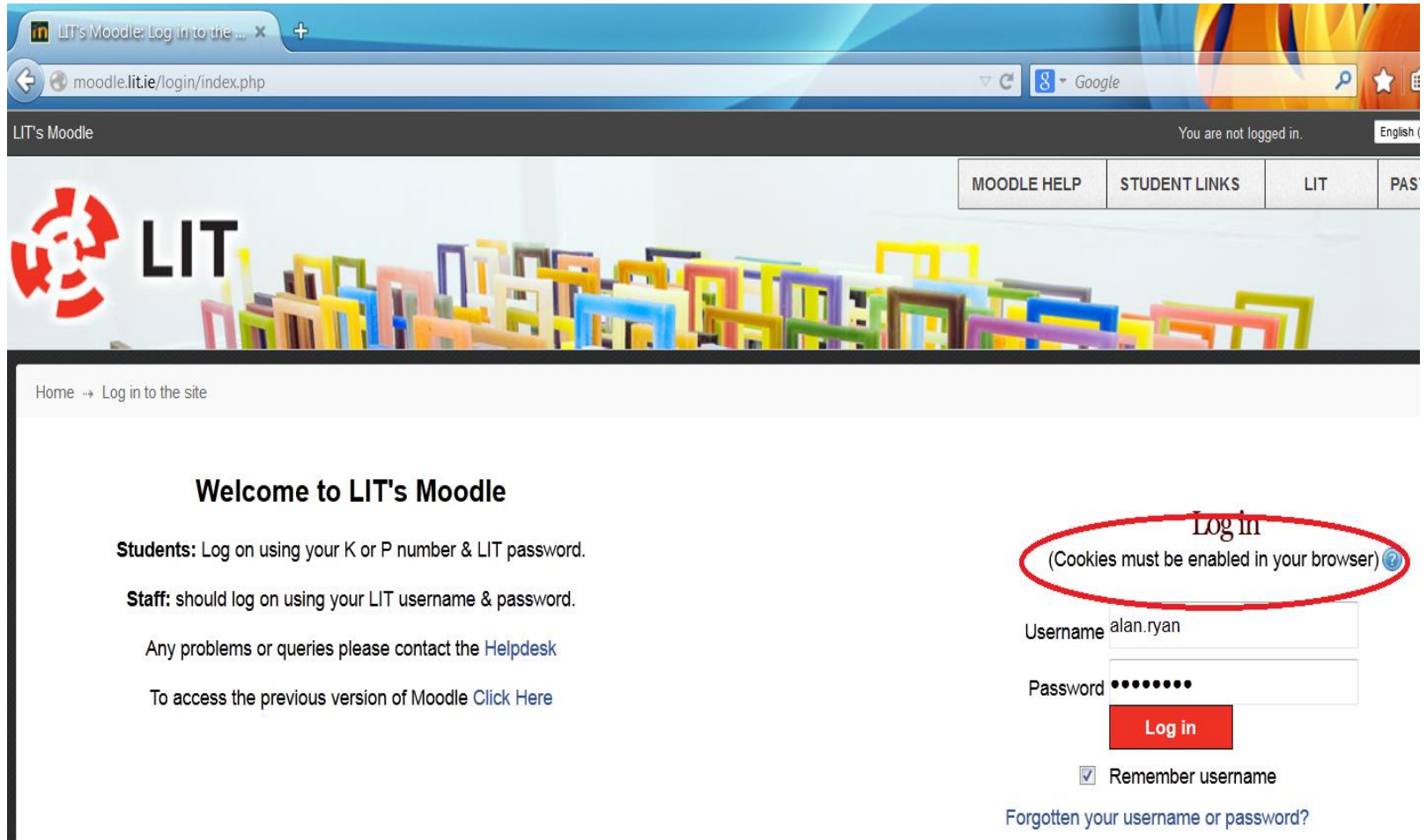
- Examples of some of these methods are as follows:

```
36 //use getAttributeNames() to return a enumeration object of all
37 //the names of all attributes stored in the session. Then use
38 //hasMoreElements() and nextElement() to loop through the names
39 //This is useful for debugging
40 Enumeration names = sess.getAttributeNames();
41 while (names.hasMoreElements()) {
42     System.out.println((String)names.nextElement());
43 }
44
45
46 //retrieve the sessions unique id
47 //useful for debugging
48 String id = sess.getId();
49
50 //specify length of session
51 sess.setMaxInactiveInterval(60*60*24); //one day
52 sess.setMaxInactiveInterval(-1); //until the browser is closed
53
54 //invalidates the session and unbinds any objects to it
55 sess.invalidate();
56
57 //check if the client is new or if the client chooses not to join the session
58 //returns true if the client is accessing the site for the first time or if
59 //cookies have been disabled on the browser
60 sess.isNew();
```

# Enabling or disabling cookies

- Two types of cookies:
  - A per-session cookie: stored on the browser until the user closes the browser.
  - A persistent cookie: can be stored on the users computer for up to three years.
- To test sessions that rely on cookies, you'll need to enable cookies in your browser.
- Conversely, to test code that's intended to work even if cookies have been disabled, you'll need to disable cookies in your browser.
- Although cookies are enabled by default on most browsers, some people choose to turn them off.
- Programmers must take this into consideration.
- Some sites take the approach of telling users that the site won't work properly if cookies are disabled.

# Enabling or disabling cookies



LIT's Moodle: Log in to the ... x

moodle.lit.ie/login/index.php

LIT's Moodle

You are not logged in.

MOODLE HELP STUDENT LINKS LIT PAS

Home → Log in to the site

## Welcome to LIT's Moodle

**Students:** Log on using your K or P number & LIT password.

**Staff:** should log on using your LIT username & password.

Any problems or queries please contact the [Helpdesk](#)

To access the previous version of Moodle [Click Here](#)

**Log in**  
(Cookies must be enabled in your browser) ?

Username alan.ryan

Password .....

**Log in**

☒ Remember username

[Forgotten your username or password?](#)

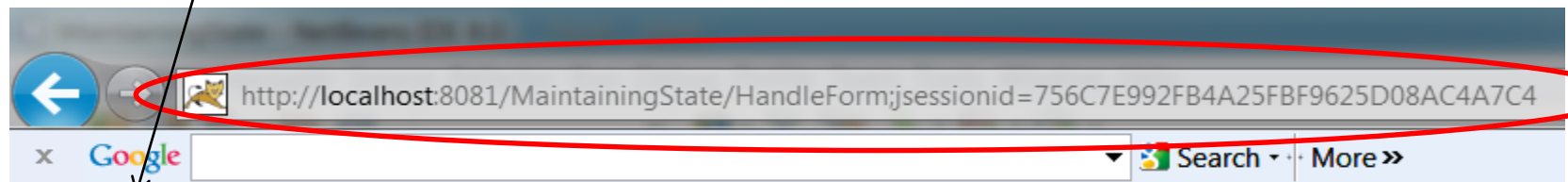
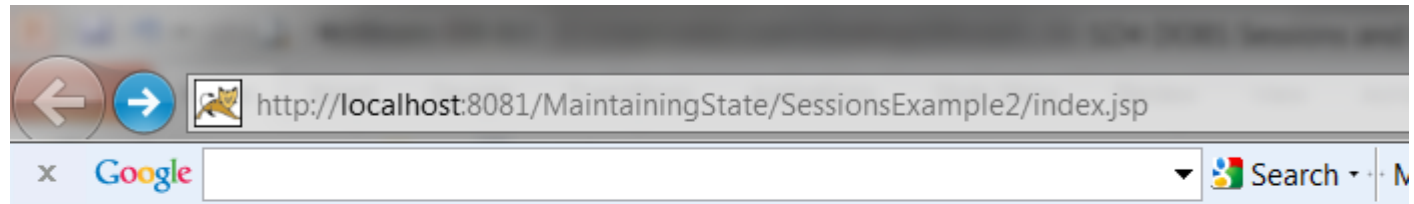
# Enabling or disabling cookies

- If this option is not acceptable to you then you'll have to use URL encoding.
- Once URL's have been encoded, session ID's are then added to the end of a URL whenever the URL is requested from a browser with disabled cookies.
- When using URL encoding, you must be sure to encode all URL's in the application.
- Use the `encodeURL()` method of the `HttpServletRequest` interface. For example in a form:

```
<form action="<%= response.encodeURL("../HandleForm") %>" method="POST">
```

- Links have to be encoded too in a similar fashion.

# Enabling or disabling cookies



Submitted Details are:  
First Name: Alan  
Last Name: Ryan  
Email Address: alan.ryan@lit.ie  
[Edit These Details](#)

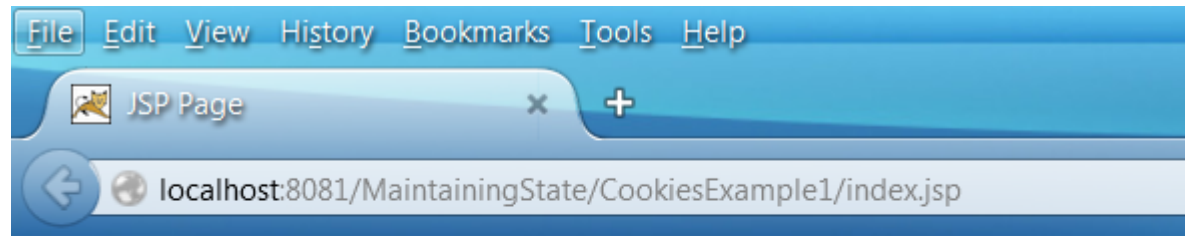
# Cookies

- Once you create a cookie, you include it in the servers response to the browser.
- The browser will store the cookie on the clients machine and will send it back to the server with all subsequent requests.
- Once stored on clients PC's, you can use cookies for many tasks.

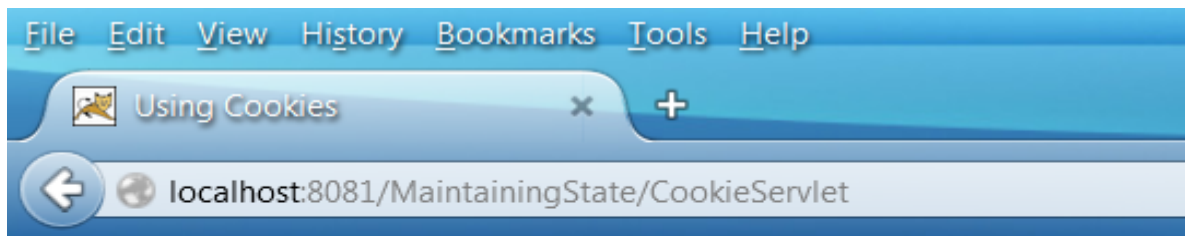
# How Cookies Work

- They are name/value pairs that are stored in the browser.
- The web application creates the cookie and sends it to the client where it is stored on the browser. The client sends it back to the server with every request.
- Cookies can be set to persist on the browser for 3 years.
- Some browsers will have cookies disabled so you can't always rely on them.
- Browsers generally accept only 20 cookies from each site and 300 in total. In addition they can limit each cookie to 4 kilobytes.
- A cookie can be associated with one or more subdomain names.

# Cookies – Example



Enter Your Name



Hello there.. Alan Ryan



# Cookies – Example

## Partial code-listing for index.jsp

```
15 <form action="../../../CookieServlet" method="post">
16
17     Enter Your Name <input type="text" name="name"/>
18     <input type="Submit" value="Send Us Your Name"/>
19
20 </form>
```

## Partial code-listing for CookieServlet

```
21 Cookie c = new Cookie("username", request.getParameter("name"));
22
23 c.setMaxAge(60 * 60 * 365 * 2); //set the age of the cookie to two years
24
25 response.addCookie(c);
26
27 RequestDispatcher dispatcher = request.getRequestDispatcher("CookiesExample1/page2.jsp");
28
29 dispatcher.forward(request, response);
```

# Cookies – Example

Partial code-listing for page2.jsp

```
11      Hello there..  
12  
13      ${cookie.username.value}  
14  
15      <br>
```

# Working With Cookies

## Constructor method of the Cookie class

Return Type	Method Name	Description
N/A	Cookie(String name, String value)	Creates a cookie instance with a specified name and value

## Useful methods of the Cookie class

Return Type	Method Name	Description
String	getName()	Gets the name of the cookie.
String	getValue()	Gets the value of the cookie
void	setPath(String path)	Specifies a path for the cookie.

# Working With Cookies

## Useful methods of the Cookie class

Return Type	Method Name	Description
void	setMaxAge(long expiry)	Sets the max age of the cookie in seconds.
void	setDomain(String pattern)	Specifies the domain within which this cookie should be presented.
void	setSecure(boolean flag)	Indicates to the browser whether the cookie should only be sent using a secure protocol, such as HTTPS or SSL.

# Working With Cookies

## Useful method of the [HttpServletResponse](#) object

Return Type	Method Name	Description
void	addCookie(Cookie c)	Adds the specified cookie to the response.

## Useful method of the [HttpServletRequest](#) object

Return Type	Method Name	Description
Cookie[]	getCookies()	Returns an array containing all of the Cookie objects the client sent with this request.

# Working With Cookies

## Creating and returning a cookie to the browser

```
Cookie c = new Cookie("username", request.getParameter("name"));

c.setMaxAge(60 * 60 * 24 * 365 * 2); //set the age of the cookie to two years

c.setPath("/"); //allow the entire app to access the cookie

response.addCookie(c); //add to response
```

## Getting the cookie from the browser

```
Cookie[] cookies = request.getCookies();

String cookieName = "studentID";

String cookieValue = "";

for(int i = 0; i < cookies.length; i++) {

    Cookie c = cookies[i];

    if(cookieName.equals(c.getName()))
        cookieValue = c.getValue();

}
```

# Using URL Rewriting and Hidden Fields

- In the early days of web programming, programmers used URL rewriting and hidden fields to track sessions.
- Today the Servlet API is used to track sessions.
- However, URL rewriting and hidden fields are still used to pass parameters between the browser and the server.
- They are handy if you need to pass data for a single request.
- Rather than storing that type of data in the session object – which takes up memory – you can use URL rewriting and hidden fields to pass data from page to page.
- We will examine URL rewriting and hidden fields later.

## References

Murach, J., (2014) *Murachs Java Servlets JSP*, 3rd edn. Mike Murach and Associates, Inc.

Jendrock E, Cervera-Navarro R, Evans I, Hasse K, Markito W  
(2014) *The Java EE 7 Tutorial*, 5th edn. Addison-Wesley Professional.

<http://docs.oracle.com/javaee/6/tutorial/doc/>