7 : Servlet Development and Deployment

IT4206 – Enterprise Application Development

Level II - Semester 4

Overview

- This topic will discuss the client server architecture, web servers, processing of HTML forms, and developing servlets.

Intended Learning Outcomes

- At the end of this lesson, you will be able to;
  - Explain Client server architecture.
  - Discuss the HTTP request and response model.
  - Develop a simple web application using servlets

List of sub topics

7.1 Client Server architecture and web servers.

7.2 HTML and HTTP.

7.3 What is a servlet?

7.4 Processing HTML forms.

7.5 Servlet life cycle.

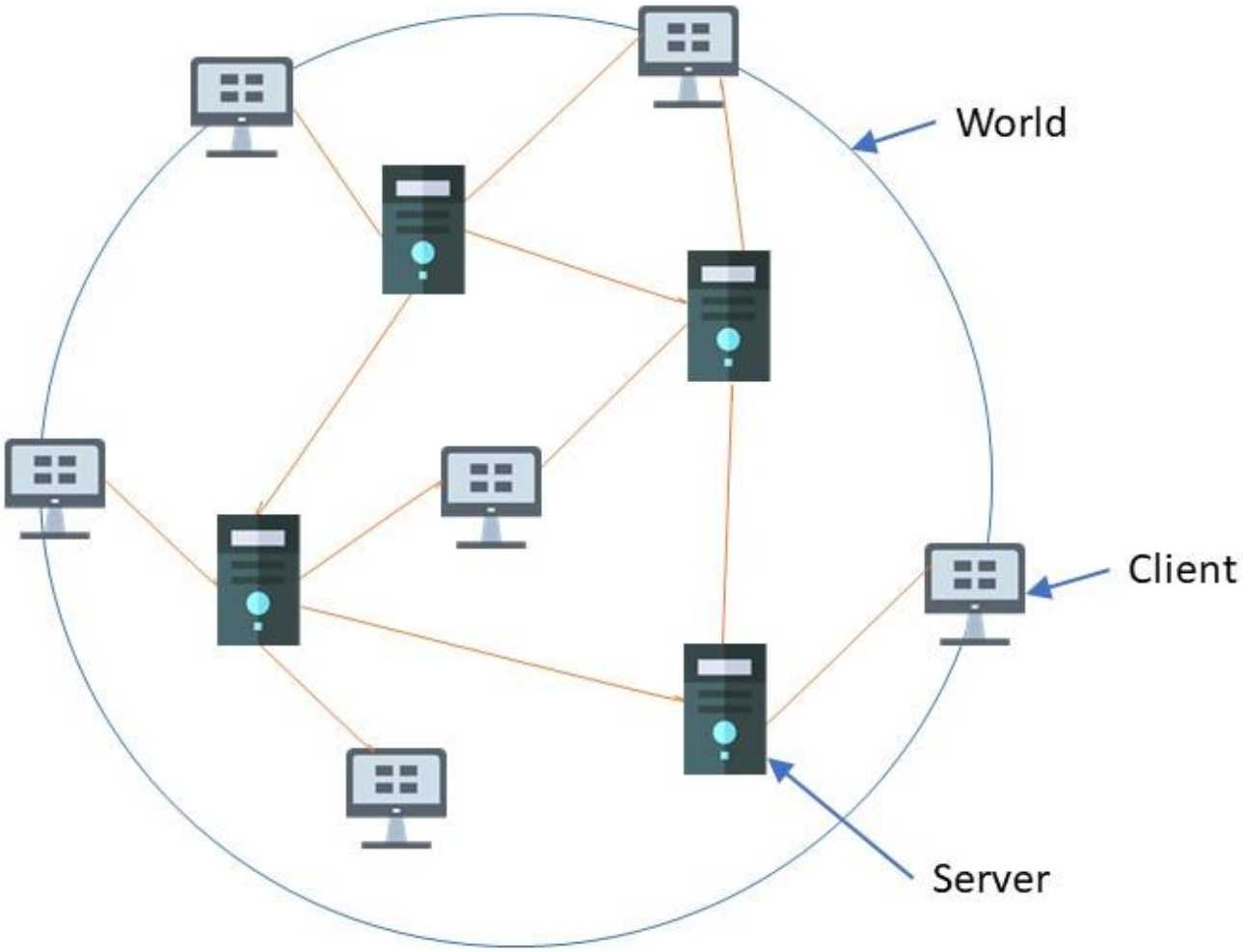
7.6 Request forwarding and response redirection.

7.7 Servlet listeners.

Client Server architecture and web servers

- The Web consist of large number of clients and servers.
- Examples for clients - web browsers (Chrome, Firefox, Safari and etc)
- Examples for servers - Apache web server
- All these clients and servers are connected using wired or wireless networks.
- Clients all around the world can access the web applications executes in the servers by using the internet.

The Web

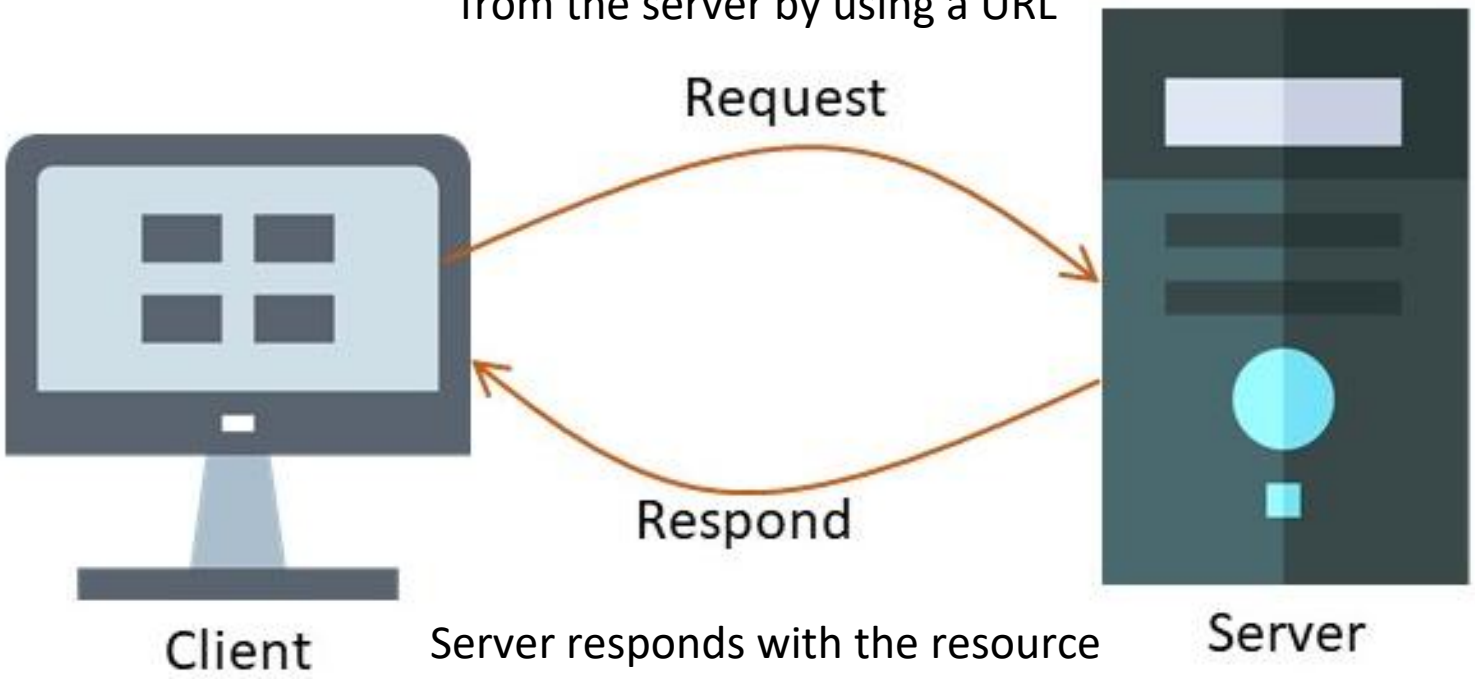


What is a web server

- Web server gets the client request and provide the respond to the client.
- The server can be a hardware device or a web server application (software).
- User can use a web browser to request some resource from the web server.
- The server gets the request of the user, find the resource and return it to the user.
- The resource can be a picture, audio clip, video clip, a document or may be a HTML web page.
- If the server failed to find what the user request, it will inform that the resource is not available.

What is a web server...(2)

Client request a resource
from the server by using a URL

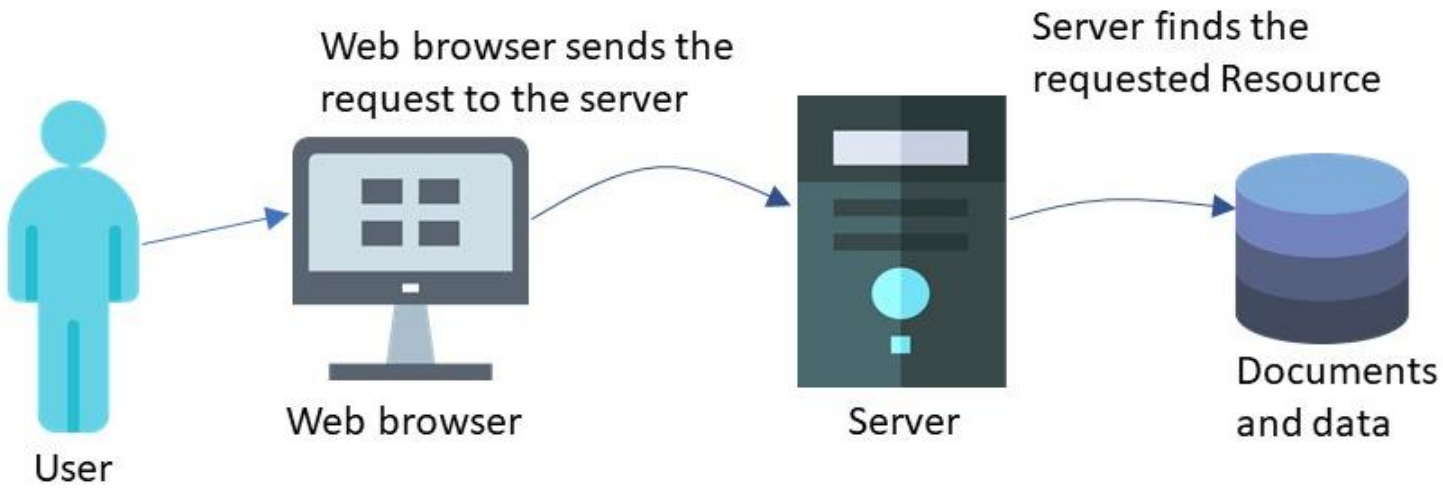


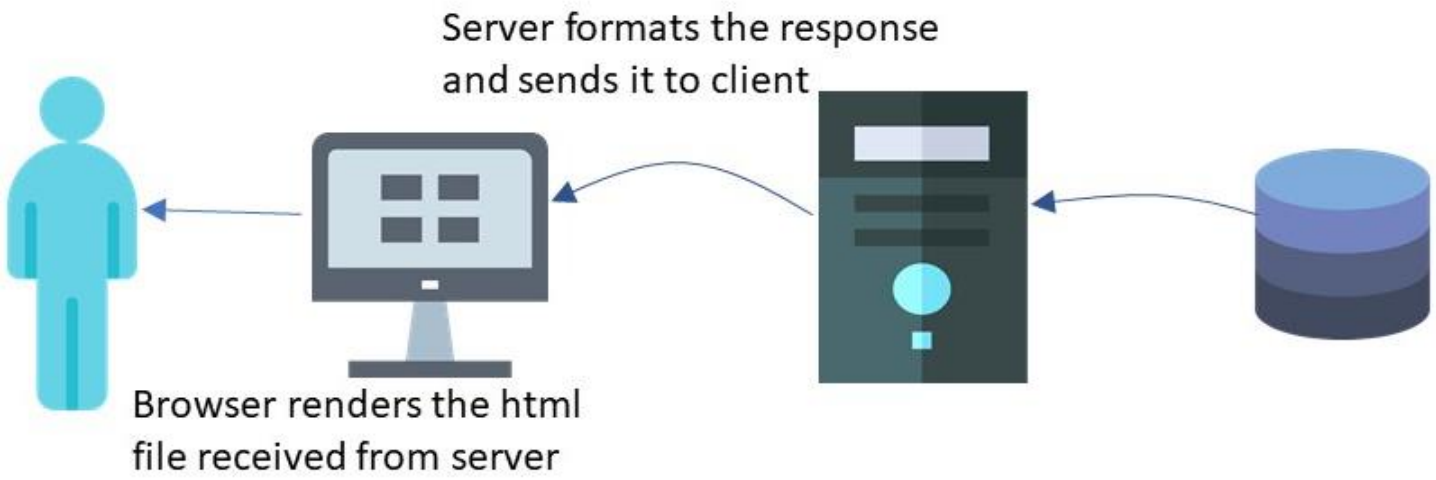
Server responds with the resource
Requested by the client

What is a Client

- Clients can be human users or web applications (eg- web browsers)
- Web browsers can be used to communicate with the web servers.
- Web browser can interpret and render the responses got from the web server.

What is a Client...(2)





HTML and HTTP

- HTML - Hyper Text Markup Language. HTML tell the browser how to display the content to the user.
- HTTP - Hyper Text Transfer Protocol. Client and server use HTTP for the web communication.
- In the communication between client and the server, client sends a HTTP request to the server. The server answers with a HTTP respond.
- The server uses HTTP to send HTML to the client.

Recall HTML

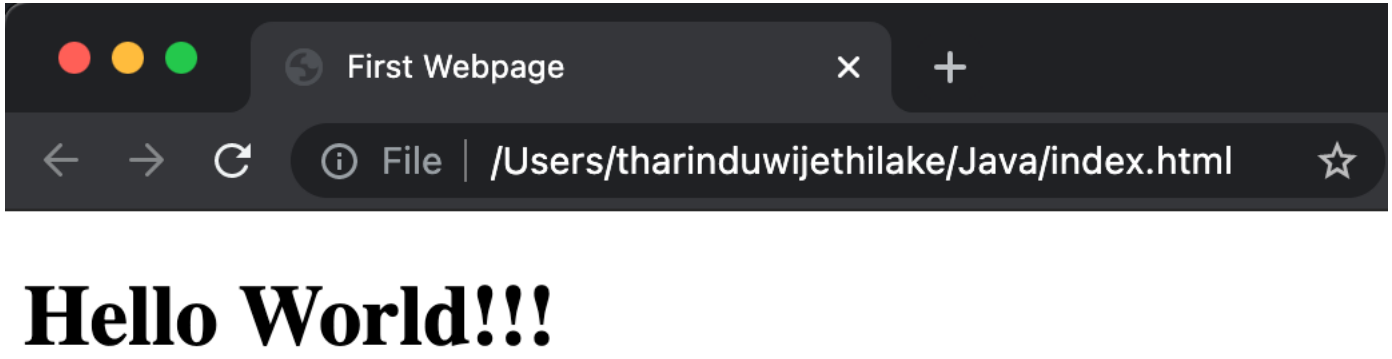
- HTML is a scripting language
- Content and styles are intermingled
- Three basic tags
  - <html> - Represents the complete HTML document
  - <head> - Contains page information.
  - <body> - Contains the visible content of your webpage
- Files are saved using the *.html, *.htm extensions
- Internet media type “text/html”
- Interpreted by your web browser
- HTML pages are static pages.

Simple HTML file

```
<!doctype html>
<html>
	<head>
		<title>First Webpage</title>
	<head>
	<body>
		<h1>Hello World!!!</h1>
	<body>
</html>
```

Simple HTML file...(2)

- Open the Web page



HTML Forms

- Forms are used to collect information from the user
- The <from> element defines a HTML form
- Different form elements such as text boxes, text areas, buttons, checkboxes are placed in the HTML form using the <input> tag.
- Input tag defines different elements based on the value submitted to the type attribute

<input type=?>

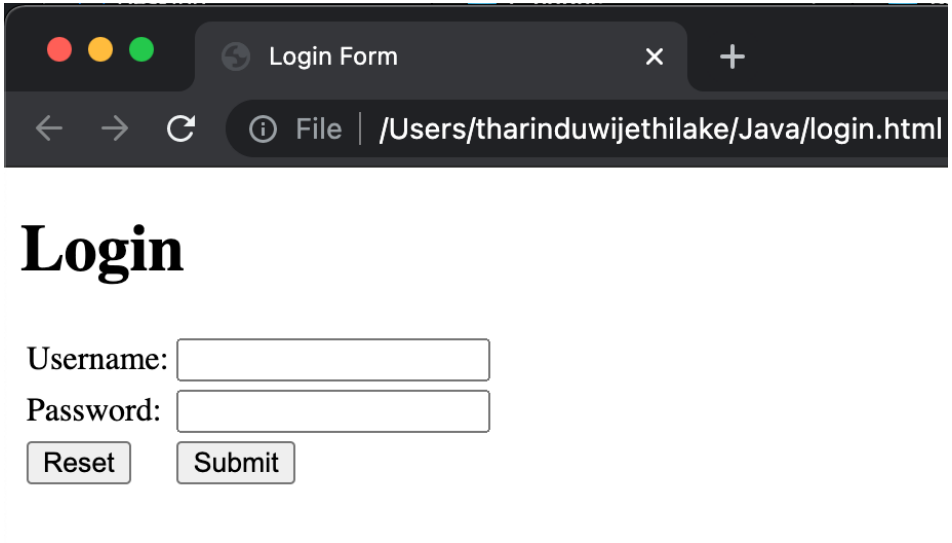
  - text -> a textbox
  - radio -> radio button
  - submit -> button with submit functionality
  - reset -> button resets the form to default

HTML Forms...(2)

- The form tag contains number of attributes used for various purposes
- The “action” and the “method” attributes are the most commonly used attribute.
- The action attribute specifies what needs to be done when the from is submitted
- The method attributes specifies the method of submission

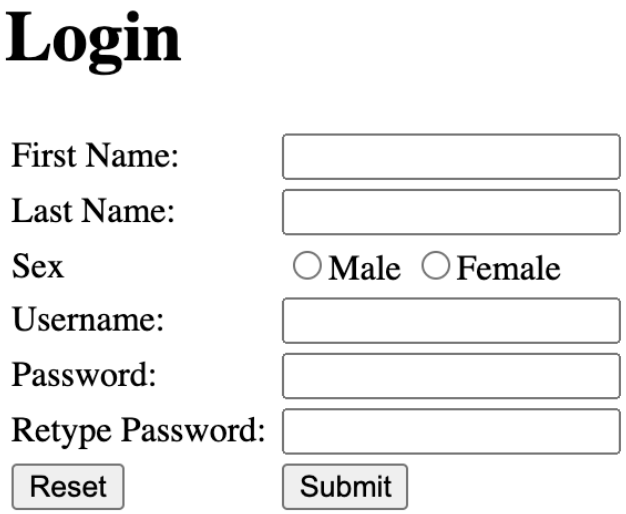
Practical Activity 1

- Create a Login screen
  - A simple Login Screen with username and password fields
  - Should have a submit button and a reset button



Practical Activity 2

- A simple registration form
  - First name, Last name, Username as text fields
  - Sex as a radio button with male selected by default
  - A submit button – When clicked should take the user to the login screen previously created.
  - A reset button



Recall Javascript

- High-level interpreted programming language for the web
- Variables are defined using the following syntax (no data type)

var <variable_name> = <value>

- Variable name can be alphanumeric
- Value can be numbers or strings

Recall Javascript...(2)

- Basic arithmetic operations and logic operations are same as in java.

- JavaScript functions are made up of three parts:
  - Function name
  - Parameters
  - Body

```
var percentage = function(num, denominator){
  var result = (num / denominator) * 100;
  return result;
}
```

- Function is called by its name with parameter values specified
  - E.g. var test = percentage(30,70);

Recall Javascript...(3)

- Can be embedded in HTML files or can be placed in a different file with the *.js extension.
  - When embedding in HTML the JavaScript is placed with the <script> tags
  - The <script> tags are placed in the <head> section of the HTML file.

- To accesses the HTML page elements JavaScript uses the document property.
  - document.forms -> will return all the forms in the HTML document
  - document.forms[“loginform”] -> will return the form with the name “loginform” from the set of forms in the HTML document.

Recall Javascript...(4)

- Validate function to check for a non-empty user name

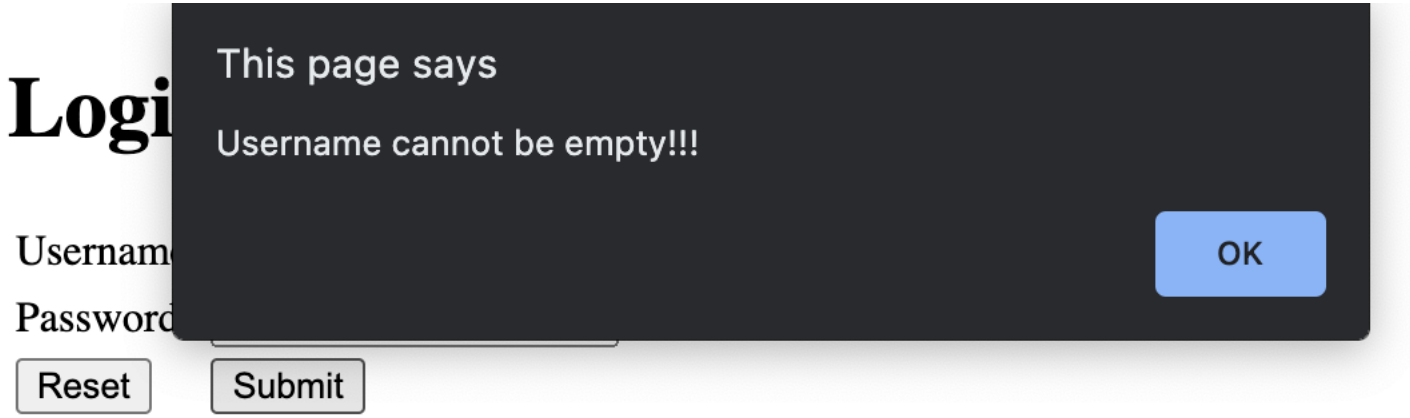
```
<script type="text/javascript">
  function checkUser() {
      var i = document.forms["loginform"]["username"].value;
      if (i == null || i == "") {
          alert("Username cannot be empty!!!");
          return false;
      }
  }
</script>
```

- The method is called with the user click the submit button using the “onsubmit” attribute in the form tag.

```
<form action="index.html" method="post" name="loginform" onsubmit="return checkUser()">
```

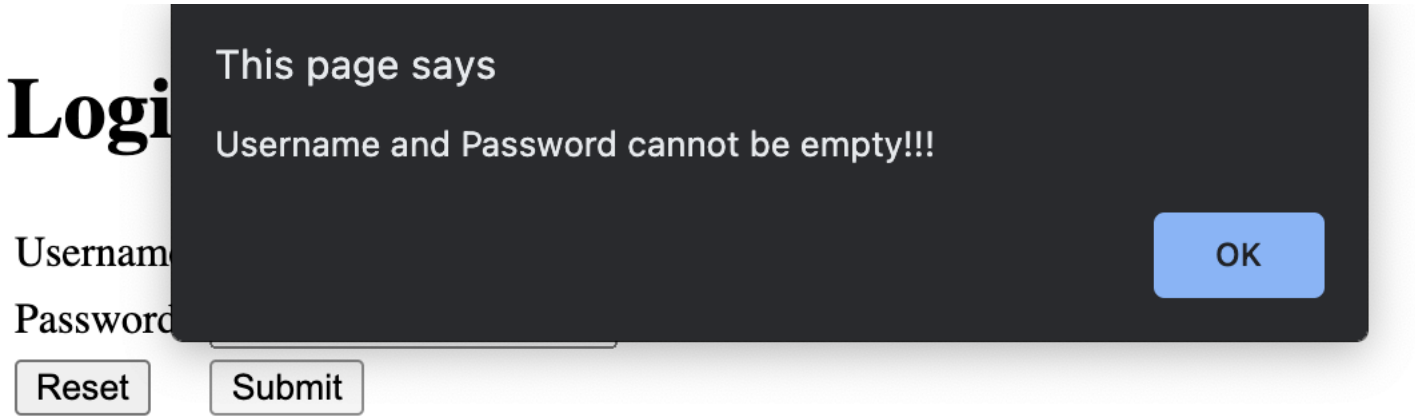
Recall Javascript...(5)

- Message will display as follows



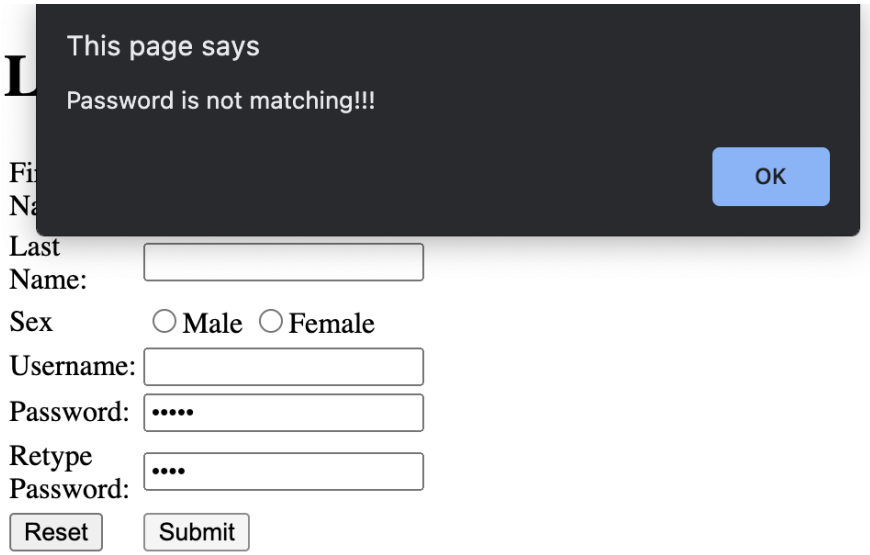
Practical Activity 3

- Validate the Login form
  - Username and password should not be empty



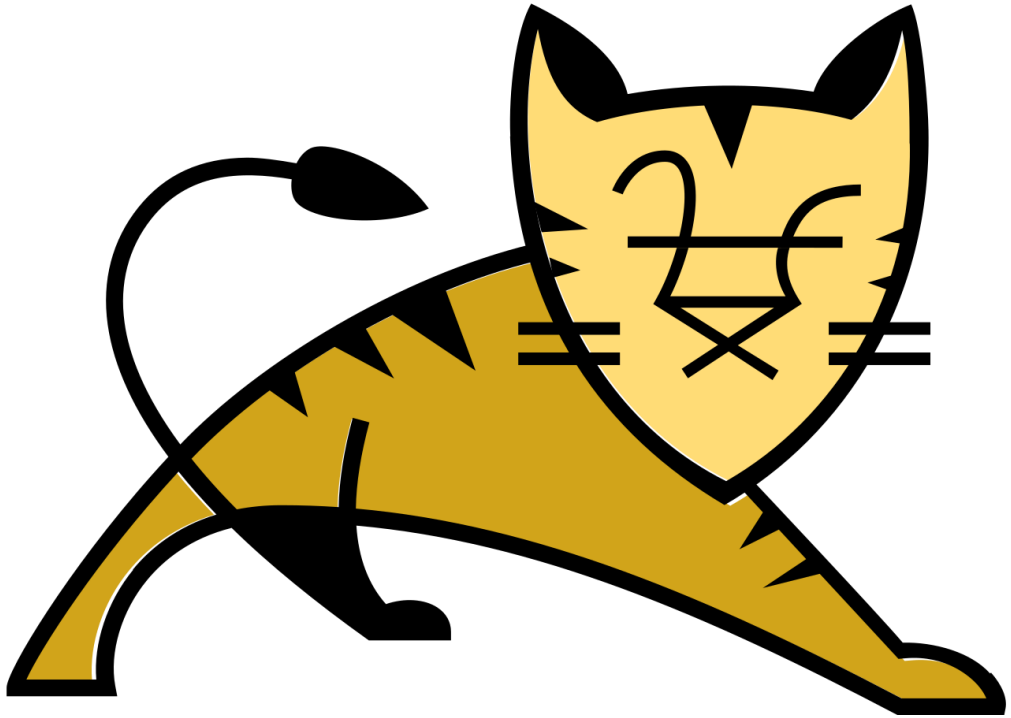
Practical Activity 4

- Validate the registration from
  - All fields should not be empty
  - Check whether the password entered in the retype password field are the same.



Apache Tomcat

- Apache Tomcat - is a free and open-source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and WebSocket technologies. Tomcat provides a "pure Java" HTTP web server environment in which Java code can run. ~wikipedia

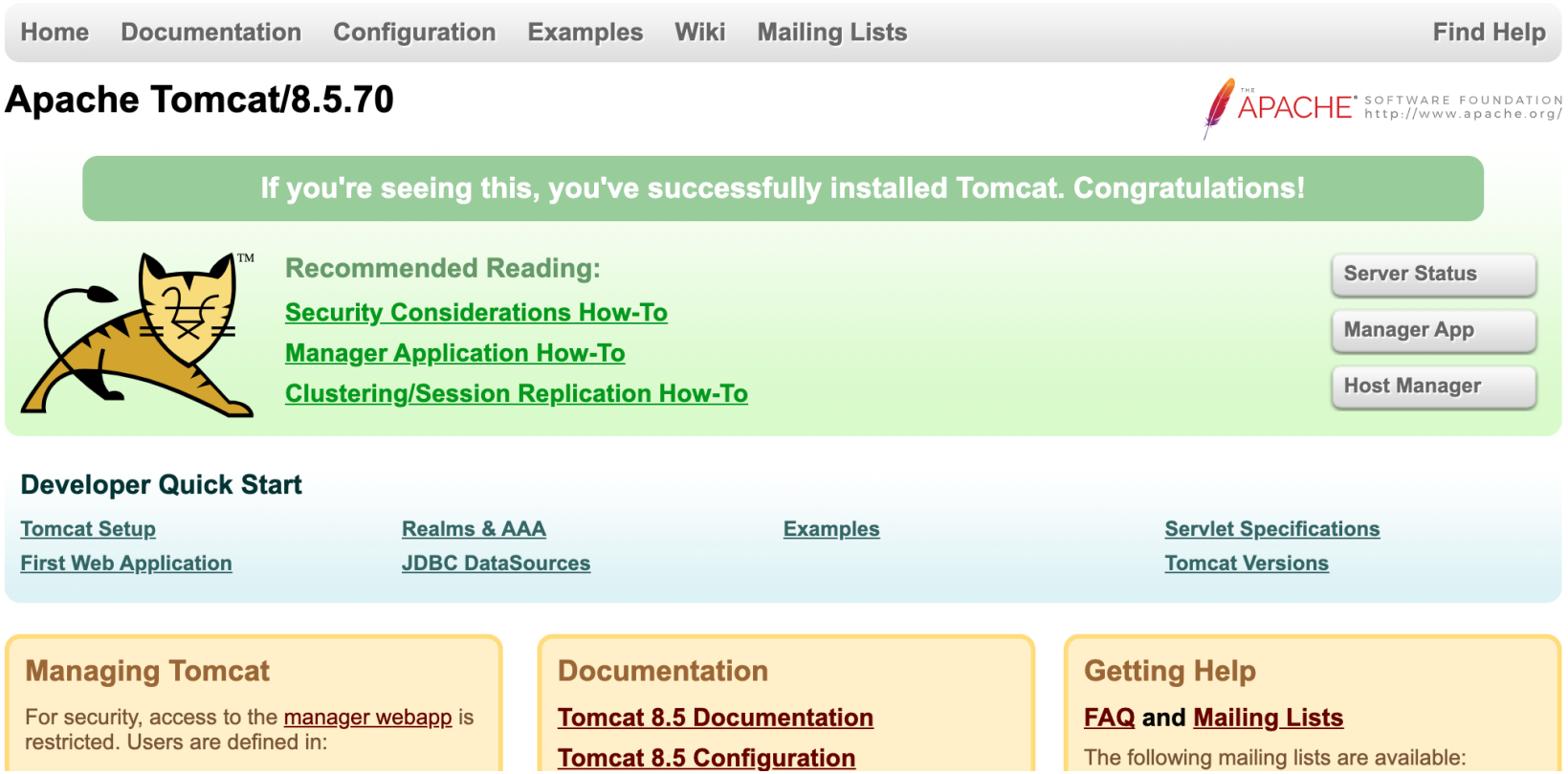


Practical Activity 5

- Download and setup Apache Tomcat (Tomcat 8 is recommended).
- https://tomcat.apache.org/download-90.cgi
- Start the Tomcat server
  - <path to tomcat>/bin
  - Run ./startup.sh to start and ./shutdown.sh to stop the server

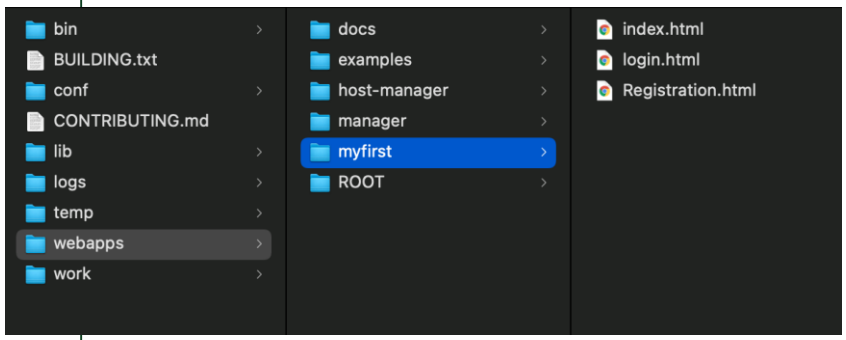
Apache Tomcat...(2)

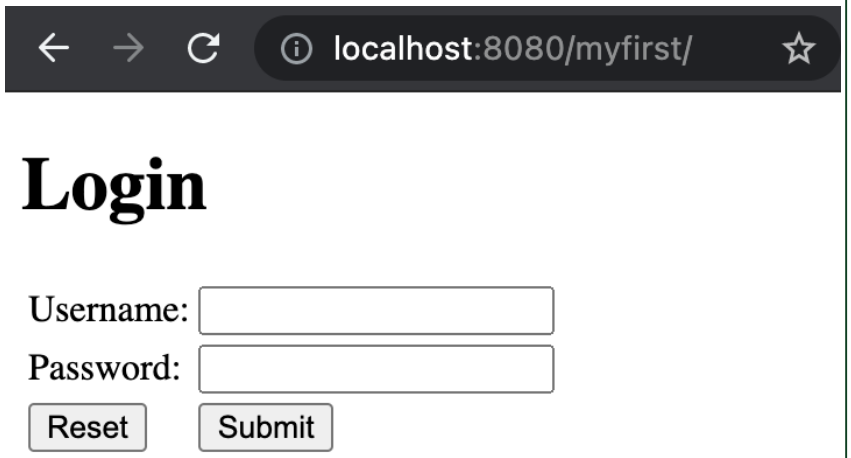
- After installing and running Apache Tomcat, goto http://localhost:8080/



Practical Activity 6

- Deploy your HTML on Tomcat.
  - Create a new directory (myfirst) in webapps of Tomcat.
  - Copy your HTML files to “myfirst” directory.
  - Start Tomcat
  - Type “localhost:8080/myfirst” in web browser to view the HTML files.





Anatomy of a HTTP Request

- HTTP method – Specifies what kind of a request is being sent and the format of the message.

Method	Description
GET	Request information from the server
POST	Submits information to the server for processing
DELETE	Delete a specified resource from the server
OPTIONS	Queries the connection options available
HEAD	Exactly like the GET, but only returns the headers no body

- URL
- Form Parameters

Anatomy of a HTTP Respond

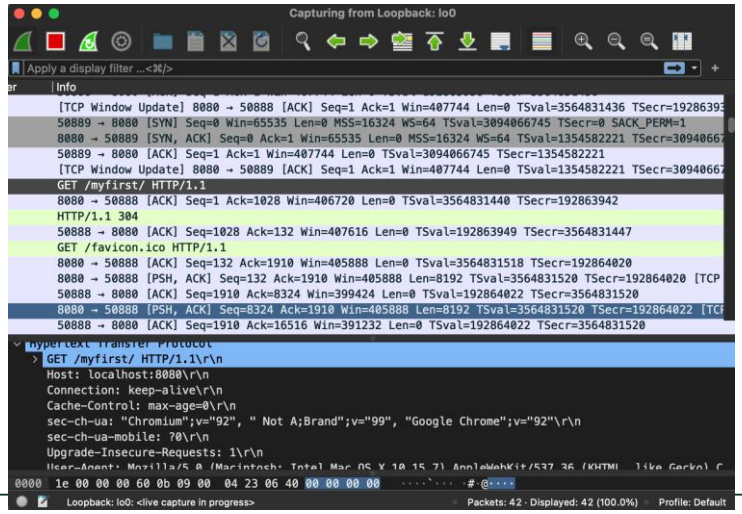
- Status Code – The code returned by the server in response is categorized into five classes based on the first digit.

Code Class	Description
1xx	Informational Messages
2xx	Success Message
3xx	Redirection
4xx	Client Error
5xx	Server Error

- Content Type – e.g. “text/html”
- Content

Practical Activity 7

- Observe GET and POST requests using the Wireshark software.
  - Install and open Wireshark application (https://www.wireshark.org/download.html)
  - Choose the interface to monitor as Lo0 (Loopback)
  - Visit “http://localhost:8080/myfirst/" while monitoring.
  - Observe the HTTP request and HTTP responds.



What is a Servlet

- Servlets are the Java platform technology of choice for extending and enhancing web applications.
- Web applications are helper applications that resides at web server and build dynamic web pages.
- Robust, scalable and secured
- Servlets can also access a library of HTTP specific calls and receive all the benefits of the Java language, including portability, performance and reusability.

What is a Servlet...(2)

- Servlets have access to the entire family of Java APIs including the JDBC™ API to access enterprise databases.

- This also provide better alternative to Common Gateway Interface (CGI), Netscape Server Application Programming Interface (NSAPI), Internet Server Application Programming Interface (ISAPI) etc.

Web Container

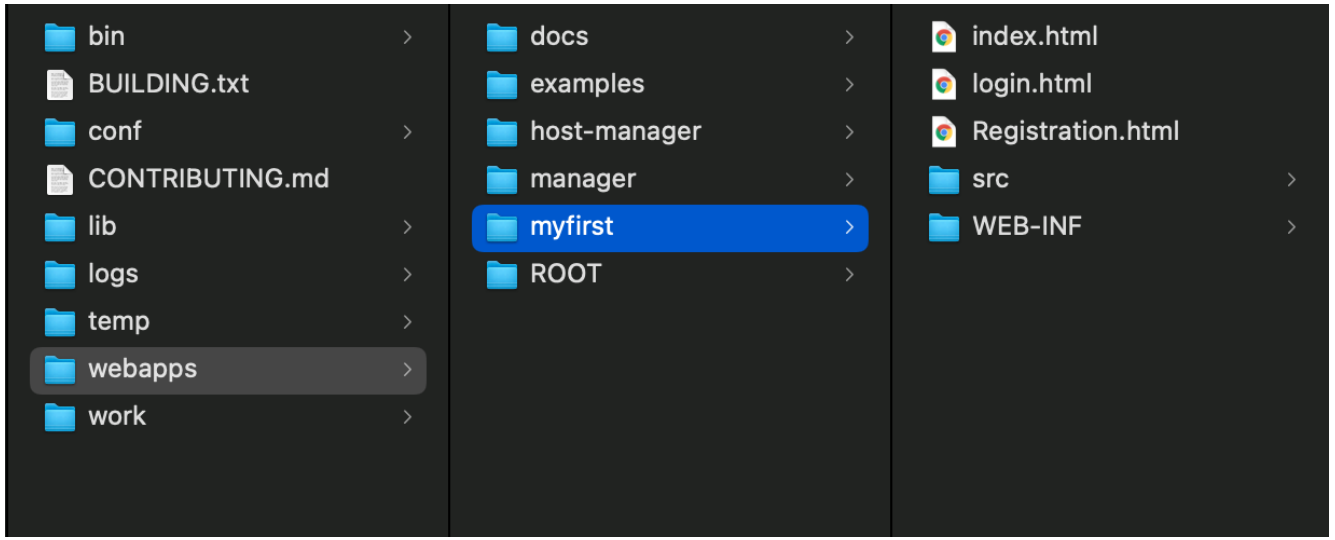
- Is the on that enables the web server to communicate with the servlet.
- Servlet lives and dies within the web container. (Life Cycle Management)
- Web container is responsible for invoking the methods on the servlet.
- Web container acts as a middleware
- Provides JSP support

Servlets

- Nothing but a Java program
- Has no main method.
- Invoked via the web container.
- Used when dynamic content is required in your website.

First Servlet Program

- We can use the same directory “myfirst” created inside the webapp directory.
- Create two directories are “src” and “WEB-INF” inside the “myfirst” directory.
  - src - for source files
  - WEB-INF - for Java classes



First Servlet Program...(2)

- Create the FirstServlet.java as follows and save it in “src” directory.

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

public class FirstServlet extends HttpServlet {

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

  PrintWriter out = response.getWriter();
  out.println("<html>");
  out.println("<head>");
  out.println("<title>First Servlet</title>");
  out.println("</head>");
  out.println("<body>");
  out.println("<h1>");
  out.println("This is the First Servlet.");
  out.println("</h1>");
  out.println("</body>");
  out.println("</html>");
 }
}
```

First Servlet Program...(3)

- Create the deployment descriptor, web.xml inside the WEB-INF directory.

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                      http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  version="3.1"
  metadata-complete="true">

  <servlet>
   <servlet-name>FirstServlet</servlet-name>
   <servlet-class>FirstServlet</servlet-class>
 </servlet>

 <servlet-mapping>
   <servlet-name>FirstServlet</servlet-name>
   <url-pattern>/FirstServlet</url-pattern>
 </servlet-mapping>

</web-app>
```

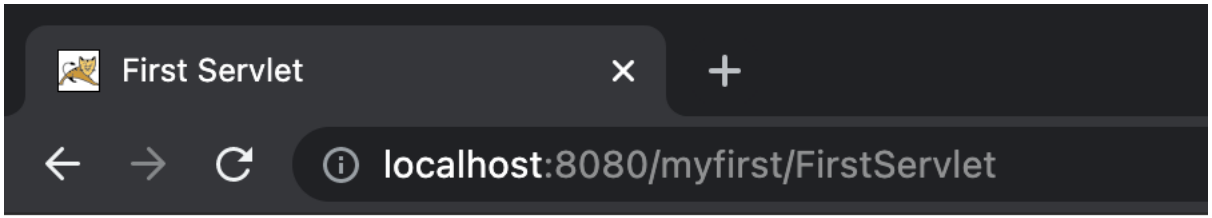
First Servlet Program...(4)

- Compile the servlet using the following command.

```
javac -cp <path to tomcat>/lib/servlet-api.jar -d ../WEB-INF/classes FirstServlet.java
```

- As given in the above command, the class path to the servlet-api.jar has to be given in the command or it has to be configured beforehand.
- The .class file will save inside the /WEB-INF/classes directory.
- Start the tomcat server and goto
  - http://localhost:8080/myfirst/FirstServlet

First Servlet Program...(5)



This is the First Servlet.

Apache Folder Structure

- Each application in the server has specific folder inside of the webapps folder.
  - At the top level inside the folder for that application consists of
    - *.html and *.jsp files
    - WEB-INF folder
      - web.xml at the top level
      - Folders named “classes” and “lib” should be at the top-level of the WEB-INF
      - The class files with the package structure is placed in the “classes” folder
- META-INF deals with manifest file and generally auto generated

Deployment Descriptor

- This is a configuration file that describes how a servlet should be deployed.
- The deployment descriptor is named web.xml
- It is placed in the WEB-INF folder
- <servlet> block allows us to take control of the servlet. Block require at least two parameters inside it.
  - <servlet-name> - This is just a variable. Does not necessarily have to match the servlet name.
  - <servlet-class> - This is the servlet class we developed before with the complete package reference.
- <servlet-mapping> allows us to take control of the servlet URL mapping. Block require at least two parameters inside it.
  - <servlet-name> - Variable name that links the actual servlet class with the URL mapping
  - <url-pattern> - The mapping we want when attempting to invoke the servlet.

Servlet API

- Servlet API consist of two packages that contains all the important classes and interfaces
  - javax.servlet
  - javax.servlet.http

Servlet Interface

- Contained in the javax.servlet package

- Contains five methods
  - Three methods are life cycle methods
    - service(ServletRequest, ServletResponse)
    - init(ServletConfig)
    - destroy()
  - Two are general purpose methods
    - getServletConfig()
    - getServeltInfo()

Generic Servlet Class

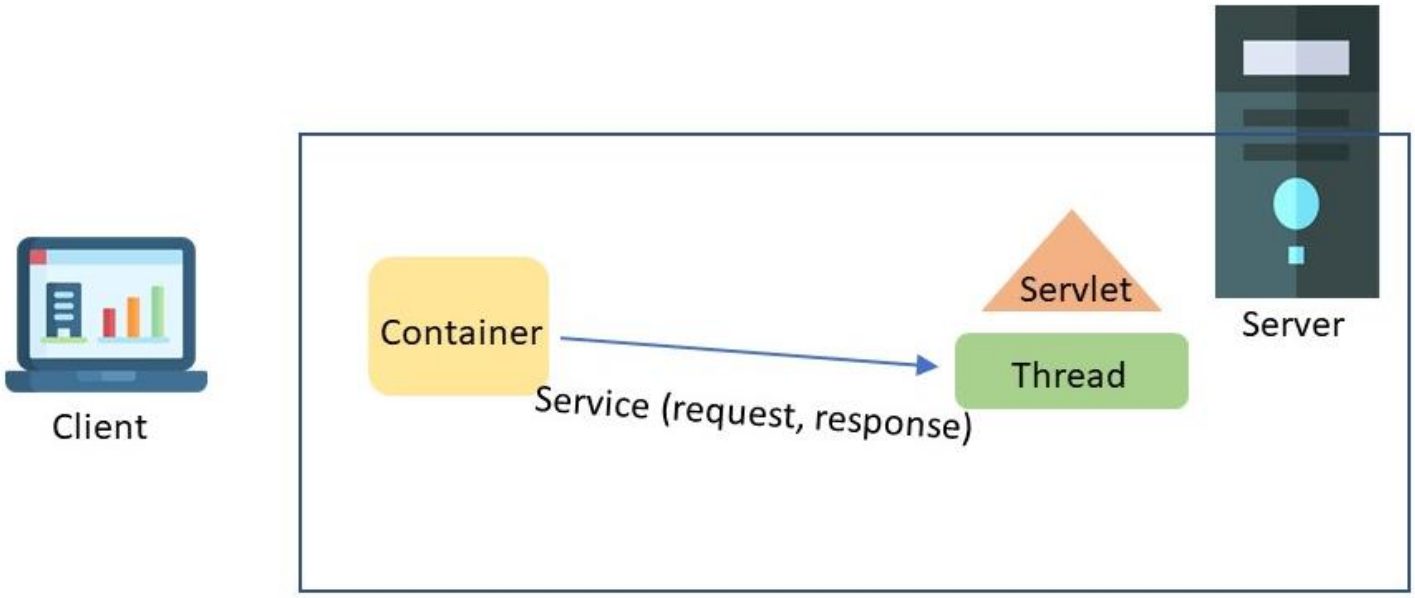
- This is an abstract class available in the `javax.servlet` package.
- Implements most of the important methods associated with servlets.
- Class implements `Servlet, ServletConfig` and `Serializable` interfaces.
- This class can handle any type of servlet request. Hence it is protocol independent.

HttpServlet Class

- `HttpServlet` is abstract class that extends the `GenericServlet` class.
- Provides HTTP specific methods.
- The `service()` method listens to the HTTP methods from the request stream and invokes them accordingly.
- The `service()` method is generally not overridden.

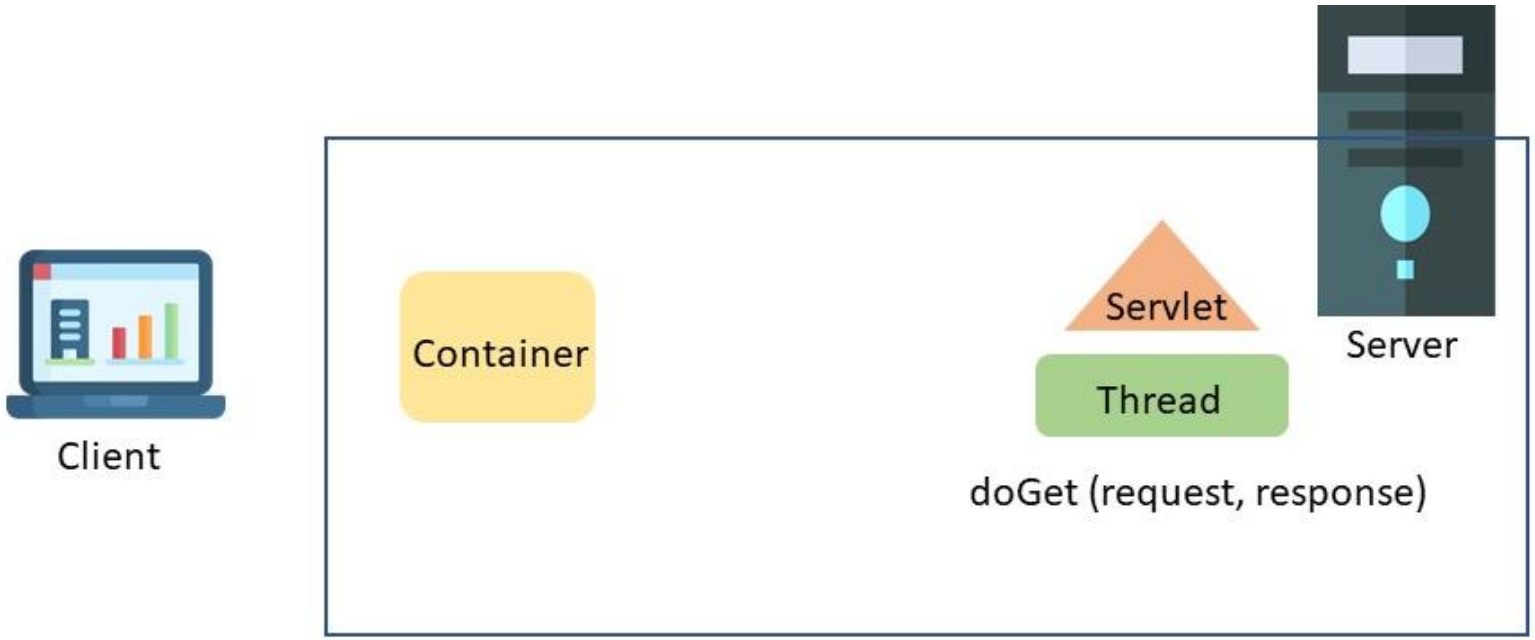
Servlet Life Cycle

- The controller
  - Create the request and response objects
  - Create/ allocate new thread for the servlet
  - Call servlet's service() method
  - Pass the request and response references as arguments



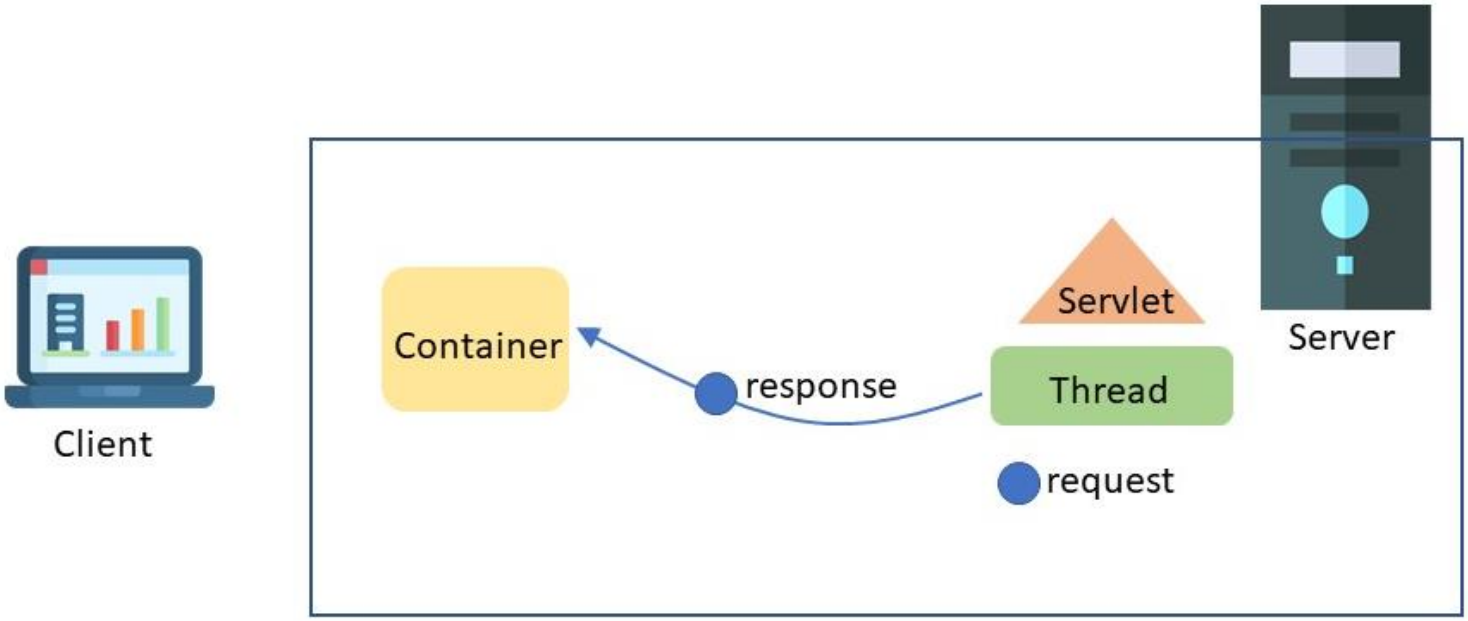
Servlet Life Cycle...(2)

- Based on the HTTP method (get/post), the service() method will select which servlet method to call.
  - For HTTP GET request, doGet() method of servlet.



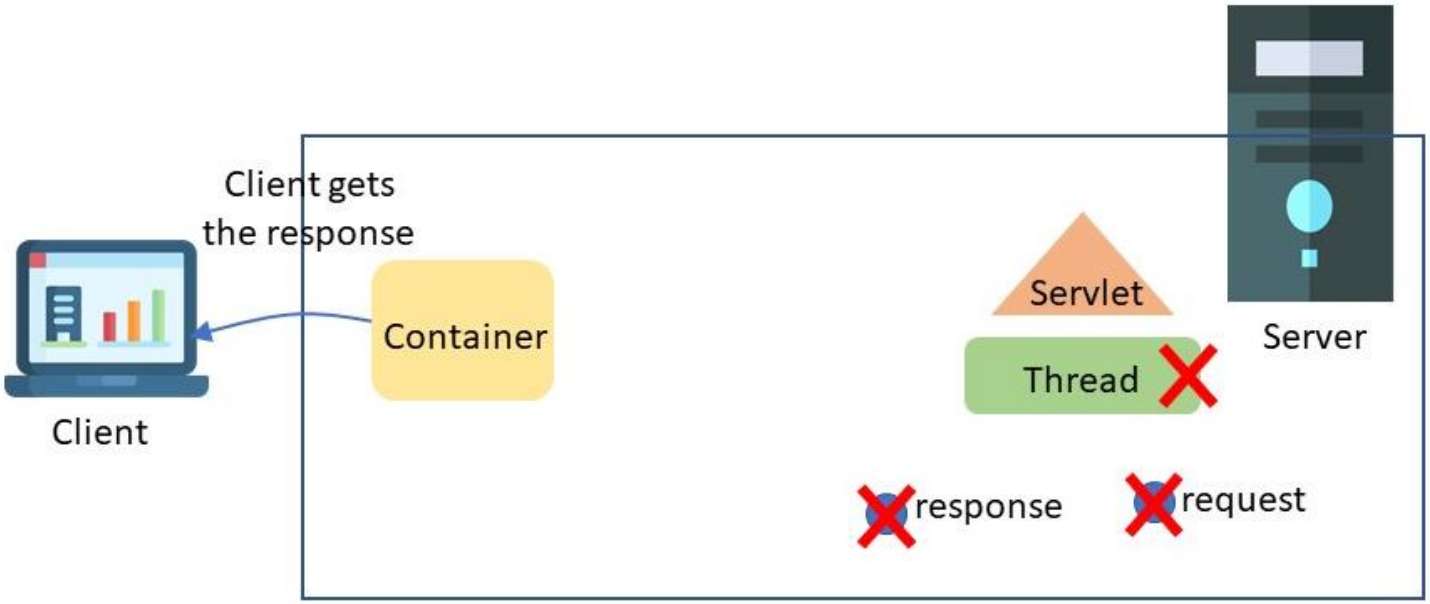
Servlet Life Cycle...(3)

- Servlet use the response object to write out the response to client.
- Response goes back to through the container.



Servlet Life Cycle...(4)

- service() method completes
- Thread dies or return to container managed thread pool
- Request and response object references fall out of scope
- Client gets the response



Servlet Life Cycle...(5)

- Loading the Servlet Class:- A servlet class is loaded when the first request for the servlet is received by the web container.
- Servlet Instance Creation:- After being loaded the web container creates an instance of the servlet
- init():- Is invoked when the web container initializes the servlet instance
- service():- Listens to the HTTP requests and handle them accordingly.
- destroy():- Web container call the destroy() method before removing the servlet instance.

Request and Response

- The purpose of developing servlets is to handle client requests and respond to them.
- Client requests are handled by implementing the `ServletRequest` interface.
- For HTTP specific calls the `HttpServletRequest` interface is used.
- Responses from the servlet to the client handled via the following two interfaces
  - `ServletResponse`
  - `HttpServletResponse`
- Both `HttpServletRequest` and `HttpServletResponse` extends the `ServletRequest` and the `ServletResponse` interfaces respectively.

Access Form Parameters

- Consider the following text field in html file.

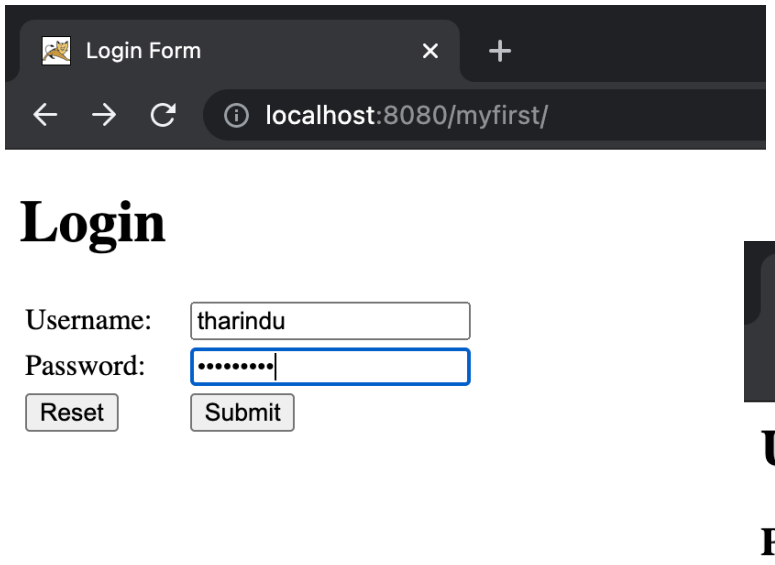
```
<input type="text" name="fname">
```

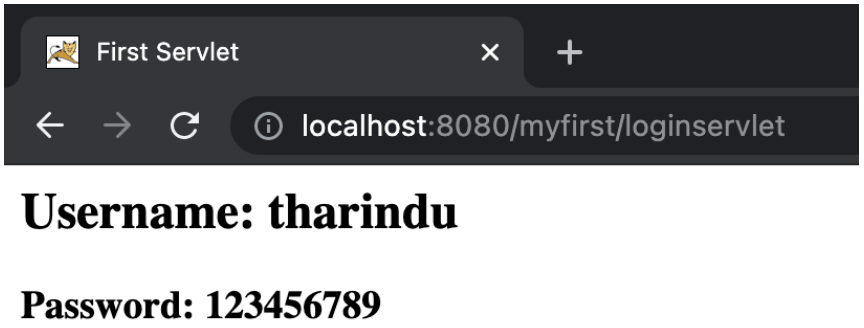
- To access the text field from the servlet `getParameter()` method is used.

```
String firstName = request.getParameter("fname");
```

Practical Activity 8

- Create a servlet application.
  - Use the login.html.
  - When user submit the username and password, display it in a separate page.





Java inside HTML

- Is there a method to put Java inside HTML?
  - Yes, we can use JSP

- Why JSP
  - Not all the HTML page designers know Java. With JSP Java developers can do Java and HTML designers can do web pages
  - Putting HTML into out.println() as arguments is not easy as we done earlier.

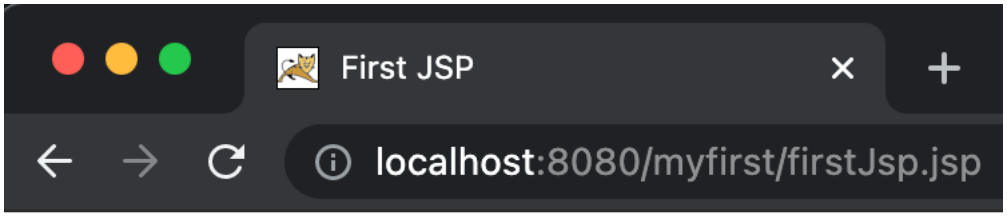
Simple JSP

- Create a new file inside myfirst directory as firstJsp.jsp and write the following code.

```
<!doctype html>
<html>
  <head>
    <title>First JSP</title>
  <head>
  <body>
    <h1>This is my First JSP</h1>
    <% out.print("This is Java inside HTML");%>
  <body>
</html>
```

Simple JSP...(2)

- Start the Tomcat server and visit http://localhost:8080/myfirst/firstJsp.jsp
- The output will be as follows

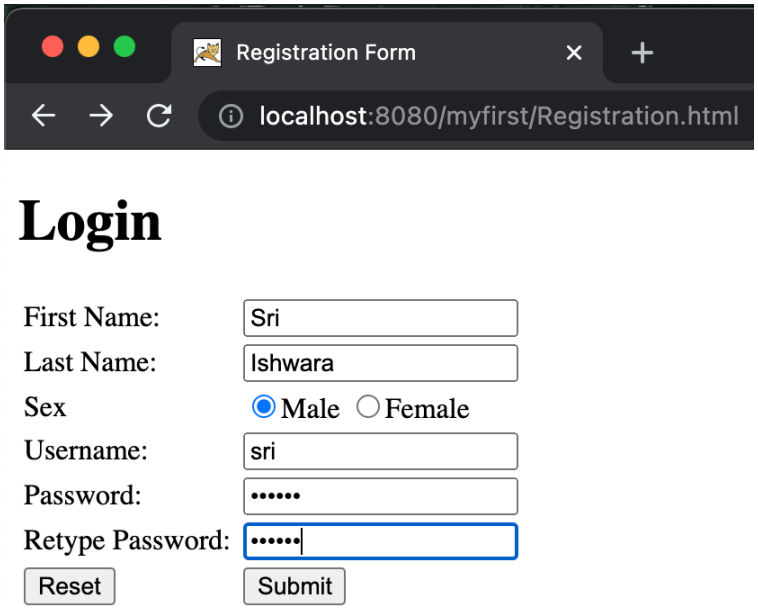


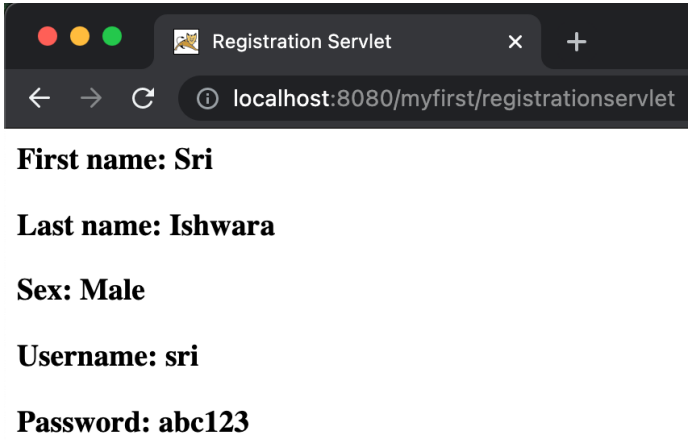
This is my First JSP

This is Java inside HTML

Practical Activity 9

- Create a new servlet to read all the parameters from registration form.
  - Use POST method.
  - Update the web.xml file





Transferring Requests

- To transfer requests between other servlets, html or jsp files we can use the following methods
  - Via the RequestDispatcher
  - Via the sentRedirect

RequestDispatcher Interface

- RequestDispatcher is an interface.
- When implemented this interface creates an objects that can be dispatched to other servlets, html files, jsp etc.
- Provides two methods

```
forward(ServletRequest request, ServletResponse response)
include(ServletRequest request, ServletResponse response)
```

- Standard method call

```
RequestDispatcher rs = request.getRequestDispatcher("servlet.do");
rs.forward(request, response);
```

- OR

```
rs.include(request, response);
```

RequestDispatcher Interface...(2)

`forward(ServletRequest request, ServletResponse response)`

- Forwards an request from a servlet to another resource
- Transfers control to the resource called

`include(ServletRequest request, ServletResponse response)`

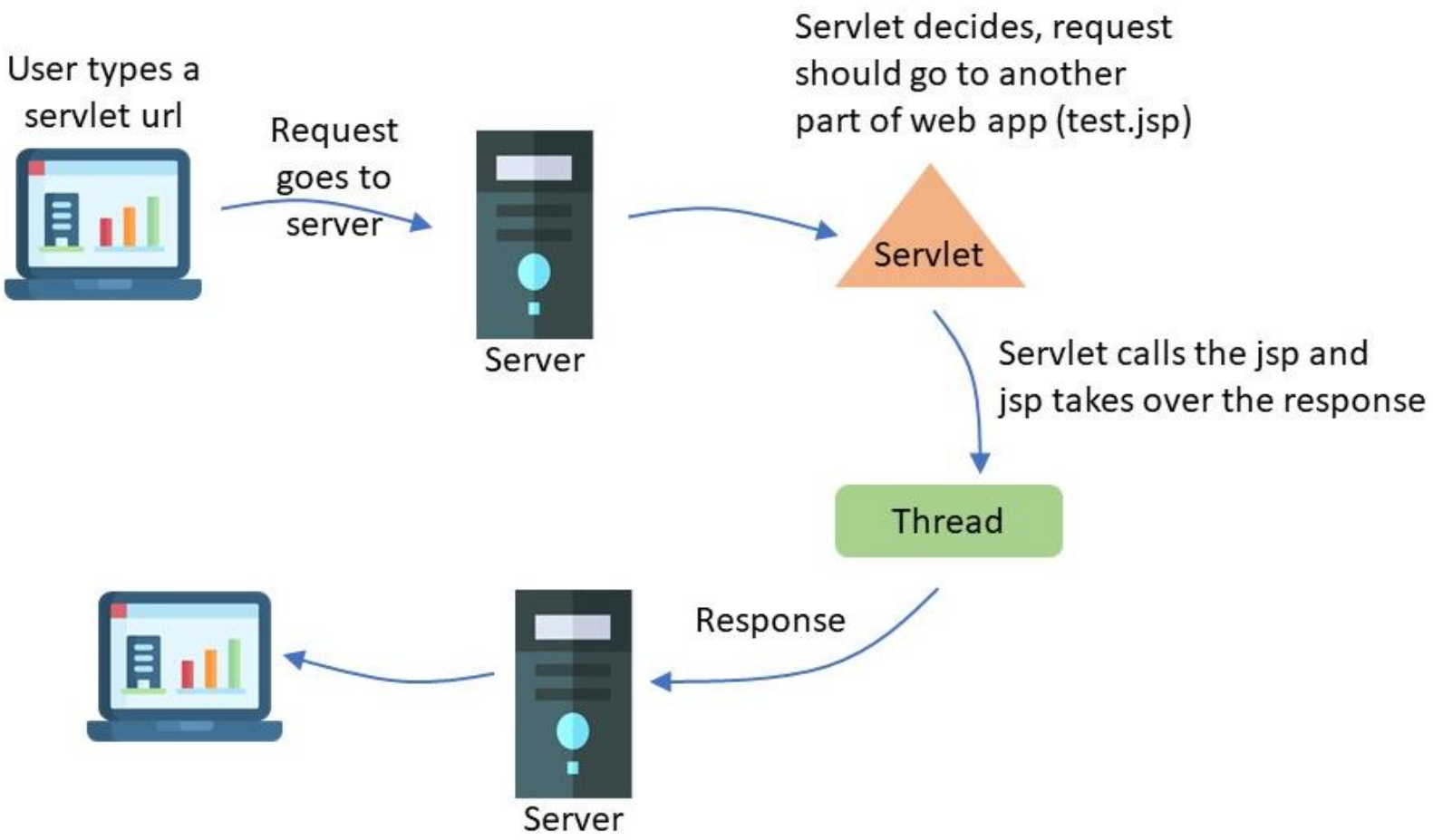
- Includes the content of a resource
- The calling servlet retains control.
  - The included web-component has limited control such as cannot set response header, sessions etc..
- Generally include the resources that are static

RequestDispatcher Interface...(3)

- This method is used to access a resource in the server side

- The client have no knowledge of the redirect. The transfer is directly do by the web container.

RequestDispatcher Interface...(4)



RequestDispatcher Interface…(5)

- Create a new servlet as dispatch.java as follows.

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

public class dispatch extends HttpServlet {

 public void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
   RequestDispatcher view = request.getRequestDispatcher("test.jsp");
   view.forward(request,response);

 }
}
```

RequestDispatcher Interface...(6)

- Create a new JSP as test.jsp as follows.

```
<!doctype html>
<html>
  <head>
    <title>Test JSP</title>
  <head>
  <body>
    <h1>This is the test JSP</h1>
  <body>
</html>
```

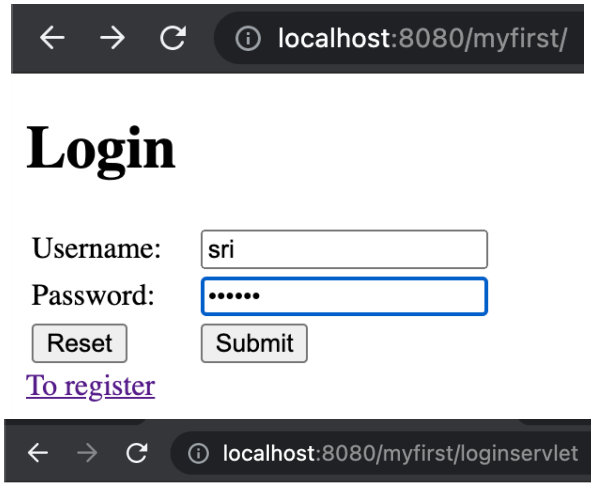
RequestDispatcher Interface...(7)

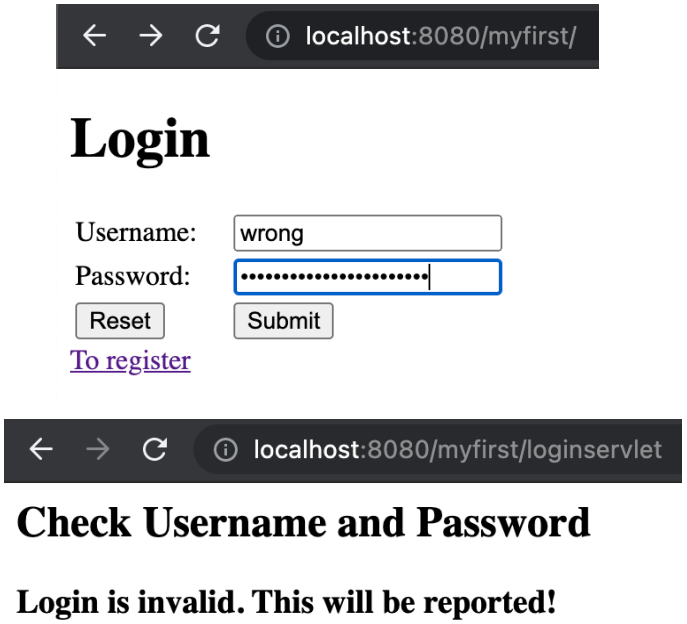
- Compile the servlet, update the web.xml and visit
  - http://localhost:8080/myfirst/dispatch

- The output will be as follows. Since we request from dispatch servlet the service is provided by the test.jsp.

Practical Activity 10

- Use the Login page.
  - If the username and password are correct, transfer control to the success servlet.
  - If the username and password are incorrect, transfer control to the failure servlet.





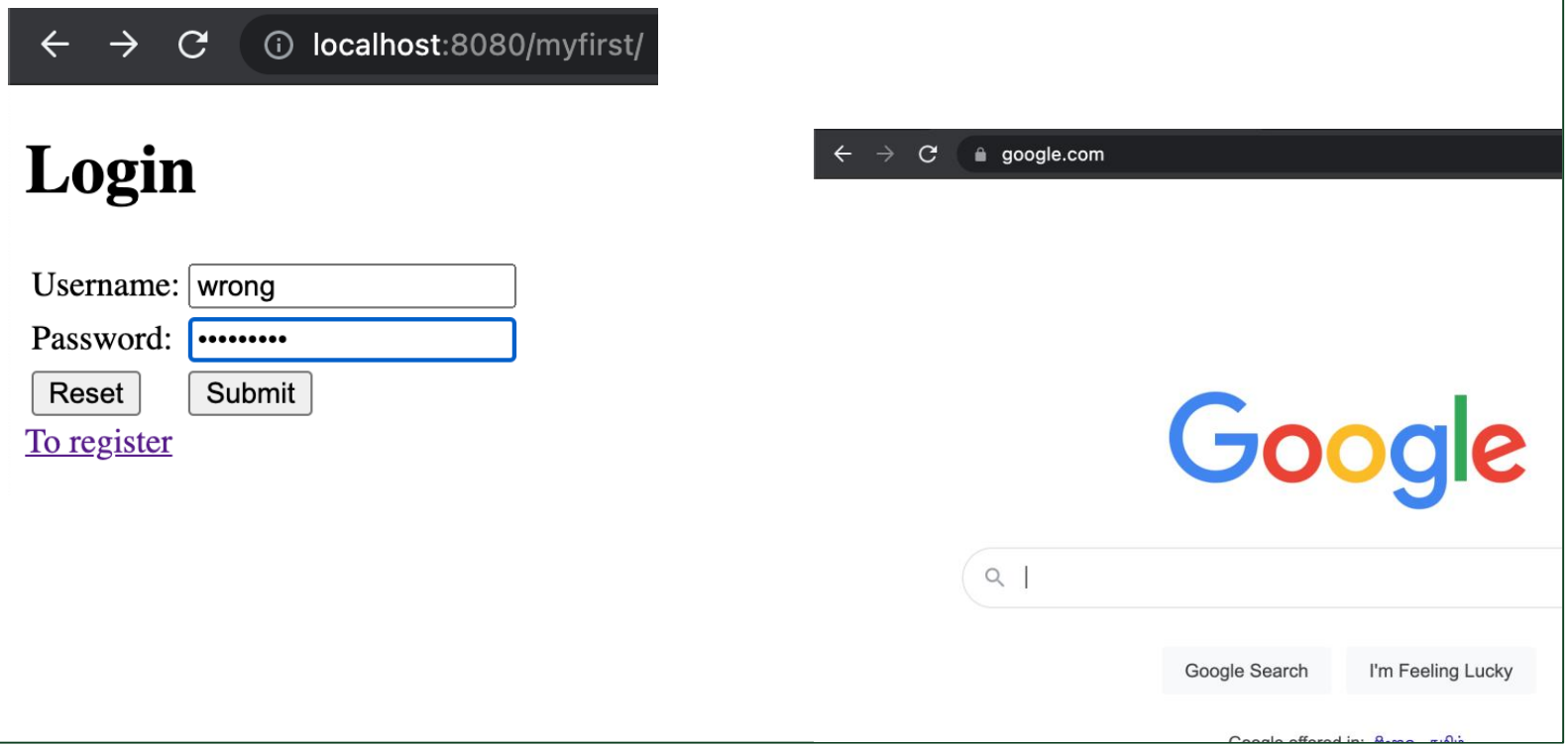
sendRedirect method

- This method is used when it is required to redirect to other resources such as html and jsp files in another server or a domain.
- This method of transfer sends control back to the browse which in turn does a new request.
- The redirect sends a header information back to the browser.

```
response.sendRedirect(http://www.google.com);
```

Practical Activity 11

- Use the Login page.
  - If the username and password are incorrect, redirect to www.google.com



Redirect Vs Request dispatch

- In request dispatch user never knows the request is send to some other party, because the URL never change.
- But in redirect, servlet ask the client to connect to the redirected URL. So the request will be going from the client.

Java Annotations

- What are Annotations?
  - Annotations are a mechanism for associating a meta-tag with the elements of the program.
  - Annotations allows the compiler or the JVM to extract program behaviors from the annotated elements and generate code when required.
  - Java annotations were first introduced in Java 5.
  - Annotations are metadata which can be applied on either annotations OR other java element in java source code.
  - Annotations do not directly affect program semantics, but they do affect the way programs are treated by tools and libraries, which can in turn affect the semantics of the running program.
  - Annotations can be read from source files, class files, or reflectively at run time.

Java Annotations...(2)

- What are Annotations?
  - There are 10 in-built annotations as of today. 5 of them are meant to be applied on custom annotations and other 5 are meant to be applied on java source code elements.
  - Annotations can be read from source files, class files, or reflectively at run time.
  - At that moment, XML was used as standard code configuration mechanism for different type of applications.

Java Annotations...(3)

- Annotations are useful to provide
  - Information to the compiler
  - Documentation →Used to measure quality in applications such as findbugs or generate reports automatically (e.g JIRA)
  - Code Generation → Generate XML files automatically
  - Runtime Processing → Unit testing, dependency injection (Spring)

Java Annotations...(4)

- Basics of Annotations
  - With any annotation there are two aspects to consider
    - Annotation itself
    - Annotation type
  - Every annotation belongs to an annotation type.
  - An annotation type is used to define an annotation.
- Steps to define an Annotation Type
  - We attach '@' just before interface keyword.
  - Methods will not have parameters.
  - Methods will not have throws clause.
  - Method return types are restricted to primitives, String, Class, enums, annotations, and arrays of the preceding types.
  - We can assign a default value to method.

Java Annotations...(5)

- Built-In Annotations
  - Java has two types of built in annotations
    - Annotations that are applied to java code
    - Annotations that are applied to other annotations.

  - Built-in Java annotations that are applied to Java code are
    - @Override
    - @SuppressWarnings
    - @Deprecated

Java Annotations...(6)

- @Override
  - @Override annotation assures that a given subclass method is overriding the method of its parent class.
  - If this is not adhere to, compiler errors would occur.

Java Annotations...(7)

- @Deprecated
  - This annotation enables the programmer to mark a method as deprecated so the compiler provides a warning.
  - Deprecated methods run the risk of being removed in future versions of the software.

Java Annotations...(8)

- @SuppressWarnings
  - This annotation indicates that compiler warnings should be suppressed for the element annotated and all of its sub-elements.

Servlet Annotations

- Annotations does not have a direct impact on the code and is not a part of the program itself.
- If annotations are used in the servlet you do not need the deployment descriptor (web.xml).
- In order to deploy annotation based servlets tomcat7 or higher servlet container is required.
- Annotations for servlets were introduced with Servlet 3.0
- Following are the annotations we will be using in this course
  - @webservlet
  - @webListener
  - @webInitParam
  - @webFilter
- Requires the import of javax.servlet.annotation

Servlet Annotations...(2)

- Converting loginservlet.java to annotation without using web.xml
  - Create a copy of the project
  - Add the following two lines to the loginservlet.java file

```
import javax.servlet.annotation.WebServlet;

@WebServlet(urlPatterns = "/loginservlet", name = "loginservlet")
```

Servlet Annotations...(3)

- Updated file should be as follows.

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

import javax.servlet.annotation.WebServlet;

@WebServlet(urlPatterns = "/loginservlet", name = "loginservlet")
public class loginservlet extends HttpServlet {

 public void doPost(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
   String username = request.getParameter("username");
```

Servlet Annotations...(4)

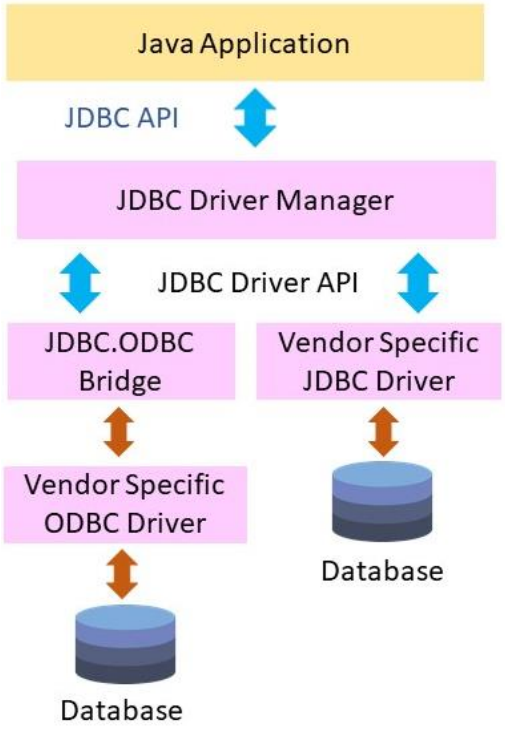
- Remove the web.xml file
- Compile the servlet file
- Restart the server and visit the web app.
- It should work as previously without any issue.
- Not the servlet is executing without the web.xml file but by using the annotation.

Practical Activity 12

- Convert the registration page to the annotations.

Java Database Connectivity (JDBC)

- What is JDBC?
  - JDBC is an API that lets you access most tabular data sources from the Java programming language
  - JDBC enables you to read from relational databases, spreadsheets etc...



Basic steps to use a database in Java

- Any database interaction with JDBC adheres to the following steps.

  - Establish a connection with the database
  - Create JDBC statements
  - Execute SQL queries and operations
  - Get the ResultSet
  - Close connections

Establishing a DB connection

- `import java.sql.*;`
- Load the vendor specific driver
  - Class.forName(“com.mysql.jdbc.driver");
  - Not required for JDBC 4.0 and above
- Make the connection

```
Connection con = DriverManager.getConnection(connectionURL);
```

- Example

```
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/users", "root", "12345678");
```

  - *root* is the username of the DBMS and *12345678* is the password. When creating the connection you have to use the username and password you used when configuring the DBMS.

Create JDBC statement(s)

- Creates a Statement object for sending SQL statements to the database

```
Statement stmt = con.createStatement() ;
```

Executing SQL Statements

```
String createUser = "Create table Users
                    (NIC CHAR(10) not null,
                    Name VARCHAR(32))";

stmt.executeUpdate(createUser);


String insertUser ="Insert into User values(567898756V,
'Sri')";

stmt.executeUpdate(insertUser);
```

Get ResultSet

```
String queryUser = "select * from User”;

ResultSet rs = stmt.executeQuery(queryUser);

while (rs.next())
{
     int age = rs.getString(“age”);
     String name = rs.getString(“NAME”);
}
```

Lets Create Simple Application with Database

- Download MySQL

https://dev.mysql.com/downloads/mysql/

- Download connector/j and copy it in Tomcat->lib directory.

https://dev.mysql.com/downloads/connector/j/

Lets Create Simple Application with Database...(2)

- Use the same login page, and save Username and Password in a database.


```
<!doctype html>
<html>
  <head>
    <title>Login Form</title>
  <head>
  <body>
    <h1>Save Login Details</h1>
    <form name="login" action="savelogin" method="post">
      <table border="0" width="20%">
          <tr>
              <td>Username: </td>
              <td><input type="text" name="username" value=""></td>
          </tr><tr>
              <td>Password: </td>
              <td><input type="password" name="password" value=""></td>
          </tr>
          <tr>
            <td><input type="reset" name="Reset" value="Reset"></td>
            <td><input type="submit" name="Submit" value="Save"></td>
        </tr>
      </table>
```

Lets Create Simple Application with Database...(3)

- Create a new database as “users” in MySQL.
- Create a new table to save Username and the Password as “logintbl”.

```
mysql> describe logintbl;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| uid   | int          | NO   | PRI | NULL    | auto_increment |
| uname | varchar(255) | YES  |     | NULL    |                |
| pass  | varchar(255) | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

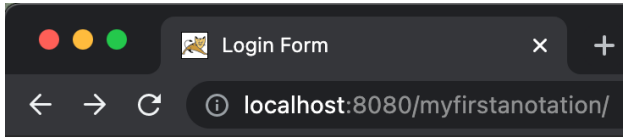
Lets Create Simple Application with Database...(4)

- Create a new servlet as “savelogin.java”

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.annotation.WebServlet;

@WebServlet(urlPatterns = "/savelogin", name = "savelogin")
public class savelogin extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
    String username = request.getParameter("username");
    String password = request.getParameter("password");
    try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/users", "root", "12345678");
            Statement stmt = con.createStatement();
            String insertUser = "INSERT INTO logintbl (uname, pass) values('"+username+"','"+password+"')";
            stmt.executeUpdate(insertUser);
            stmt.close();
            con.close();
            PrintWriter out = response.getWriter();
            out.println("<html><head><title>Data Saved</title></head>");
            out.println("<body><h2>Data Saved...</h2></body>");
            out.println("</html>");
        }
        catch (Exception e) {
            e.printStackTrace();
        }
  }
}
```

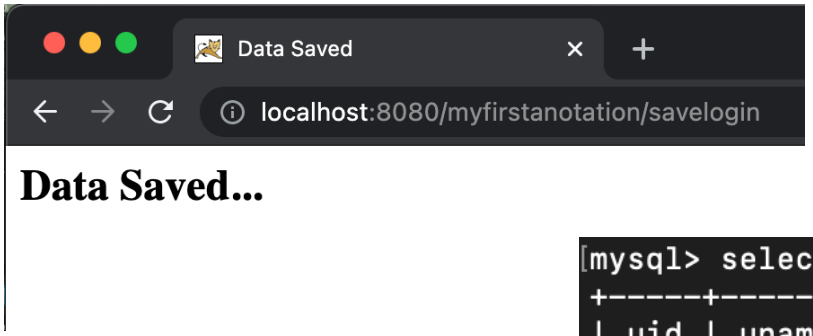
Lets Create Simple Application with Database...(5)

- Compile the servlet and restart the Tomcat server.
- Visit the web app using the web browser.
- Add a Username and Password and click save button.



Save Login Details

Username: Test_Username
Password: ••••••••••
Reset Save




```
mysql> select * from logintbl;
+-----+---------------+------------+
| uid | uname         | pass       |
+-----+---------------+------------+
|  11 | Test_Username | Password@1 |
+-----+---------------+------------+
1 row in set (0.00 sec)
```

Lets Create Simple Application with Database...(6)

- Let's view the data in the "logintbl".
- Create a servlet called "viewlogin.java"
- Update the login.html form to link the viewlogin servlet.
- Remember: when adding servlet to href, the request will send as GET.

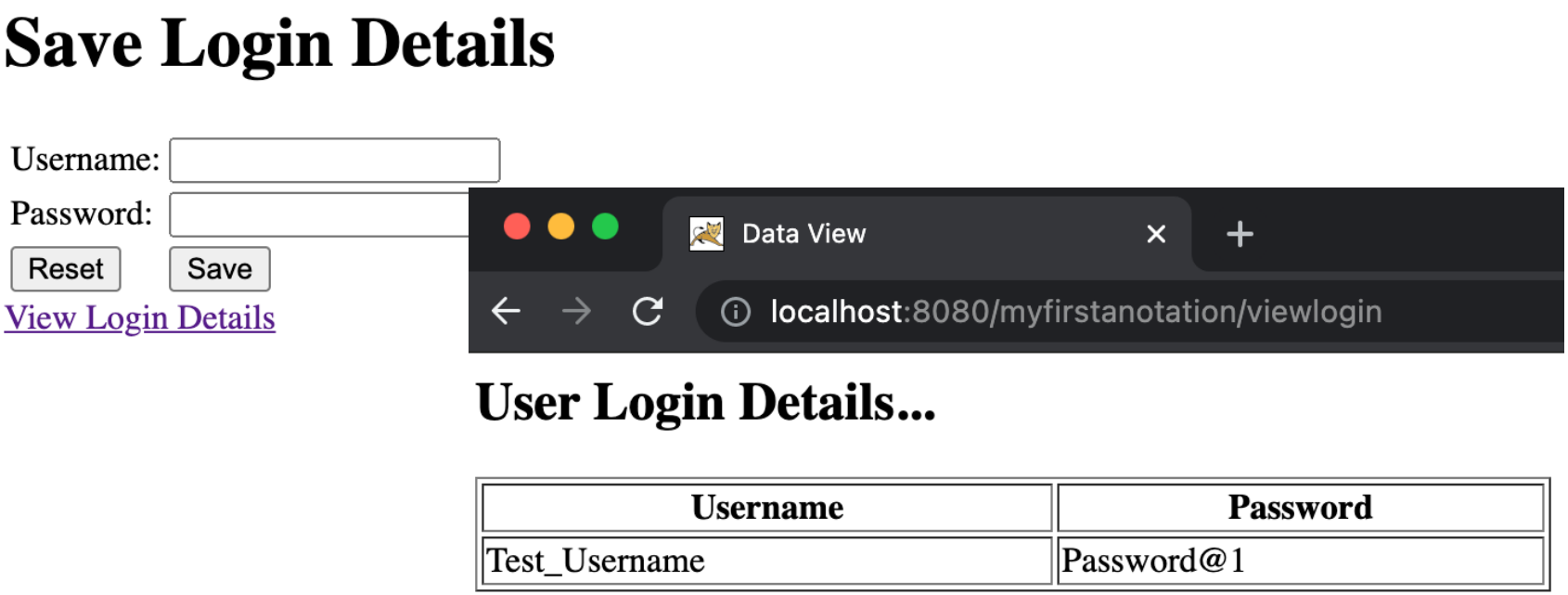
```
            <td><input type="password" name="password" value=""></td>
          </tr>
          <tr>
            <td><input type="reset" name="Reset" value="Reset"></td>
            <td><input type="submit" name="Submit" value="Save"></td>
          </tr>
        </table>
      </form>
      <a href="/myfirstanotation/viewlogin">View Login Details</a>
    <body>
</html>
```

Lets Create Simple Application with Database...(7)

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.annotation.WebServlet;

@WebServlet(urlPatterns = "/viewlogin", name = "viewlogin")
public class viewlogin extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
    try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/users", "root", "12345678");
            Statement stmt = con.createStatement();
            String viewUsers = "SELECT * FROM logintbl";
            ResultSet rs = stmt.executeQuery(viewUsers);
            PrintWriter out = response.getWriter();
            out.println("<html><head><title>Data View</title></head>");
            out.println("<body><h2>User Login Details...</h2>");
            out.println("<table border=1 width='50%'><tr><th>Username</th><th>Password</th></tr>");
            while (rs.next())
              {
                String username = rs.getString("uname");
                String password = rs.getString("pass");
                out.println("<tr><td>"+username+"</td><td>"+password+"</td>");
              }
            out.println("</table></body></html>");
            stmt.close();
            con.close();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
  }
}
```

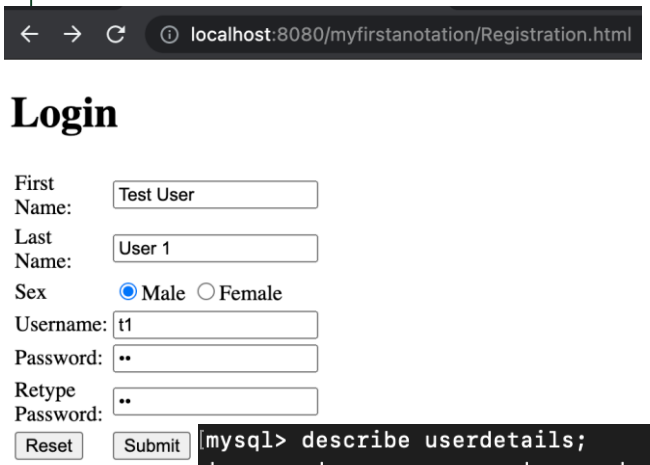
Lets Create Simple Application with Database...(8)

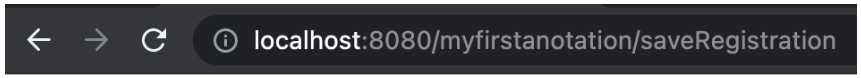
- Compile the servlet and restart the Tomcat server.
- Visit the login page.
- Click the “View Login Details” link.



Practical Activity 13

- Create a table to store user registration information in the database.
- Create a servlet application to save the user registration information in the table.



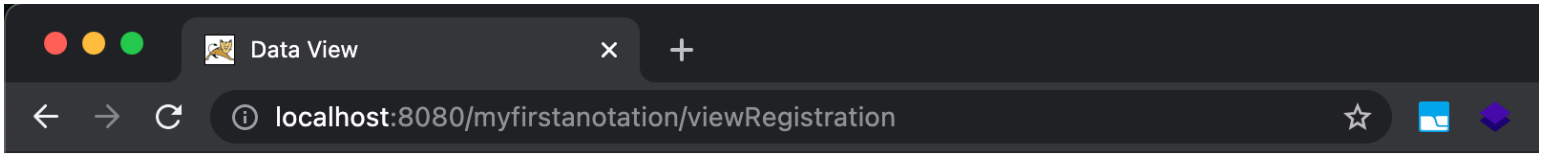



```
mysql> select * from userdetails;
+-----+-----------+--------+------+-------+------+
| uid | fname     | lname  | sex  | uname | pass |
+-----+-----------+--------+------+-------+------+
|   1 | Test User | User 1 | Male | t1    | p1   |
+-----+-----------+--------+------+-------+------+
1 row in set (0.00 sec)
```

```
mysql> describe userdetails;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| uid   | int          | NO   | PRI | NULL    | auto_increment |
| fname | varchar(255) | YES  |     | NULL    |                |
| lname | varchar(255) | YES  |     | NULL    |                |
| sex   | varchar(10)  | YES  |     | NULL    |                |
| uname | varchar(255) | YES  |     | NULL    |                |
| pass  | varchar(255) | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

Practical Activity 14

- Create a servlet application to view the user registration information.



User Registration Details...

First Name	Last Name	Gender	Username
Test User	User 1	Male	t1

Sessions

- HTTP is a stateless protocol. So how does it know who we are?

- Four ways of managing sessions
  - Cookies
  - URL Rewriting
  - HttpSession
  - Hidden fields

HttpSession

- This is an interface that provides a way to identify a user across the website. (multiple visits, pages etc.)
- The web container generates an unique session ID and identifies the user.
- How does the request response model handle this?

- Creating a session

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

- Retrieving a session

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

- Destroying a session

```
session.invalidate();
```

HttpSession...(2)

- Lets create an app with session. Use the same login html.
- Change the loginservlet.java as follows. Assign the username to a session variable called “session_username”.

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
  String username = request.getParameter("username");
  String password = request.getParameter("password");

  HttpSession session=request.getSession();
  session.setAttribute("session_username",username);

 PrintWriter out = response.getWriter();
 out.println("<html>");
 out.println("<head>");
 out.println("<title>First Servlet</title>");
 out.println("</head>");
 out.println("<body>");
 out.println("<h2>");
 out.println("Thank you for Login");
 out.println("</h2>");
 out.println("<h3>");
 out.println("<a href='sessionServlet'>Check Session</a>");
 out.println("</h3>");
 out.println("</body>");
 out.println("</html>");
```

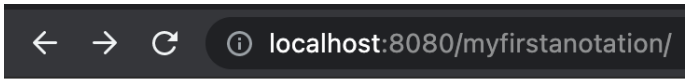
HttpSession...(3)

- Create a new servlet called “sessionServlet.java” as follows. This will obtain the *username* from the session variable and assign to *name* variable.

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

import javax.servlet.annotation.WebServlet;

@WebServlet(urlPatterns = "/sessionServlet", name = "sessionServlet")
public class sessionServlet extends HttpServlet {

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

   HttpSession session=request.getSession(false);
   String name =(String)session.getAttribute("session_username");

  PrintWriter out = response.getWriter();
  out.println("<html>");
  out.println("<head>");
  out.println("<title>Session Servlet</title>");
  out.println("</head>");
  out.println("<body>");
  out.println("<h2>");
  out.println("Username from session");
  out.println("</h2>");
  out.println("<h3>");
  out.println("Hello "+name);
  out.println("</h3>");
  out.println("</body>");
  out.println("</html>");
 }
}
```

HttpSession...(4)

- Compile the files and visit the webapp using browser.



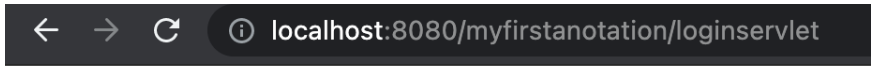
Save Login Details

Username: Tharindu

Password: •••••

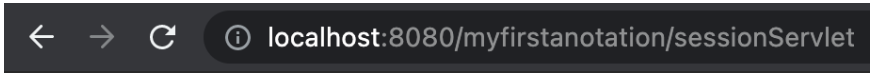
Reset Login

View Login Details



Thank you for Login

Check Session

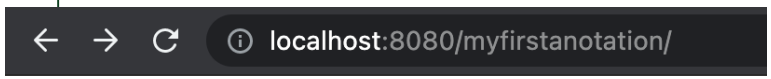


Username from session

Hello Tharindu

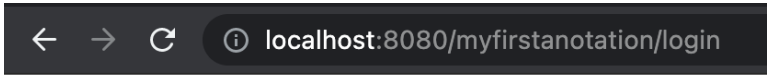
Practical Activity 15

- Create a servlet application obtain full name from the database, save the full name to a session variable.
- If the username and password are correct redirect to success page and display full name using session variables.
- If username and password incorrect, redirect to failure page.



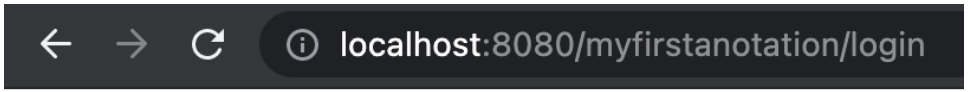
Save Login Details

Username: sri
Password: ••••••
Reset Login



Check Username and Password

Login is invalid. This will be reported!



Welcome

Hello Sri Ishwara

Cookies

- Cookies are small files that are transferred from the server to the client.
- Initial cookie is transferred with the initial response to the client request.
- Cookies are stored in a predefined location in client machine.
- Cookies have a predefined lifetime and it can be configured.
- Cookies are added to the response object using the `addCookie()` method.
- The `getCookies()` method retrieves the cookies from the request object.

Cookies...(2)

- Creating a cookie

```
Cookie cookie = new Cookie(<attribute name>, <value>);
```

- Adding to the response

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

- Retrieving from the request

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

Cookies...(3)

- Update the success.java servlet to create a cookie.

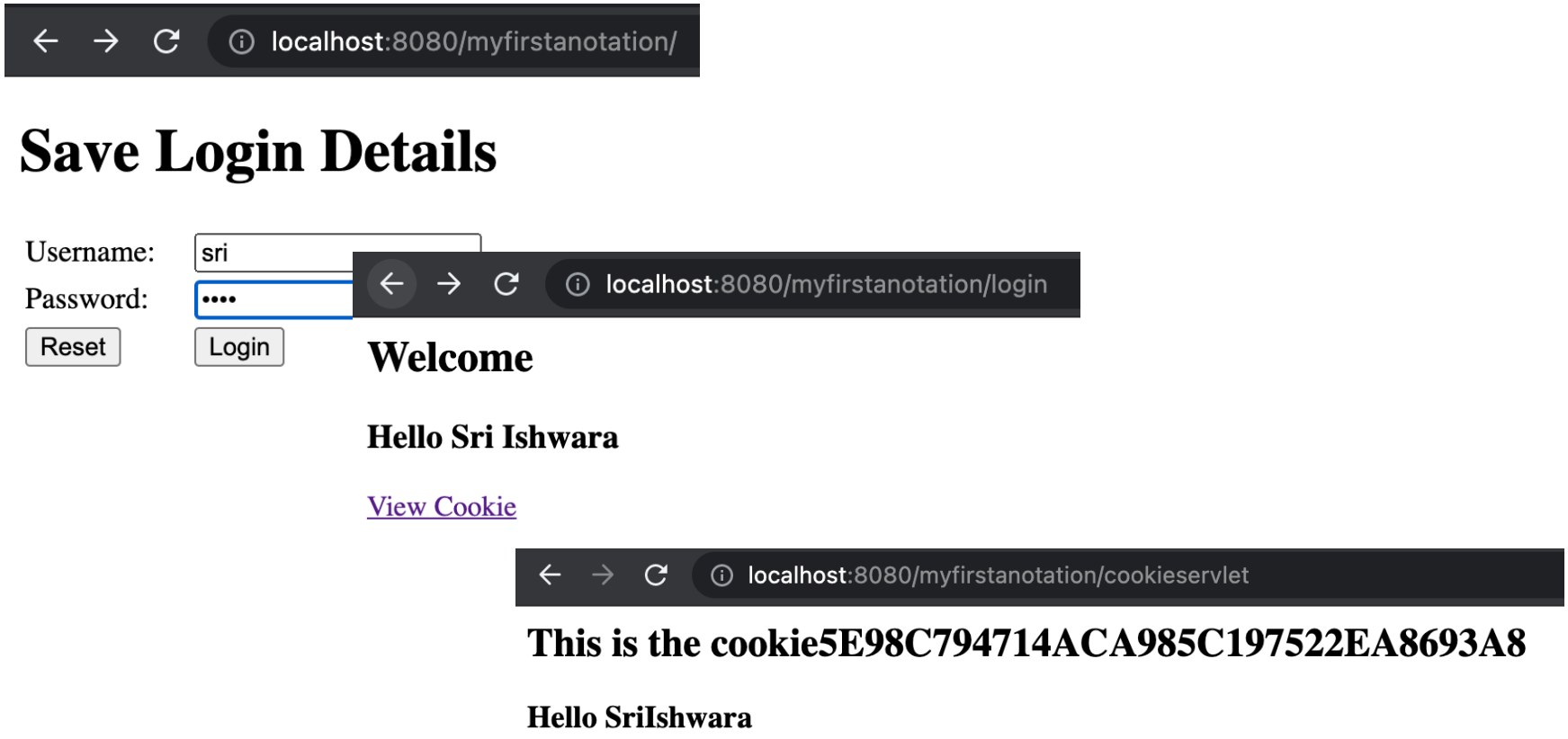
```
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
  HttpSession session=request.getSession(false);
  String fname =(String)session.getAttribute("session_fname");
  String lname =(String)session.getAttribute("session_lname");

  Cookie cookie =new Cookie("fullname_cookie",fname+lname);
  response.addCookie(cookie);

 PrintWriter out = response.getWriter();
 out.println("<html>");
 out.println("<head>");
 out.println("<title>Success Servlet</title>");
```

Cookies...(4)

- Create a new servlet called cookieservlet.java to retrieve values from the cookie.

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.annotation.WebServlet;

@WebServlet(urlPatterns = "/cookieservlet", name = "cookieservlet")
public class cookieservlet extends HttpServlet {
 public void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
   PrintWriter out = response.getWriter();
   Cookie cookie[] =request.getCookies();
   if(cookie!=null){
     String cookievalue=cookie[0].getValue();
     String fullname=cookie[1].getValue();
     out.println("<html><head><title>Cookie Servlet</title></head>");
     out.println("<body>");
     out.println("<h2>This is the cookie" + cookievalue);
     out.println("</h2><h3>Hello "+fullname);
     out.println("</h3></body></html>");
   }else{
     out.println("<html><head><title>Cookie Servlet</title></head>");
     out.println("<body><h2>No Cookie</h2></body></html>");
   }
 }
}
```

Cookies...(5)



Practical Activity 16

- Create a servlet application with a login form.
- Check the user login details with the database.
- If the username and password are correct redirect to a welcome page while saving full name of the user to a cookie.
- Display the full name of the user in the welcome page obtained from the cookie.
- If the cookie is not available redirect the user to the login form.

URL Rewriting

- This method is used when the client browser has disable the cookies.
- Disabling cookies disables the browsers ability to identify the user.
- URL Rewriting will always work

```
response.encodeURL("redirect URL");
```

Events & Listeners

- What is an Event?

- In general an event represents an occurrence of something.
- There are two types of servlet events
  - Servlet context-levels
  - Session level
- The servlet events can be of two other subcategories (4
  - Lifecycle changes (Initialization/Destroying of an application)
  - Attribute changes (modification of attributes)

Servlet Event Listeners

- Event listener listens to the events occurring in the web container.
- Listeners can be implemented by implementing the appropriate interfaces.
  - javax.servlet
  - javax.servlet.http
- Four categories of events

Servlet context lifecycle changes → javax.servlet.ServletContextListener

Servlet context attribute changes → javax.servlet.ServletAttributeListener

Session context lifecycle changes → javax.servlet.http.HttpSessionListener

Session context attribute changes → javax.servlet.http.HttpSessionAttributeListener

Setting up a Listener

- The event listeners are declared in the web.xml file directly under the `<web-app>` tag.
- Each listener declaration is within the `<listener></listener>` tags.
  - The `<listener-class>` specifies the actually class implementation of the listener.

Listener	Methods
ServletContextListener	contextDestroyed(ServletContextEvent e) contextInitialized(ServletContextEvent e)
ServletRequestAttributeListener ServletContextAttibuteListener	attributeAdded(…) attributeRemoved(….) attributeReplaced(…)
HttpSessionListener	sessionCreated(HttpSessionEvent e) sessionDestroyed(HttpSessionEvent e)
ServletRequestListener	requestDestroyed(servletRequestEvent e) requestInitialized(servletRequestEvent e)

Java Server Pages (JSP)

- What is JSP?
  - Nothing much but HTML and some java code.
  - Presents dynamic content
  - Handles the presentation logic in a MVC architecture.
  - Primary difference between servlets and jsp are its purpose of use.
    - Servlets → for business logic processing (HTML in Java)
    - JSP → presentation logic processing (Java in HTML)

Java Server Pages (JSP)...(2)

- Has three lifecycle methods as the servlets
  - jspinit()
  - _jspService()
  - jspDestroy()

- In JSP methods that start with an underscore ("_") character cannot be overridden.

- What ever is written in the JSP file, it will eventually be translated to a servlet.

JSP Elements

- There are four types of JSP elements
  - Declarations
  - Scriptlets
  - Expressions
  - Directives

JSP Declarations

- Use `<%! ... %>` to wrap declarations
- Instantiate objects/variables/methods for use in scriptlets
  - E.g. `<%! int k = 5 %>`
- The code is placed outside of the `service()` method.
- Represents reusable code.

JSP Scriptlets

- Use `<%...%>` to wrap code blocks
- Contains pure java code.
- Business logic pertaining to view goes in scriptlets. (e.g. validations)
- May need to have imports using the page directives
- Code is directly placed in the `service()` method.

JSP Expressions

- Use <%= … %> to wrap an expression
- Evaluated and the output is displayed to the client.
- Whatever the code inserted in between the expression tags should be able to evaluate to a value.

Simple JSP

- Let's update the login.java to dispatch the request to a JSP if the login credentials are correct.

```
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/users", "root", "12345678");
Statement stmt = con.createStatement();
String viewUsers = "SELECT * FROM userdetails where uname='"+username+"' AND pass='"+password+"';";
ResultSet rs = stmt.executeQuery(viewUsers);
if(rs.next()){
      String fname = rs.getString("fname");
      String lname = rs.getString("lname");
      HttpSession session=request.getSession();
      session.setAttribute("session_fname",fname);
      session.setAttribute("session_lname",lname);
      RequestDispatcher success = request.getRequestDispatcher("successlogin.jsp");
      success.forward(request,response);

}else{
 RequestDispatcher failure = request.getRequestDispatcher("failure");
 failure.forward(request,response);
}
```

Simple JSP...(2)

- Create a new JSP file as successlogin.jsp in the project directory as follows.

```
<!doctype html>
<html>
  <head>
    <title>Successful Login - JSP</title>
  <head>
  <body>
    <h1>Welcome to the JSP Homepage </h1>
    <% String fname =(String)session.getAttribute("session_fname");%>
    <% String lname =(String)session.getAttribute("session_lname");%>
    <% out.print("Welcome "+fname+" "+lname);%>
  <body>
</html
```

Simple JSP...(3)

- Compile the login.java, restart the Tomcat server and visit the web app.
- When you enter the correct login credentials, the output should be as follows.

JSP Directives

- These are preprocessing commands given to the JSP engine.
- These contains no java code or business logic
- Three types of directives (Include, Page, Tag Library)

- Use `<%@ directive …. %>` to wrap an expression
  - Include → `<%@ include file="header.jsp" %>`
  - Page → `<%@ page import ="nav.html"%>`
  - Tags → `<%@ taglib uri="TAGS" prefix=".." %>`

JSP Directives...(2)

- Page directive is applied to the current JSP page and contains a number of predefined attributes
  - Import → imported content like java.util.*;
  - isThreadSafe → true by default
  - isSession → true by default
  - isErrorPage → false by default

- The include directive is used whenever a file needs to be included in a JSP file.
  - For example the header and the footer of the webpage.
  - Included at the translation time (very important to observe this fact)

JSP Directives...(3)

- Tag libraries are used to clean up the JSP page with the addition of common custom JSP logic and HTML generation.

- Contains three basic components
  - Tag handler class
  - The mapping descriptor file
  - Taglib directive

JSP Implicit Objects

JAVA API	Implicit Object
HttpServletRequest	request
HttpServletReponse	response
HttpSession	session
ServletContext	application
ServletConfig	config
PrintWriter	out
Throwable [Exception]	Exception
PageContext	pageContext
this [Object]	page

Standard Actions

- Standard actions are available to JSP pages

- Standard actions take the form `<jsp:…>…</jsp:…>` and are pre-defined

Action	Description
forward	forward the request
useBean	work with objects [get/create]
setProperty	set the bean object properties
getProperty	get the bean object properties
include	Imports resources [html, jsp, etc]

JSF

- JavaServer Faces (JSF) - used to build component based, event oriented web interfaces. It allows to access server side logics and data. JSF is an XML document that represents formal components in a logical tree not like JSP which is an HTML embedded with server-side capabilities.

Read more: https://docs.oracle.com/javaee/6/tutorial/doc/bnaph.html

Jakarta EE

- Java 2EE, J2EE, Jakarta EE are different names for the same thing.
- Oracle has the rights for the “Java” brand.
- Therefore, Eclipse Foundation legally had to rename Java EE, and the community voted and picked **Jakarta EE**. Still JEE.

Practical Activity 17

- Create a website with header, footer and navigation bar with different JSP files.

Practical Activity 18

- Continue the website create in Activity 17 and create a user report including all the user details.