SCTR's
Pune Institute of Computer Technology, Dhankawadi, Pune 411043

Department of Computer Engineering

Lab Manual of

Web Technology Lab (310256)

For T.E. (Computer Engineering)

Academic Year: 2017-2018

Semester: VI

(Version – 1)

Preparation of Lab Manual Contributed By

Sr No	Name of experiment	Faculty name
1	Study assignment • Installation and configuration of Apache Tomcat server on Linux. • Installation and configuration of JBoss server on Linux. • Installation and configuration of GlassFish server on Linux. • Installation and configuration of WebSphere server on Linux.	Prof Ashwini Jewalikar
2	Design and develop a web application using HTML, CSS, XML from the given list: 1. Online pizza order application 2. Student information system for training and placement department 3. Leave management application 4. Blogging platform 5. Meeting room booking application 6. Exam cell automation application	Prof Preeti Jain
3	Perform fields validations using JavaScript / JQuery on previous selected application.	Prof Pavan Jaiswal
4	Add dynamic web application features to previously selected application using Servlet, JSP and backend (MySQL / MongoDB).	Prof Madhuri Wakode
5	Add dynamic web application features to previously selected application using PHP, MySQL database connectivity and AJAX controls.	Prof Parag Jambhulkar
6	Redesign, develop and deploy assignment no 4 using Strut.	Prof Pavan Jaiswal
7	Redesign, develop and deploy assignment no 5 using Angular JS.	Prof Anurag Andhare
8	Design, develop and deploy different web application (choose from above list) using EJB/CMS/JSF/Spring/Bootstrap.	Prof Pavan Jaiswal
9	Mini project	

Index

ASSIGNMENT NO: 1

TITLE	**Installation and configuration of web servers and application servers**
PROBLEM STATEMENT /DEFINITION	Study assignment • Installation and configuration of Apache Tomcat server on Linux. • Installation and configuration of JBoss server on Linux. • Installation and configuration of GlassFish server on Linux. • Installation and configuration of WebSphere server on Linux.
OBJECTIVE	• To understand commands to install mentioned application servers. • To understand difference between web server and application server
S/W PACKAGES AND HARDWARE APPARATUS USED	Apache-tomcat 8.5.24,jdk8, Jboss, Glassfish servers, mysql, PC with the configuration as Latest Version of 64 bit Operating Systems, Open Source Fedora-GHz. 8 G.B. RAM, 500 G.B. HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	1. https://tutorialforlinux.com/how-to-install-apache-tomcat-java-ee-server-on-linux-distributions-easy-guides/ 2. https://www.tutorialspoint.com/articles/tag/apache-tomcat-server 3. *www.w3schools.com*
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	• Title • Problem Definition • Objectives • Theory • Installation steps of all the servers • Test cases • Output • Conclusion

Web Server and Application server

Most of the times these terms Web Server and Application server are used interchangeably.

Following are some of the key differences in features of Web Server and Application Server:

- Web Server is designed to serve HTTP Content. App Server can also serve HTTP Content but is not limited to just HTTP. It can be provided other protocol support such as RMI/RPC
- Web Server is mostly designed to serve static content, though most Web Servers have plugins to support scripting languages like Perl, PHP, ASP, JSP etc. through which these servers can generate dynamic HTTP content.
- Most of the application servers have Web Server as integral part of them, that means App Server can do whatever Web Server is capable of. Additionally App Server have components and features to support Application level services such as Connection Pooling, Object Pooling, Transaction Support, Messaging services etc.
- As web servers are well suited for static content and app servers for dynamic content, most of the production environments have web server acting as reverse proxy to app server. That means while servicing a page request, static contents (such as images/Static HTML) are served by web server that interprets the request. Using some kind of filtering technique (mostly extension of requested resource) web server identifies dynamic content request and transparently forwards to app server

Components of Tomcat

1. Catalina : It is the Servlet Container of Tomcat.
2. Coyote : Coyote acts as a connector and supports HTTP 1.1
3. Jasper : It is the Tomcat's JSP Engine.
4. Cluster : A component for load balancing to manage large applications.
5. High availability : A Tomcat component to schedule system upgrades and changes without affecting live environment.
6. Web Application : Manage Sessions, Support deployment across different environments.

This article will walk you throughout the process of installing Apache Tomcat 8 (i.e. 8.5.14) on Linux systems, which includes RHEL, CentOS, Fedora, Debian, Ubuntu, etc.

Step 1: Installing Java 8

1. Before installing Tomcat make sure you have the latest version of Java Development Kit (JDK) installed and configured on the system. It is preferred to use oracle Java.
To install latest Oracle Java JDK (jdk-8u131) on Linux, you may like to refer our recent posts on Oracle jdk/jre/jar

Step 2: Download and Install Apache Tomcat 8

2. Once latest Java installed and configured correctly on the system, we will move forward to download and install latest stable version of Tomcat 8

Steps for Tomcat installation

1.Download tomcat tar file.

2.create tomcat installtion directory anywhere using command mkdir command.

e.g. mkdir /opt/tomcat_installation

3.cp apache-tomcat-{version}.tar.gz /opt/tomcat_installation

4.cd /opt/tomcat_installation

5.tar -xvf apache-tomcat-{version}.tar.gz

6.It will extract apache-tomcat-{version}.tar.gz

7.It will also include bin directory where there are binaries for tomcat.

8.cd bin.

9../startup.sh

It will show Tomcat started.

10.Now copy html file that you want to host on tomcat server into

tomcat_installation/webapps/

11.Open browser type http://localhost:8080/hello. if hello.html is copied in step number 10

in /opt/tomcat_installation/ webapps/ROOT/

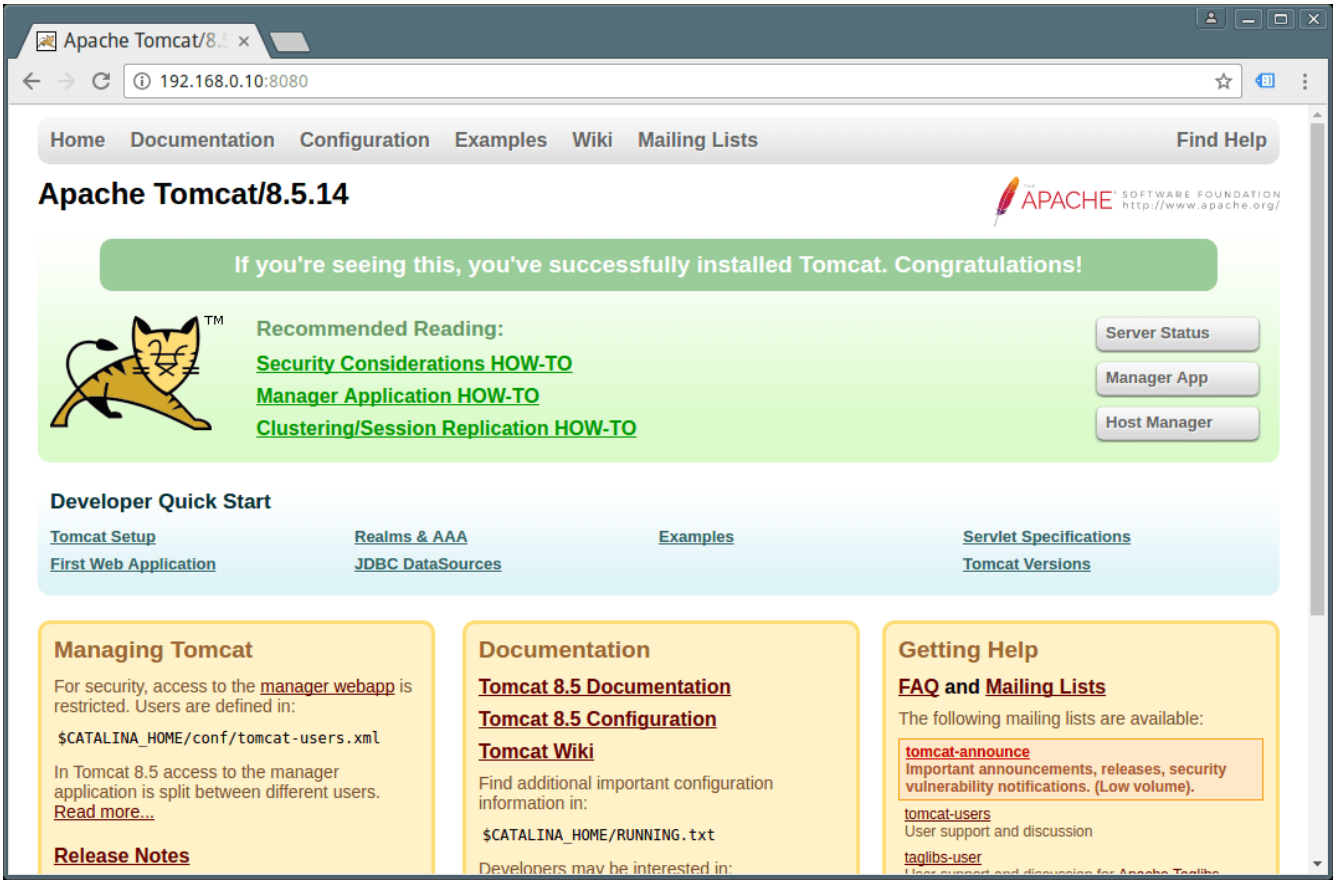
12.Create lib directory using mkdir command inside tomcat_installation/webapps/

13.Now in order to host jsp pages which will connect to database we have to copy jstl-1.2.jar and mysql-connector.jar to tomcat_installation/webaps/ftp://192.168.4.87/pub.

14.shutdown tomcat server using ./shutdown.sh command from bin directory.

15.cp jsp files tomcat_installation/webapps/

Once ‘Tomcat Started‘, you can point your browser to http://127.0.0.1:8080 and you should see something as:



Getting started with JBoss AS 7 in Fedora

From a terminal, install JBoss AS 7 using dnf or yum:

sudo dnf -y install jboss-as
sudo yum -y install jboss-as

Start the JBoss AS 7 system service:

sudo systemctl start jboss-as.service

Connect to the JBoss AS 7 management console (for the system instance):

sudo -u jboss-as sh -c "jboss-cli -c"

When you connect to the management console, the server will send a secret key challenge to the client. The client can only pass the challenge if it has physical direct access to the file system and the same permissions as the user running the server. Otherwise, you'd need to create and use a proper management user.

Stop the JBoss AS 7 system service:

sudo systemctl stop jboss-as.service

Create a user instance of JBoss AS 7:

jboss-as-cp -l $HOME/jboss-as-user-instance

Start the JBoss AS 7 user instance:

$HOME/jboss-as-user-instance/bin/standalone.sh

standalone.sh is a script generated by jboss-as-cp that effectively runs this command

JBOSS_BASE_DIR=$HOME/jboss-as-user-instance /usr/share/jboss-as/bin/standalone.sh

-c standalone-web.xml

Connect to the JBoss AS 7 management console (for the user instance):

jboss-cli -c

GLASSFISH:

And between the best GlassFish Advantages you find: it's Free, No vendor Lock-in and Easy Adaptability (Supports MySQL, .NET, Eclipse and NetBeans Integration).

GlassFish is the Java EE reference implementation

1. Support Latest version of the JEE 7 Specification
2. Commercial-Professional Support Available

Download GlassFish 4 JEE 7 App Server for Linux:

Here Get GlassFish 4 latest-glassfish-unix.sh

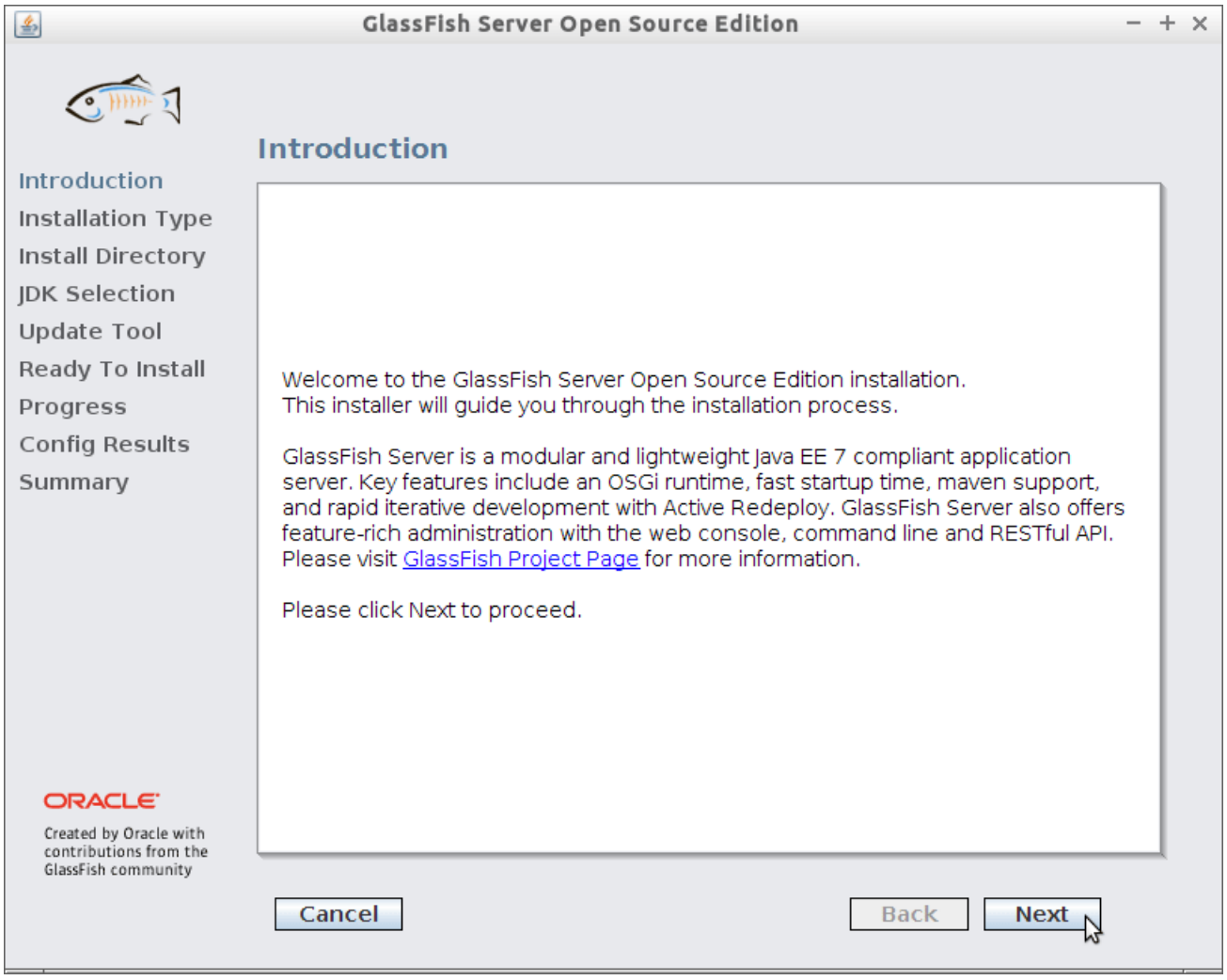
Start GlassFish 4 Installation

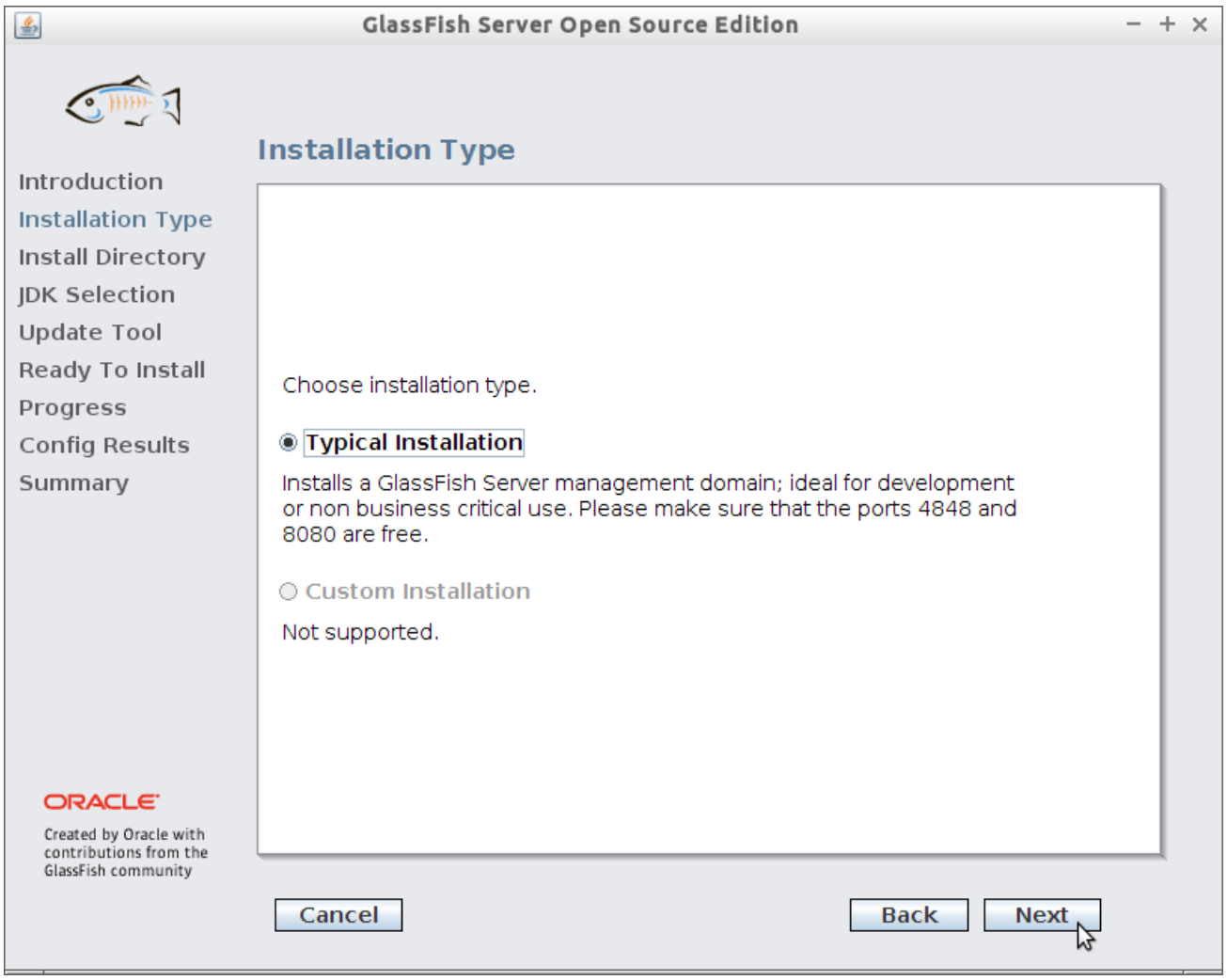
sudo su

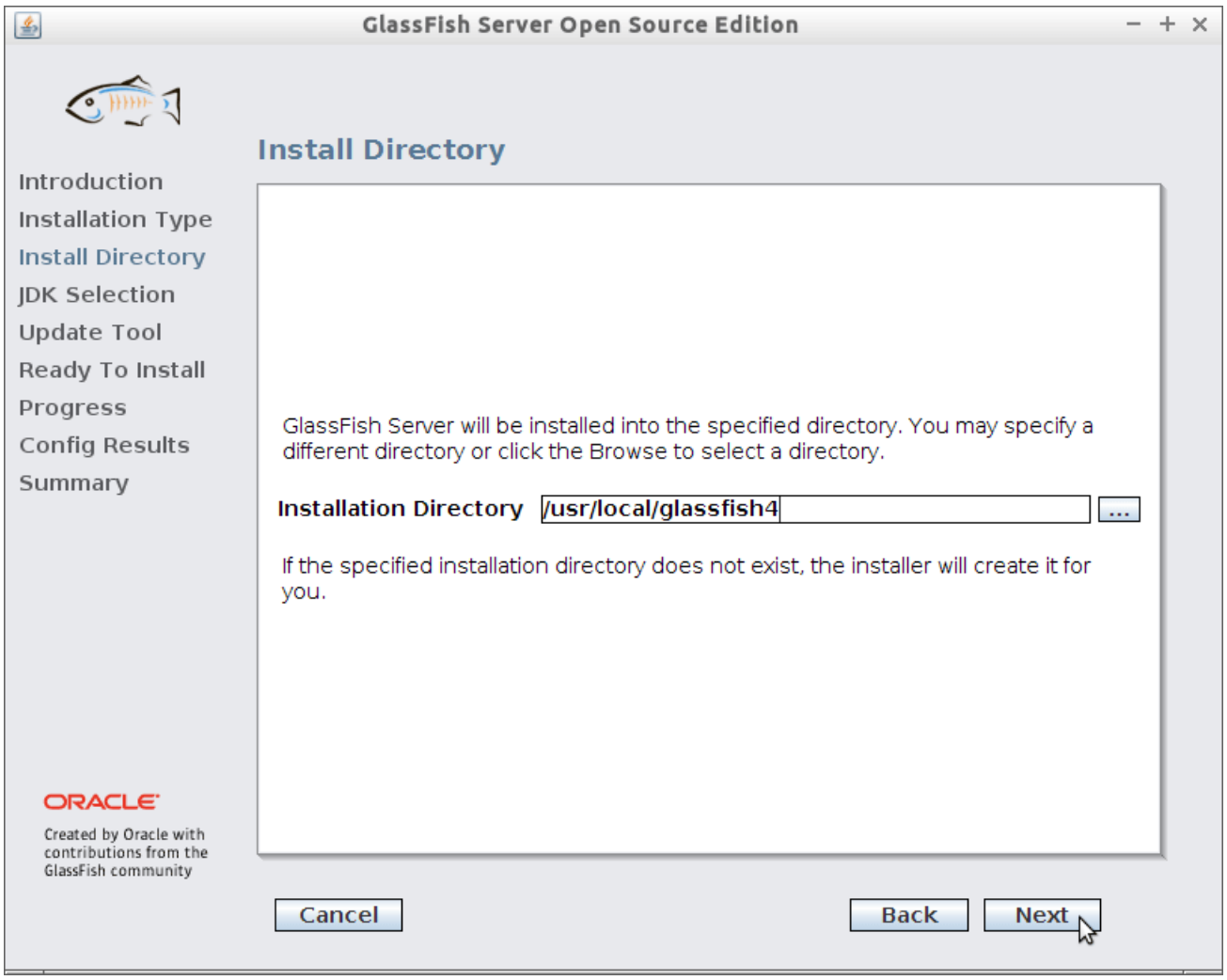
If Got "User is Not in Sudoers file" then Look: Solution

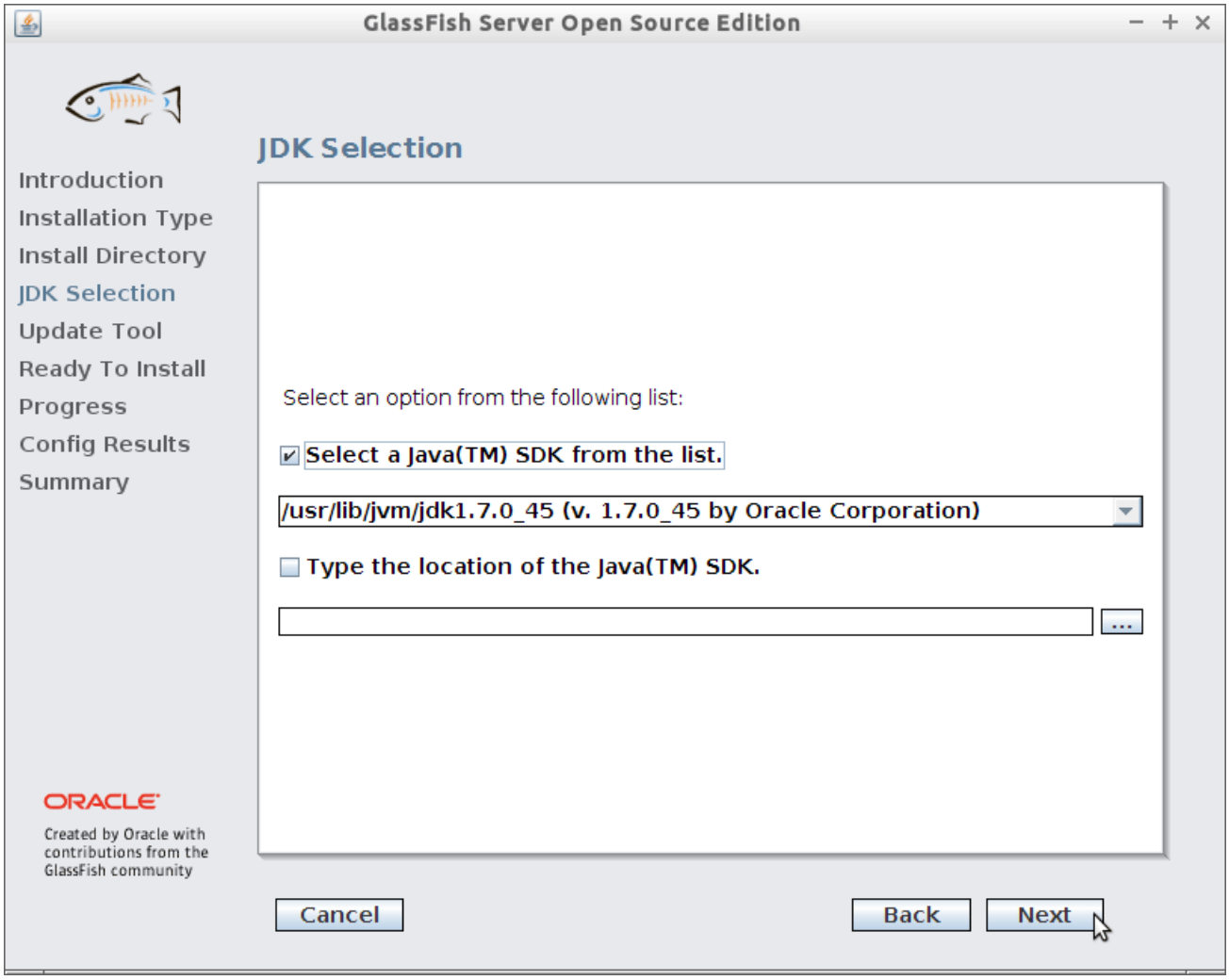
cd </path/2>/latest-glassfish-unix.sh
chmod +x ./latest-glassfish-unix.sh

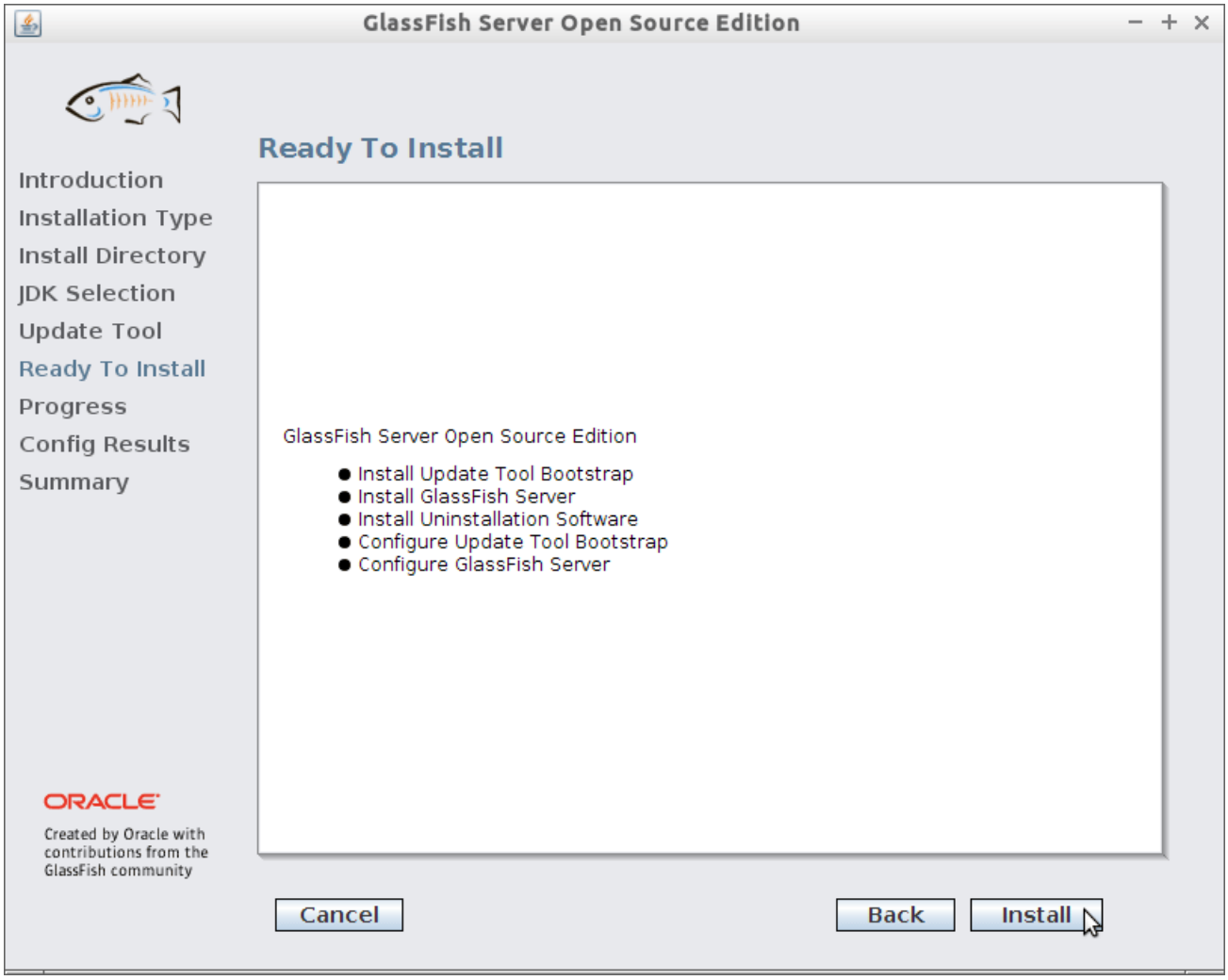
Installing GlassFish 4 JEE 7 App Server

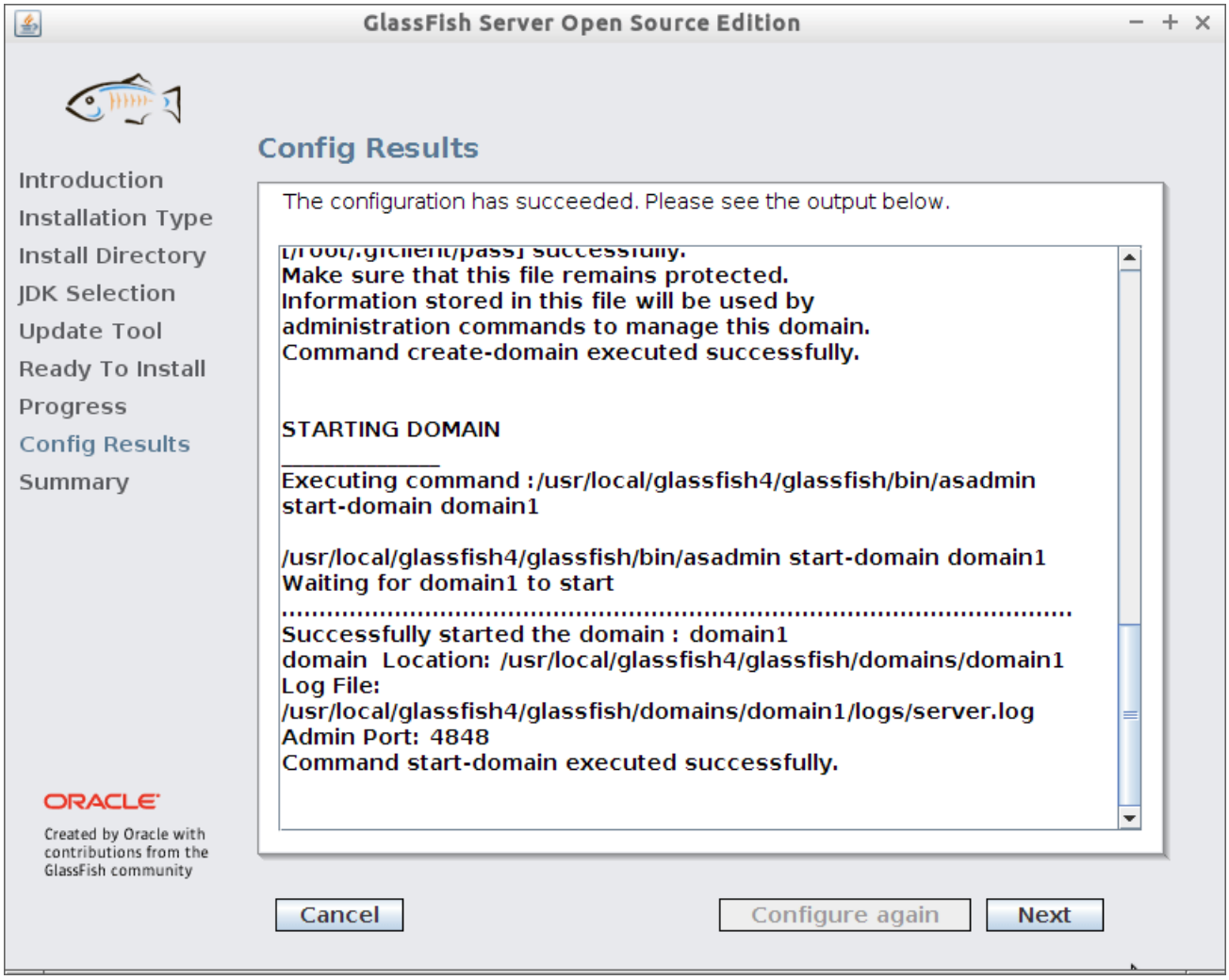


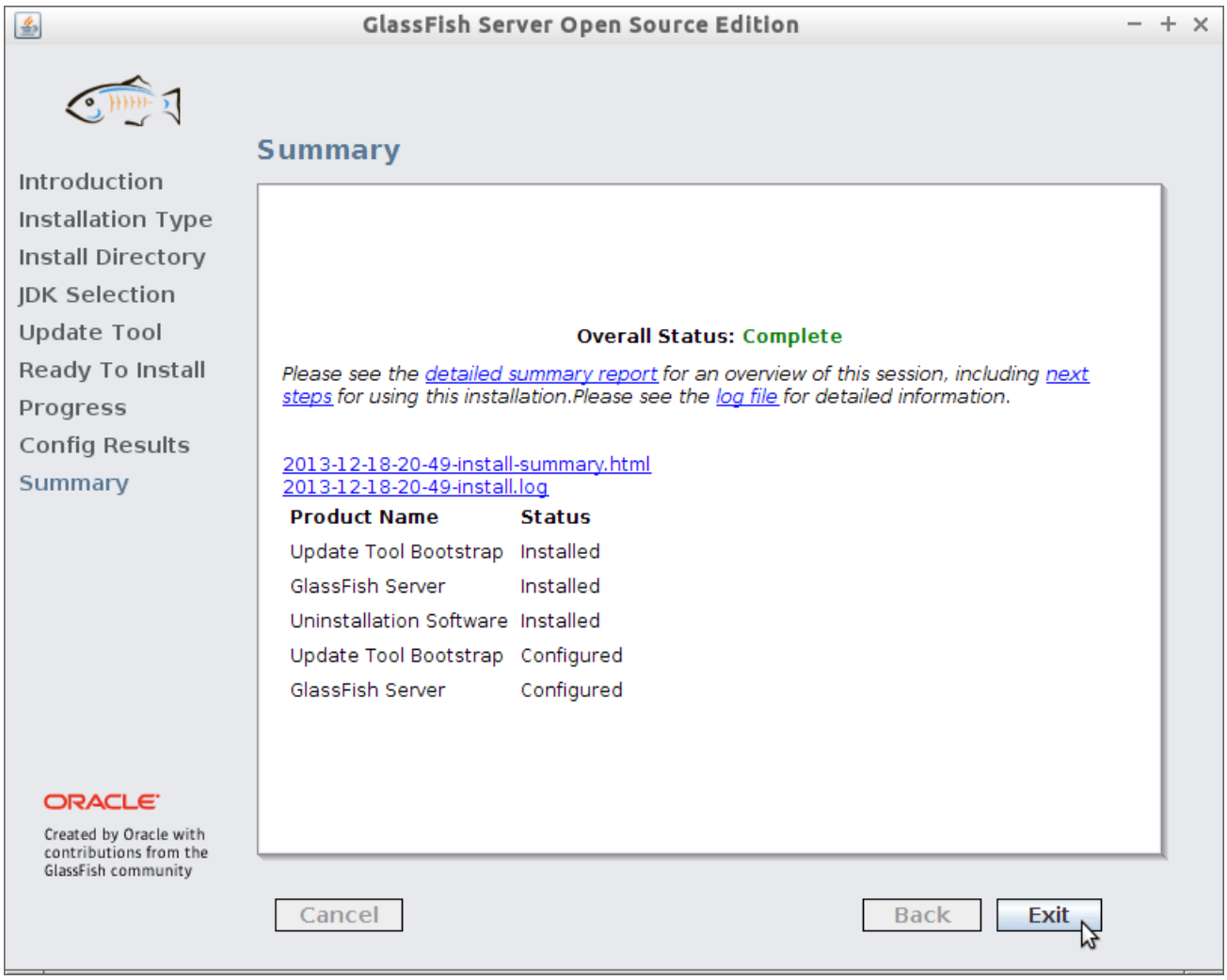

GlassFish Server Open Source Edition
Installation Type
Introduction
Installation Type
Install Directory
JDK Selection
Update Tool
Ready To Install
Progress
Config Results
Summary
Choose installation type.
Typical Installation
Installs a GlassFish Server management domain; ideal for development or non business critical use. Please make sure that the ports 4848 and 8080 are free.
Custom Installation
Not supported.
ORACLE
Created by Oracle with contributions from the GlassFish community
Cancel
Back
Next


GlassFish Server Open Source Edition
Install Directory
Introduction
Installation Type
Install Directory
JDK Selection
Update Tool
Ready To Install
Progress
Config Results
Summary
GlassFish Server will be installed into the specified directory. You may specify a different directory or click the Browse to select a directory.
Installation Directory
/usr/local/glassfish4
...
If the specified installation directory does not exist, the installer will create it for you.
ORACLE
Created by Oracle with contributions from the GlassFish community
Cancel
Back
Next


GlassFish Server Open Source Edition
JDK Selection
Introduction
Installation Type
Install Directory
JDK Selection
Update Tool
Ready To Install
Progress
Config Results
Summary
Select an option from the following list:
Select a Java(TM) SDK from the list.
/usr/lib/jvm/jdk1.7.0_45 (v. 1.7.0_45 by Oracle Corporation)
Type the location of the Java(TM) SDK.
...
ORACLE
Created by Oracle with contributions from the GlassFish community
Cancel
Back
Next


GlassFish Server Open Source Edition
Ready To Install
Introduction
Installation Type
Install Directory
JDK Selection
Update Tool
Ready To Install
Progress
Config Results
Summary
GlassFish Server Open Source Edition
Install Update Tool Bootstrap
Install GlassFish Server
Install Uninstallation Software
Configure Update Tool Bootstrap
Configure GlassFish Server
ORACLE
Created by Oracle with contributions from the GlassFish community
Cancel
Back
Install


GlassFish Server Open Source Edition
Config Results
Introduction
Installation Type
Install Directory
JDK Selection
Update Tool
Ready To Install
Progress
Config Results
Summary
The configuration has succeeded. Please see the output below.
[/root/.gfclient/pass] successfully.
Make sure that this file remains protected.
Information stored in this file will be used by
administration commands to manage this domain.
Command create-domain executed successfully.
STARTING DOMAIN
Executing command :/usr/local/glassfish4/glassfish/bin/asadmin
start-domain domain1
/usr/local/glassfish4/glassfish/bin/asadmin start-domain domain1
Waiting for domain1 to start
Successfully started the domain : domain1
domain Location: /usr/local/glassfish4/glassfish/domains/domain1
Log File:
/usr/local/glassfish4/glassfish/domains/domain1/logs/server.log
Admin Port: 4848
Command start-domain executed successfully.
ORACLE
Created by Oracle with contributions from the GlassFish community
Cancel
Configure again
Next



Access Glassfish Admin Console on Browser: http://localhost:4848

References:

1. https://tutorialforlinux.com/how-to-install-apache-tomcat-java-ee-server-on-linux-distributions-easy-guides/
2. https://www.tutorialspoint.com/articles/tag/apache-tomcat-server
3. *www.w3schools.com*

Oral Questions:

1. How to start / stop Apache tomcat?

2. What is the default port for HTTP and HTTPs
3. List the important configuration file name of tomcat.
4. How to check version of running Apache tomcat?
5. How to know if web server is running?
6. How to install Apache web server?
7. Differentiate between web server and application server.
8. What is Glassfish application server?
9. How to start and stop the Default Domain?
10. How to start administration console in GlassFish?

ASSIGNMENT NO: 2

TITLE	***Design with CSS, HTML , XML***
PROBLEM STATEMENT /DEFINITION	Design & develop a web application using HTML , CSS , XML from the given list : 1. Online pizza order application 2. Student information system for training & placement department 3. Leave management application 4. Blogging platform 5. Meeting room booking application 6. Exam cell automation application
OBJECTIVE	• To develop web pages using HTML • To optimize page styles & layout with CSS • To distinguish between HTML & XML
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating System open source Fedora 20 or higher equivalent or Windows Networked computer with internet access Editor : IDE , Eclipse or any simple equivalent editor (i.e. text based & WYSIWYG based) Web browser / Internet explorer 7
REFERENCES	1. Web enabled commercial application development using HTML , DHTML , JavaScript , Perl CGI by Ivan Bayross 2. Musiciano C. Kennedy B., ' HTML & XHTML ' , 5th or higher edition , O'Reilly / SPD Publications , ISBN B1 – 7366 – 517 – 1 3. Learning XML by Erik T. Ray , O'reilly 4. Internet & World Wide Web , How to Programe , 3rd or higher edition , H. M. Deitel , P. J. Deitel , A. B. Goldberg , Pearson education 5. McKinnon A., McKinnon L., ' XML ' , Vikas Publishing House , 2004 , ISBN 981 – 254 – 299 – X 6. https://www.w3schools.com/xml/
STEPS	Refer to steps below
INSTRUCTIONS FOR WRITING JOURNAL	• Title • Problem Statement • Description (HTML vs XML & various tags in it CSS) • Block diagrams (design part)

	• Troubleshooting (if any) • Conclusion • References

Problem Statement:

Design & develop a web application using HTML , CSS , XML from the given list :

1. Online pizza order application
2. Student information system for training & placement department
3. Leave management application
4. Blogging platform
5. Meeting room booking application
6. Exam cell automation application

Objectives:

1) To develop web pages using HTML
2) To optimize page styles & layout with CSS
3) To distinguish between HTML & XML

Outcomes:

1) Define the key terms relevant to coding HTML and CSS, including: tag, attribute, element, entity, selector, header, table, ordered list, unordered list, link, heading, paragraph ; et cetra .
2) Describe the function of common tags & styles in short snippets of code & predict the output of the same .
3) They will be able to create well–formed & valid XML documents, write DTDs & Schemas & deliver XML documents over the Web using different style sheets
4) Define & compare the concepts of multimedia , hypermedia & hypertext .

Theory:

Introduction:

Files that travel across the largest network in the world , the Internet , & carry information from ‘ Server ’ to ‘ Client ’ that requested them are called ‘ **Web pages** / HTML documents ’ . Individual who develops these web pages is called ‘ **Web Developer** ’ . Web Pages are created using HTML syntax . The organization of web pages into directories & files stored on the HDD of a computer is called ‘ **Web Site** ’ creation . As studied in previous assignment , the Server Computer runs special software called ‘ **Web Server** ’ software that allows :

- Web Site Management
- Accept a client’s request for information
- Respond to a client’s request by providing the page with the required information

Computers that offer the facility to read information stored in web pages are called ‘ **Web Clients** ’ . Web Clients run special software called a ‘ **Browser** ’ that allows to :

- Connect to an appropriate Server

- Query the Server for the information to be read
- Provides an interface to read the information returned by the Server

Following points emphasize the requirements of a good web site :

***First Impression* –** Did the initial page grab the attention ?

***Interface Design* –** Is the menu interface interactive enough & visually interesting ?

***Corporate Mildew* –** Is the site trapped in a web of corporate look , feel & canned marketing speak ?

***Coriolis Effect* –** Does the site generate enough currents of interest based on design & content for the user to comeback ?

HTML :

The language used to develop web pages is called **H**yper**T**ext **M**arkup **L**anguage which is interpreted by a Browser . HTML is a set of special codes that can be embedded in text to add formatting & linking information . **HTML Tags** are instructions that are embedded directly into text of document . It is a signal to a browser that it should do something other than just throw text up on the screen . HTML tags can be of two types :

Paired Tags &

Singular Tags

Some HTML tags require additional information to be supplied to them that are known as *Attributes* of a tag . Attribute(s) are written immediately following the tag , separated by a *space* . The creation of textual content of Web Site is done in any editor viz; Notepad / Eclipse / IDE ; et cetra & saved as *filename.**htm / .html*** file .

Tag	Name	Example
<!--	comment	<!--This can be viewed in the HTML part of a document-->
<a –	anchor	 Visit PICT
	bold	Example
<big>	big (text)	<big> Example </big>
<body>	body of HTML document	<body> The content of your HTML page </body>
 	line break	The contents of your page The contents of your page
	center	This will center your contents
<dd> <dl> <dt>	definition description definition list definition term	<dl> <dt> definition term </dt> <dd> definition of the term </dd> </dl>
	Emphasis	This is an example of using emphasis tag
<embed>	embed object	<embed src = “your file” width = “100%” height = “60” align =

		“center”>
	Font	 Example
<h1> <h2> <h3> <h4> <h5> <h6>	heading 1 heading 2 heading 3 heading 4 heading 5 heading 6	Heading 1 Example Heading 2 Example Heading 3 Example Heading 4 Example Heading 5 Example Heading 6 Example
head	heading of HTML document	<head>Contains elements describing the document</head>
<hr>	horizontal rule	<hr width="50%" size="3" noshade />
<html>	hypertext markup language	<html> <head> <meta> <title>Title of your web page</title> </head> <body>HTML web page contents </body> </html>
	Italic	Example
<img>	Image	
<input>	input field	**Ex:** <form method=post action="/cgibin/example.cgi"> <table border="0" cellspacing="0" cellpadding="2"><tr><td bgcolor="#8463ff"><input type="text" size="10" maxlength="30"></td><td bgcolor="#8463ff" valign="Middle"> <input type="image" name="submit" src="yourimage.gif"></td></tr> </table> </form> **Ex:** <form method=post action="/cgibin/example.cgi"> Select an option: <input type="radio" name="option"> Option 1 <input type="radio" name="option" checked> Option 2 <input type="radio" name="option"> Option 3 Select an option: <input type="checkbox" name="selection"> Selection 1 <input type="checkbox" name="selection" checked>Selection 2 <input type="checkbox" name="selection"> Selection 3 <input type="Submit" value="Submit"> <input type="Reset" value="Clear"> </form>
<menu>	Menu	<menu> <li type="disc">List item 1

 	list item ordered list	<li type="circle">List item 2 <li type="square">List item 3 </MENU> <ol type="i"> - List item 1 - List item 2
<link>	Link	<head> <link rel="stylesheet" type="text/css" href="style.css" /> </head>
<marquee>	scrolling text	<marquee bgcolor="#cccccc" loop="-1" scrollamount="2" width="100%">Example Marquee</marquee>
<meta>	meta	<meta http-equiv="Pragma" content="nocache">
<option>	listbox option	<form method=post action="/cgibin/example.cgi"> Select an option: <select> <option>option 1</option> <option selected>option 2</option> </select> </form>
<p>	paragraph	<p align="center"> This is an example displaying the use of the paragraph tag
<small>	small (text)	Example
<strike>	deleted text	<strike>Example</strike>
<table>	table	<table cellpadding="2" cellspacing="2" width="100%"> <tr> <th>Column 1</th> <td bgcolor="#cccccc">Column 1</td> <td bgcolor="#cccccc">Column 2</td> </tr> <tr> <td>Row 2</td> <td>Row 2</td> </tr> </table>
<title>	document title	<title>Title of your HTML page</title>
<u>	underline	<u>Example</u>
<tt>	teletype	<tt>Example</tt>
<style>	CSS	

DHTML (Dynamic HTML):

It combines HTML with Cascading Style Sheets (CSS) & Scripting Languages . HTML specifies a web page's element lie table , frame , paragraph , bulleted list ; etc. CSS can be used to determine an element's size , color , position & number of other features .

CSS (Cascading Style Sheets) :

Style Sheets are powerful mechanism for adding styles to Web documents that enforces standards & uniformity throughout a web site & provide numerous attributes to create dynamic effects . Style information can be associated with the web page in several ways :

- by embedding the style information directly through a STYLE attribute
- by embedding the style information directly through a < STYLE > header
- by embedding the style information directly through < LINK > element

Order of importance for adding style sheets into the document :

I. Inline styles
II. Embedded styles
III. Linked styles
IV. Imported styles
V. Default browser styles

Advantages:

- ✓ ability to make global changes to all documents from a single location
- ✓ greater author control over appearance of text & its placement on the page
- ✓ reduced clutter of multiple opening & closing tags on individual text elements
- ✓ simplified modification of page design through style editing
- ✓ eliminating the need for clumsy HTML workarounds to achieve basic layout effects
- ✓ great improvement of the design potential for HTML pages without introducing a large no. of new proprietary tags or compromising ability of other browser to effectively display the document text

XML – Nuts & Bolts:

- DTD
- XSD – eXtensible Schema Definition
- XSL – eXtensible Style Languages
- XML Linking Languages (XPath , Xlink & Xpointer)
- XML Namespaces

Advantages of Schemas:

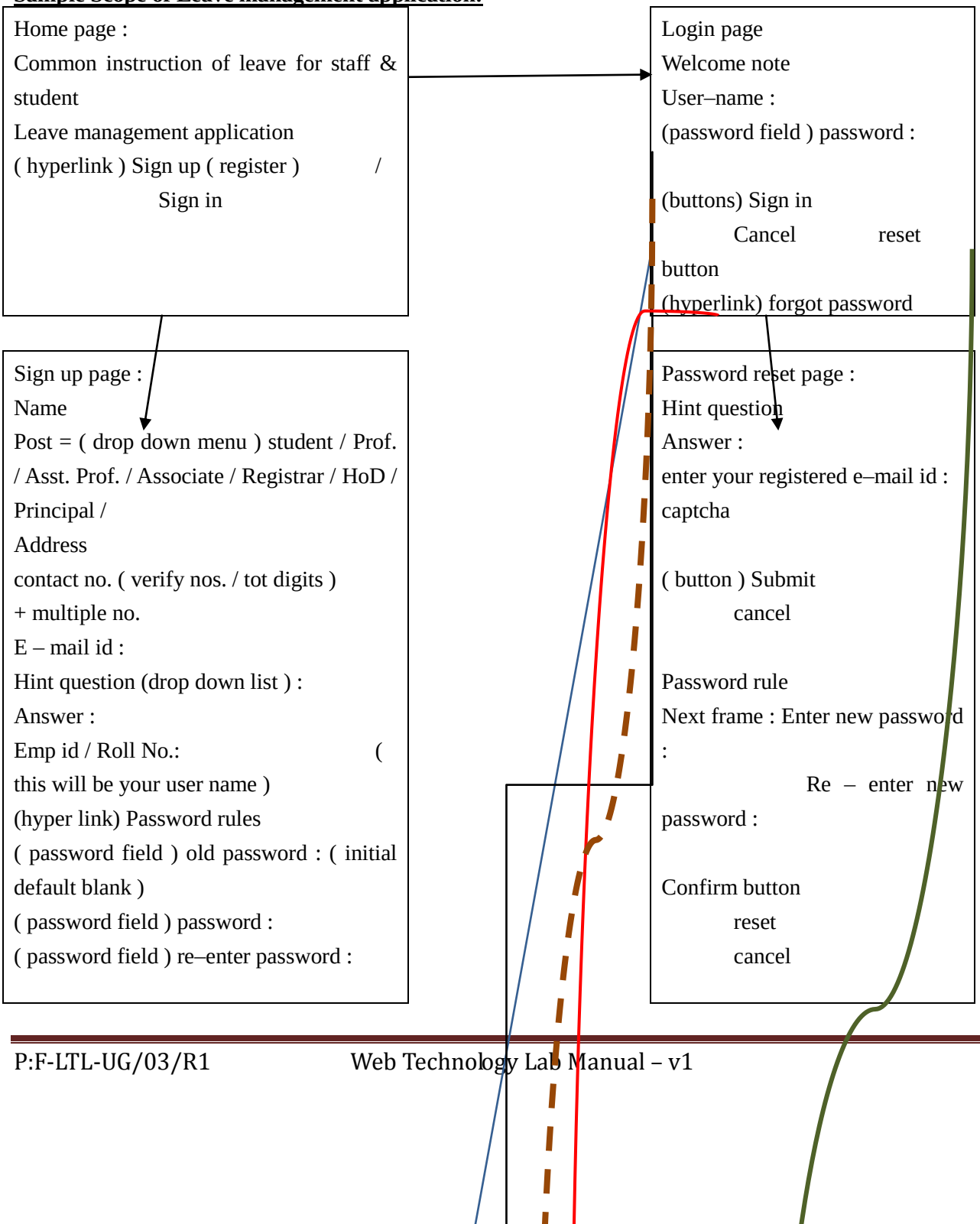
- ✓ easier to validate the correctness of data
- ✓ easier to work with data from database
- ✓ easier to define data facets & data patterns
- ✓ easier to convert data between different data types

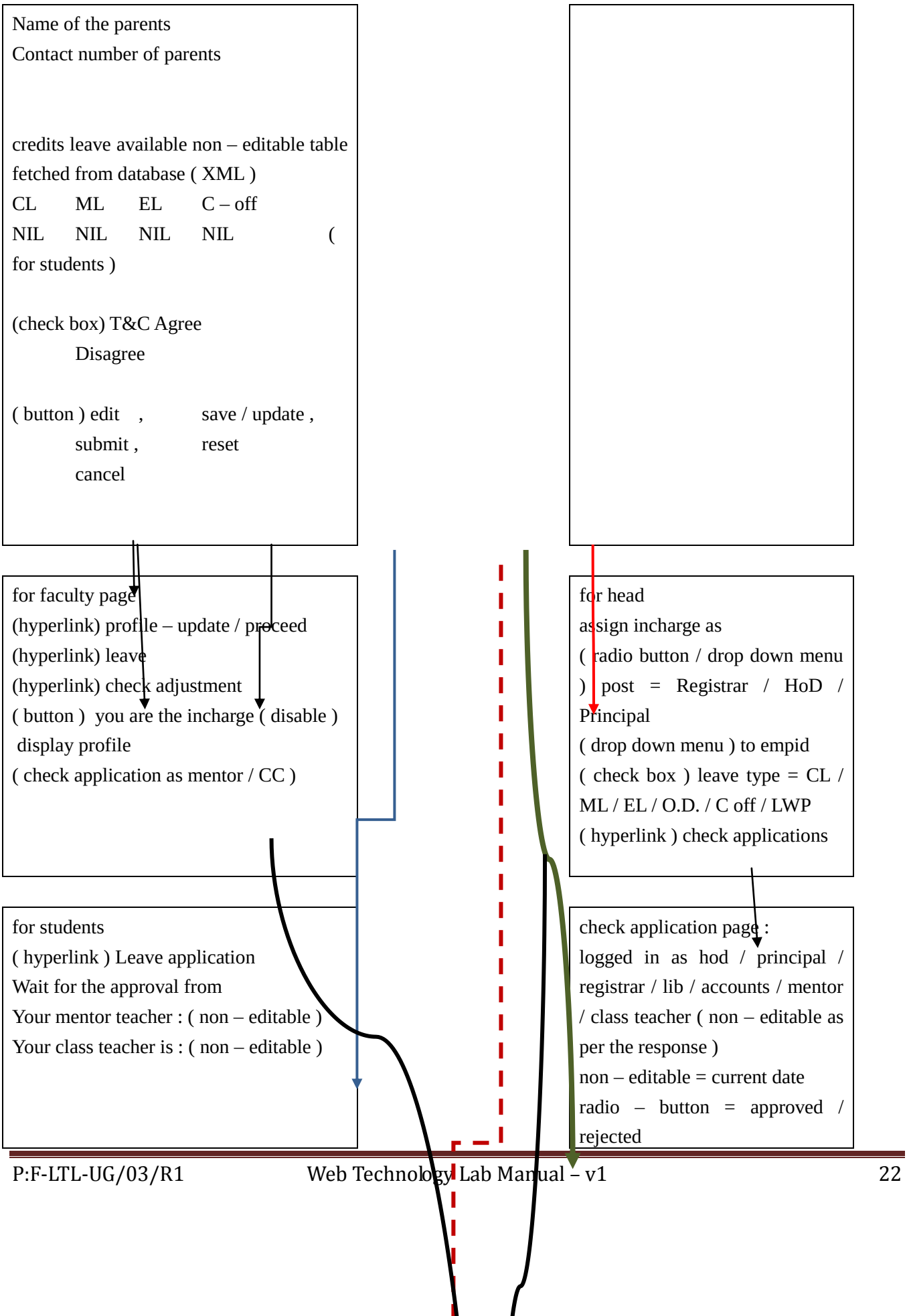
HTML vs XML

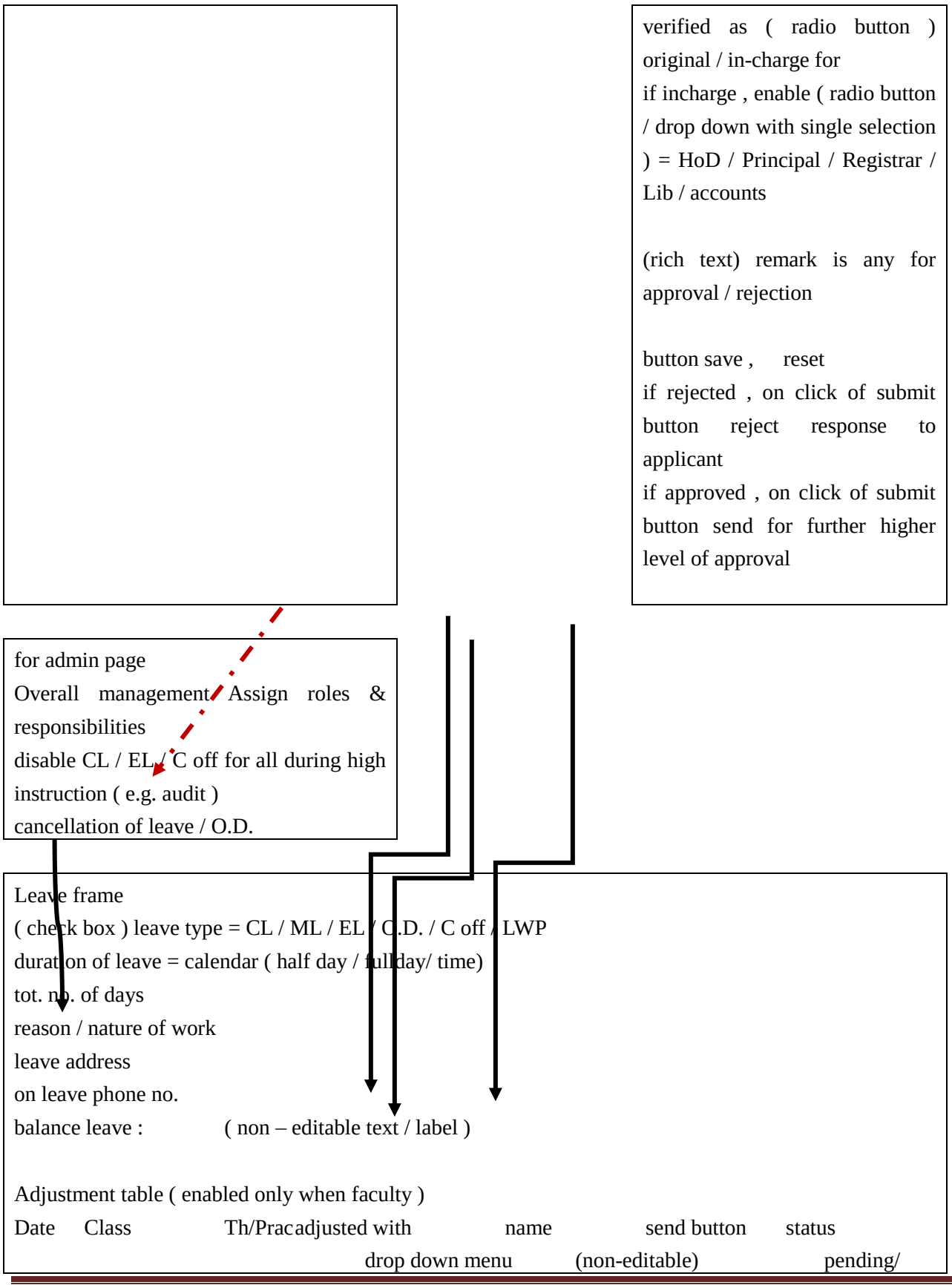
HyperText Markup Language	**Extensible Markup Language**
used to display data	used to transport & store data
it is markup language	provides framework to define markup languages
not case sensitive	case sensitive

it is presentation language	neither presentation nor programming lang.
predefined tags	custom tag
no strict rules	strict rules
Static	dynamic
does not preserve white spaces	preserve white spaces
list out the limitations	list out the limitations

Sample Scope of Leave management application:




Name of the parents
Contact number of parents
credits leave available non – editable table fetched from database (XML)
CL ML EL C – off
NIL NIL NIL NIL (
for students)
(check box) T&C Agree
Disagree
(button) edit , save / update ,
submit , reset
cancel
for faculty page
(hyperlink) profile – update / proceed
(hyperlink) leave
(hyperlink) check adjustment
(button) you are the incharge (disable)
display profile
(check application as mentor / CC)
for head
assign incharge as
(radio button / drop down menu) post = Registrar / HoD / Principal
(drop down menu) to empid
(check box) leave type = CL / ML / EL / O.D. / C off / LWP
(hyperlink) check applications
for students
(hyperlink) Leave application
Wait for the approval from
Your mentor teacher : (non – editable)
Your class teacher is : (non – editable)
check application page :
logged in as hod / principal / registrar / lib / accounts / mentor / class teacher (non – editable as per the response)
non – editable = current date
radio – button = approved / rejected


verified as (radio button) original / in-charge for
if incharge , enable (radio button / drop down with single selection) = HoD / Principal / Registrar / Lib / accounts
(rich text) remark is any for approval / rejection
button save , reset
if rejected , on click of submit button reject response to applicant
if approved , on click of submit button send for further higher level of approval
for admin page
Overall management Assign roles & responsibilities
disable CL / EL / C off for all during high instruction (e.g. audit)
cancellation of leave / O.D.
Leave frame
(check box) leave type = CL / ML / EL / C.D. / C off / LWP
duration of leave = calendar (half day / full day/ time)
tot. no. of days
reason / nature of work
leave address
on leave phone no.
balance leave : (non – editable text / label)
Adjustment table (enabled only when faculty)
Date Class Th/Prac adjusted with drop down menu name (non-editable) send button status pending/

emp id ok

changes as
per response from
adjuster

upload proofs
upload parents' signed document

(buttons) ok cancel edit cancel leave

Note :

- *on every final button pop – up for confirmation*
- *on every page there must be buttons of back , Home page & Logout & accordingly pages must be interlinked*
- *make use of CSS & XML DTD / Schema wherever applicable*

Algorithmic steps:

1. Finalize the scope of given list of web application
2. Task distribution
3. Design the respective application in lab note book & get it verified by respective lab subject teacher
4. Write the code snippet in the editor (HTML , CSS , XML) & visualize the same in browser
5. Testing

Testing:

Test id	**Test case**	**Expected o/p**	**Actual o/p**
1.	All web pages must be properly interlinked		
2.	Every page must have back button , home page button & logout with appropriate navigation		
3.	After the click of final button , pop – up confirmation is required		
4.	Web document is expected to be user –		

	friendly with formal design		
5.	Style overloading		
6.	Provide the alternate values to attributes of tags w.r.t. compatibility of browser . explorer		

References:

1. Web enabled commercial application development using HTML , DHTML , JavaScript , Perl CGI by Ivan Bayross
2. Musiciano C. Kennedy B., ' HTML & XHTML ' , 5th or higher edition , O'Reilly / SPD Publications , ISBN B1 – 7366 – 517 – 1
3. Learning XML by Erik T. Ray , O'reilly
4. Internet & World Wide Web , How to Programe , 3rd or higher edition , H. M. Deitel , P. J. Deitel , A. B. Goldberg , Pearson education
5. McKinnon A., McKinnon L., ' XML ' , Vikas Publishing House , 2004 , ISBN 981 – 254 – 299 – X

https://www.w3schools.com/xml/

Oral Questions:

1. Compare HTML with XML.
2. What is the difference between form get and form post?
3. What is the importance of the HTML DOCTYPE?
4. What is web application?
5. What is markup language?
6. What is DOM document & XPath?
7. Can we have empty XML tag?
8. Can we replace HTML with XML?

Extra Assignments for practice:

1) Create a specimen of a corporate web page . Divide the browser screen into two frames . The frame on the left will be a menu consisting of hyperlinks . Clinking on any one of these links will lead to a new page , which must open in the target frame , which is on RHS .

2) Design following frame :

Content of Frame 1	Content of Frame 3

	Content of Frame 4
Content of Frame 2	
	Content of Frame 5

Solution :

```
<html>
<head> <title>  Nested frames   </title>          </head>
<frameset       cols = " 40% " , * >      <framset        rows = " 50% " , * >
        <frame          src = "frame1.html"/>           <frame src = "frame2.html/>"
</frameset>
<frameset       rows = " 20% " , " 35% " , *>
        <frame          src = "frame3.html"/>           <frame          src = "frame4.html"/>
        <frame          src = "frame5.html"/>           </frameset>     </frameset>     </html>
```

3) Design following table which includes caption , border , cellpadding & cellspacing

NAME	MARKS		
	PowerBuilder	VisualBasic	Developer2000
Shilpa	21	45	30
Vaishali	26	30	40

Answer =

```
< HTML >
<HEAD>  <TITLE>Working With Table</TITLE>      </HEAD>
<BODY BGCOLOR=LIGHTGREY>
<B>Specifing ROWSPAN and COLSPAN Attributes !</B>      <BR><BR><BR><BR>
<CENTER>    <TABLE BORDER=1 WIDTH=50% ALIGN=CENTER>
<TR>            <TH ROWSPAN=2>NAME              <TH COLSPAN=3>MARKS    </TR>
<TR>  <TH>PowerBuilder              <TH>VisualBasic                 <TH>Developer2000    </TR>
<TR ALIGN=CENTER>       <TD> Shilpa    <TD> 21        <TD> 45        <TD> 30        </TR>
<TR ALIGN=CENTER>       <TD> Vaishali  <TD> 26        <TD> 30        <TD> 40 </TR>
<CAPTION ALIGN=bottom><B><BR>Mark Sheet</B></CAPTION>          </TABLE>
</CENTER>    </BODY>       </HTML>
```

4) Create a document with two links to an external document . The first link should lead to the beginning of the external document . The second link should lead to a particular section in the external document . In the external document specify a link that will lead to a particular section within it .

5) XML design

6) Design a Web page for CYBERSHOP INC, using style sheets with the following Specifications : i. Define a style class '.Maxx' with the attributes viz; font – size , color , font – weight & font – family i. Use the defined Style class wherever the text ' CYBERSHOP INC ' appears on the web document i. Use unordered listing giving the list of services offered by CYBERSHOP INC v. Define three segments using <DIV> … </DIV> tags with background colors Blue , Green & Goldenrod positioned accordingly with the some text .	<html> <head> <title>Cybershop Inc</title> <style type="text/css"> ul{List-style:square} .MAXX{Font-Size:26pt;Color:green;Font-weight:BOLD;Font-family:CURSIVE} </style> <body bgcolor="pink"> <P CLASS="MAXX">CYBERSHOP INC</P> Chatting Printing Hacking class Ebusiness <b> Chat Print Hacking class <b> </body>

ASSIGNMENT NO: 3

TITLE	***Perform validations using JavaScript / JQuery***
PROBLEM STATEMENT /DEFINITION	Perform validations using JavaScript / JQuery on one of the applications from the given list : 1. Online pizza order application 2. Student information system for training & placement department 3. Leave management application 4. Blogging platform 5. Meeting room booking application 6. Exam cell automation application
OBJECTIVE	• To understand in depth working of JavaScript and Jquery • To use JavaScript and Jquery for providing validations, animation and effects to any web application.
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating System open source Fedora 20 Networked computer with internet access Editor : IDE : Netbeans 8.1 Web browser Mozilla Firefox, Google Chrome
REFERENCES	1. https://javascript.info/intro 2. https://www.tutorialspoint.com/javascript/index.htm 3. http://www.thedevelopertips.com/JavaScript/JS/Login-form-Validation-in-javascript.aspx?id=5 4. https://learn.jquery.com/about-jquery/ 5. https://aliteralmind.wordpress.com/2014/10/28/login_jquery_validation/
STEPS	Refer to steps below
INSTRUCTIONS FOR WRITING JOURNAL	• Title • Problem Statement • Theory • Design part • Troubleshooting (if any) • Conclusion • References

JAVASCRIPT

An Introduction to JavaScript

Let's see what's so special about JavaScript, what we can achieve with it and which other technologies play well with it.

What is JavaScript?

JavaScript was initially created to *"make webpages alive"*.

The programs in this language are called *scripts*. They can be written right in the HTML and execute automatically as the page loads.

Scripts are provided and executed as a plain text. They don't need a special preparation or a compilation to run.

In this aspect, JavaScript is very different from another language called Java.

Why JavaScript?

When JavaScript was created, it initially had another name: "LiveScript". But Java language was very popular at that time, so it was decided that positioning a new language as a "younger brother" of Java would help.

But as it evolved, JavaScript became a fully independent language, with its own specification called ECMAScript, and now it has no relation to Java at all.

At present, JavaScript can execute not only in the browser, but also on the server, or actually on any device where there exists a special program called the JavaScript engine.

The browser has an embedded engine, sometimes it's also called a "JavaScript virtual machine".

Different engines have different "codenames", for example:

- V8 – in Chrome and Opera.
- SpiderMonkey – in Firefox.
- There are other codenames like "Trident", "Chakra" for different versions of IE, "ChakraCore" for Microsoft Edge, "Nitro" and "SquirrelFish" for Safari etc.

The terms above are good to remember, because they are used in developer articles on the internet. We'll use them too. For instance, if "a feature X is supported by V8", then it probably works in Chrome and Opera.

How engines work?

Engines are complicated. But the basics are easy.

1. The engine (embedded if it's a browser) reads ("parses") the script.
2. Then it converts ("compiles") the script to the machine language.
3. And then the machine code runs, pretty fast.

The engine applies optimization on every stage of the process. It even watches the compiled script as it runs, analyzes the data that flows through it and applies optimization to the machine code based on that knowledge. At the end, scripts are quite fast.

What can in-browser JavaScript do?

The modern JavaScript is a "safe" programming language. It does not provide low-level access to memory or CPU, because it was initially created for browsers which do not require it.

The capabilities greatly depend on the environment that runs JavaScript. For instance, Node.JS supports functions that allow JavaScript to read/write arbitrary files, perform network requests etc.

In-browser JavaScript can do everything related to webpage manipulation, interaction with the user and the webserver.

For instance, in-browser JavaScript is able to:

- Add new HTML to the page, change the existing content, modify styles.
- React to user actions, run on mouse clicks, pointer movements, key presses.
- Send requests over the network to remote servers, download and upload files (so-called AJAX and COMET technologies).
- Get and set cookies, ask questions to the visitor, show messages.
- Remember the data on the client-side ("local storage").

What CAN'T in-browser JavaScript do?

JavaScript's abilities in the browser are limited for the sake of the user's safety. The aim is to prevent an evil webpage from accessing private information or harming the user's data.

The examples of such restrictions are:

- JavaScript on a webpage may not read/write arbitrary files on the hard disk, copy them or execute programs. It has no direct access to OS system functions.

Modern browsers allow it to work with files, but the access is limited and only provided if the user does certain actions, like "dropping" a file into a browser window or selecting it via an <input> tag.

There are ways to interact with camera/microphone and other devices, but they require a user's explicit permission. So a JavaScript-enabled page may not sneakily enable a web-camera, observe the surroundings and send the information to the NSA.

- Different tabs/windows generally do not know about each other. Sometimes they do, for example when one window uses JavaScript to open the other one. But even in this case, JavaScript from one page may not access the other if they come from different sites (from a different domain, protocol or port).

This is called the "Same Origin Policy". To work around that, *both pages* must contain a special JavaScript code that handles data exchange.

The limitation is again for user's safety. A page from http://anysite.com which a user has opened must not be able to access another browser tab with the URL http://gmail.com and steal information from there.

- JavaScript can easily communicate over the net to the server where the current page came from. But its ability to receive data from other sites/domains is crippled. Though possible, it requires explicit agreement (expressed in HTTP headers) from the remote side. Once again, that's safety limitations.

What makes JavaScript unique?

There are at least *three* great things about JavaScript:

- Full integration with HTML/CSS.
- Simple things done simply.
- Supported by all major browsers and enabled by default.

Combined, these three things exist only in JavaScript and no other browser technology.

That's what makes JavaScript unique. That's why it's the most widespread tool to create browser interfaces.

While planning to learn a new technology, it's beneficial to check its perspectives. So let's move on to the modern trends that include new languages and browser abilities.

Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features −

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multiprocessor capabilities.

Once again, JavaScript is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages.

Languages "over" JavaScript

The syntax of JavaScript does not suit everyone's needs. Different people want different features.

That's to be expected, because projects and requirements are different for everyone.

So recently a plethora of new languages appeared, which are *transpiled* (converted) to JavaScript before they run in the browser.

Modern tools make the transpilation very fast and transparent, actually allowing developers to code in another language and autoconverting it "under the hood".

Examples of such languages:

- CoffeeScript is a “syntactic sugar” for JavaScript, it introduces shorter syntax, allowing to write more precise and clear code. Usually Ruby devs like it.
- TypeScript is concentrated on adding “strict data typing”, to simplify development and support of complex systems. It is developed by Microsoft.
- Dart is a standalone language that has its own engine that runs in non-browser environments (like mobile apps). It was initially offered by Google as a replacement for JavaScript, but as of now, browsers require it to be transpiled to JavaScript just like the ones above.

There are more. Of course even if we use one of those languages, we should also know JavaScript, to really understand what we’re doing.

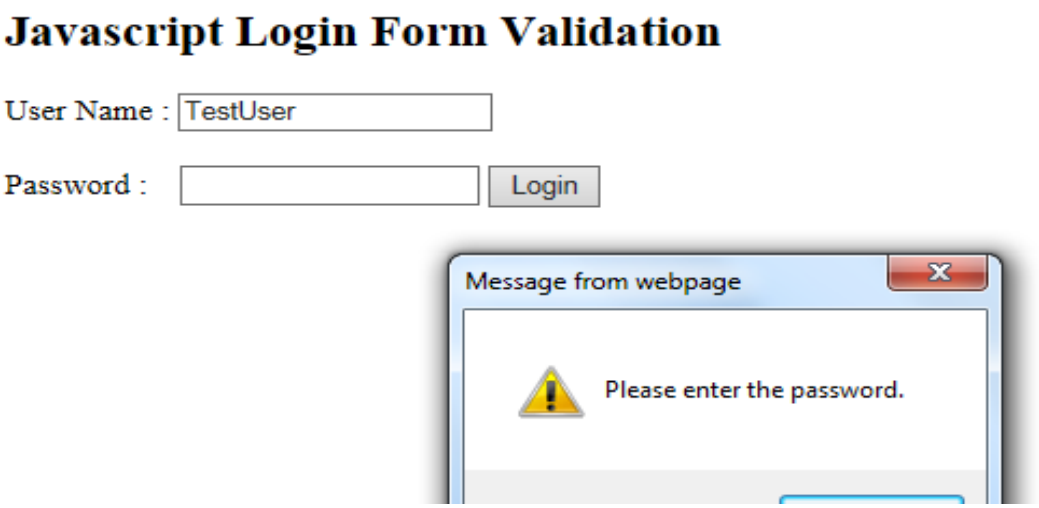
Login Form Validation using JavaScript

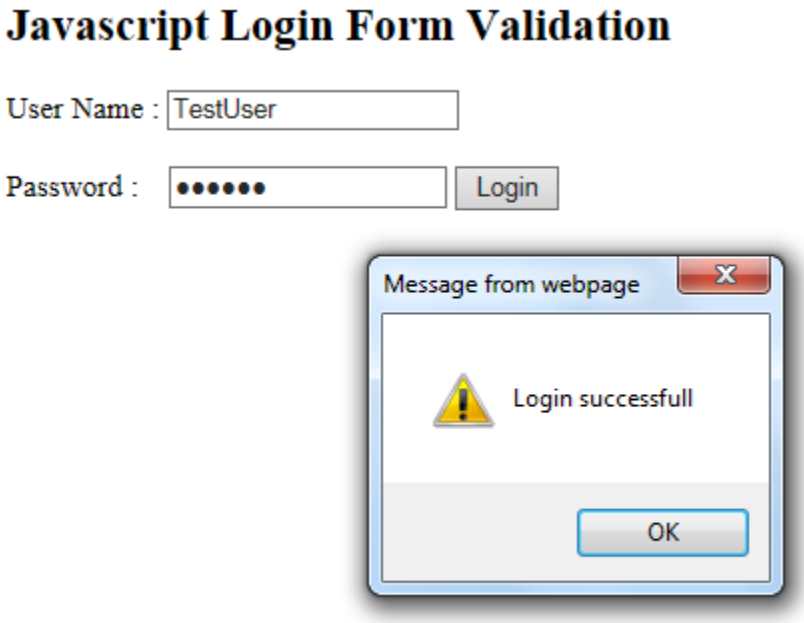
JavaScript code:

```
<script language = "JavaScript">
      function validate() {
         var username = document.getElementById("username").value;
         var password = document.getElementById("password").value;
         if (username == null || username == "") {
            alert("Please enter the username.");
            return false;
         }
         if (password == null || password == "") {
            alert("Please enter the password.");
            return false;
         }
         alert('Login successful');
      }
</script>
```

HTML code:

```
<html>
<body>
<form id="form1" runat="server">
   <div class="container">
      <div class="main">
         <h2> Javascript Login Form Validation</h2>
         <form id="form_id" method="post" name="myform">
         <label> User Name :</label>
         <input type="text" name="username" id="username" />
         <br><br>
         <label> Password :</label>
         <input type="password" name="password" id="password" />
         <input type="button" value="Login" id="submit" onclick="validate();" />
         </form>
      </div>    </div>    </form>
</html> </body>
```





Summary of JavaScript

- JavaScript was initially created as a browser-only language, but now it is used in many other environments as well.
- At this moment, JavaScript has a unique position as the most widely-adopted browser language with full integration with HTML/CSS.
- There are many languages that get "transpiled" to JavaScript and provide certain features. It is recommended to take a look at them, at least briefly, after mastering JavaScript.

JQUERY

About JQuery

Getting started with jQuery can be easy or challenging, depending on your experience with JavaScript, HTML, CSS, and programming concepts in general. In addition to these articles, you can read about the history of jQuery and the licensing terms that apply to jQuery projects. You can also make a donation to help the jQuery team continue to improve jQuery.

One important thing to know is that jQuery is just a **JavaScript library**. All the power of jQuery is accessed via JavaScript, so having a strong grasp of JavaScript is essential for understanding, structuring, and debugging your code. While working with jQuery regularly can, over time, improve your proficiency with JavaScript, it can be hard to get started writing jQuery without a working knowledge of JavaScript's built-in constructs and syntax. Therefore, if you're new to JavaScript, we recommend checking out the JavaScript basics tutorial on the Mozilla Developer Network (MDN).

How JQuery works?

This is a basic tutorial, designed to help you get started using jQuery. If you don't have a test page setup yet, start by creating the following HTML page:

```
<!doctype html>
<html>
<head>
   <meta charset="utf-8">
   <title>Demo</title>
</head>
<body>
   <a href="http://jquery.com/">jQuery</a>
   <script src="jquery.js"></script>
   <script>

   // Your code goes here.

   </script>
</body>
</html>
```

The src attribute in the <script> element must point to a copy of jQuery. Download a copy of jQuery from the Downloading jQuery page and store the jquery.js file in the same directory as your HTML file.

Note: When you download jQuery, the file name may contain a version number, e.g., jquery-x.y.z.js. Make sure to either rename this file to jquery.js or update the src attribute of the <script> element to match the file name.

Launching Code on Document Ready

To ensure that their code runs after the browser finishes loading the document, many JavaScript programmers wrap their code in an onload function:

```
window.onload = function() {
    alert( "welcome" );
 };
```

Unfortunately, the code doesn't run until all images are finished downloading, including banner ads. To run code as soon as the document is ready to be manipulated, jQuery has a statement known as the ready event:

```
$( document ).ready(function() {

    // Your code here.

});
```

Note: The jQuery library exposes its methods and properties via two properties of the window object called jQuery and $. $ is simply an alias for jQuery and it's often employed because it's shorter and faster to write.

For example, inside the ready event, you can add a click handler to the link:

```
$( document ).ready(function() {

    $( "a" ).click(function( event ) {

        alert( "Thanks for visiting!" );

    });

});
```

Copy the above jQuery code into your HTML file where it says // Your code goes here. Then, save your HTML file and reload the test page in your browser. Clicking the link should now first display an alert pop-up, then continue with the default behavior of navigating to http://jquery.com.
For click and most other events, you can prevent the default behavior by calling event.preventDefault() in the event handler:

```
$( document ).ready(function() {

    $( "a" ).click(function( event ) {

        alert( "As you can see, the link no longer took you to jquery.com" );

        event.preventDefault();

    });

});
```

Try replacing your first snippet of jQuery code, which you previously copied in to your HTML file, with the one above. Save the HTML file again and reload to try it out.

Complete Example
The following example illustrates the click handling code discussed above, embedded directly in the HTML <body>. Note that in practice, it is usually better to place your code in a separate JS file and load it on the page with a <script> element's src attribute.

```
<!doctype html>
<html>
<head>
   <meta charset="utf-8">
   <title>Demo</title>
</head>
<body>
   <a href="http://jquery.com/">jQuery</a>
   <script src="jquery.js"></script>
   <script>

   $( document ).ready(function() {
      $( "a" ).click(function( event ) {
         alert( "The link will no longer take you to jquery.com" );
         event.preventDefault();
      });
   });

   </script>
</body>
</html>
```

Callbacks and Functions

Unlike many other programming languages, JavaScript enables you to freely pass functions around to be executed at a later time. A *callback* is a function that is passed as an argument to another function and is executed after its parent function has completed. Callbacks are special because they patiently wait to execute until their parent finishes. Meanwhile, the browser can be executing other functions or doing all sorts of other work.

To use callbacks, it is important to know how to pass them into their parent function.

Callback without Arguments

If a callback has no arguments, you can pass it in like this:

```
$.get( "myhtmlpage.html", myCallBack );
```

When $.get() finishes getting the page myhtmlpage.html, it executes the myCallBack() function.

- **Note:** The second parameter here is simply the function name (but *not* as a string, and without parentheses).

Callback with Arguments

Executing callbacks with arguments can be tricky.

Wrong

This code example will *not* work:

```
$.get( "myhtmlpage.html", myCallBack( param1, param2 ) );
```

The reason this fails is that the code executes myCallBack(param1, param2) immediately and then passes myCallBack()'s *return value* as the second parameter to $.get(). We actually want to pass the function myCallBack(), not myCallBack(param1, param2)'s return value (which might or might not be a function). So, how to pass in myCallBack() *and* include its arguments?

Right

To defer executing myCallBack() with its parameters, you can use an anonymous function as a wrapper. Note the use of function() {. The anonymous function does exactly one thing: calls myCallBack(), with the values of param1 and param2.

```
$.get( "myhtmlpage.html", function() {

   myCallBack( param1, param2 );

});
```

When $.get() finishes getting the page myhtmlpage.html, it executes the anonymous function, which executes myCallBack(param1, param2).

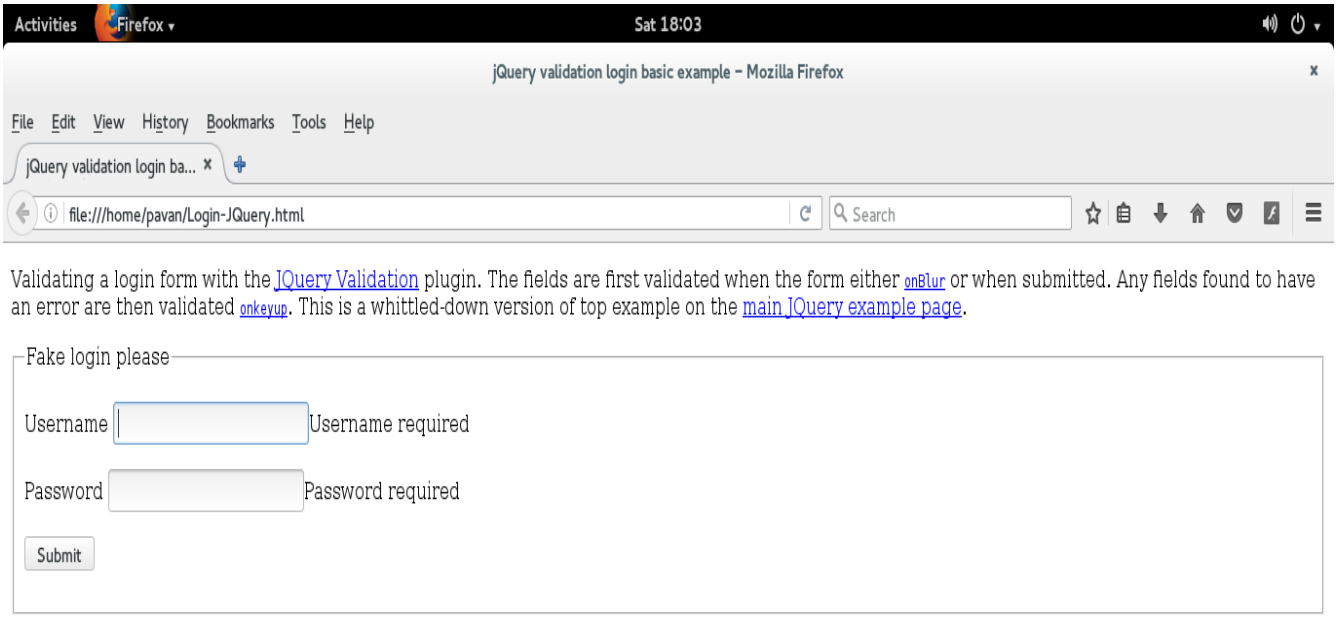
Login form validations using JQuery

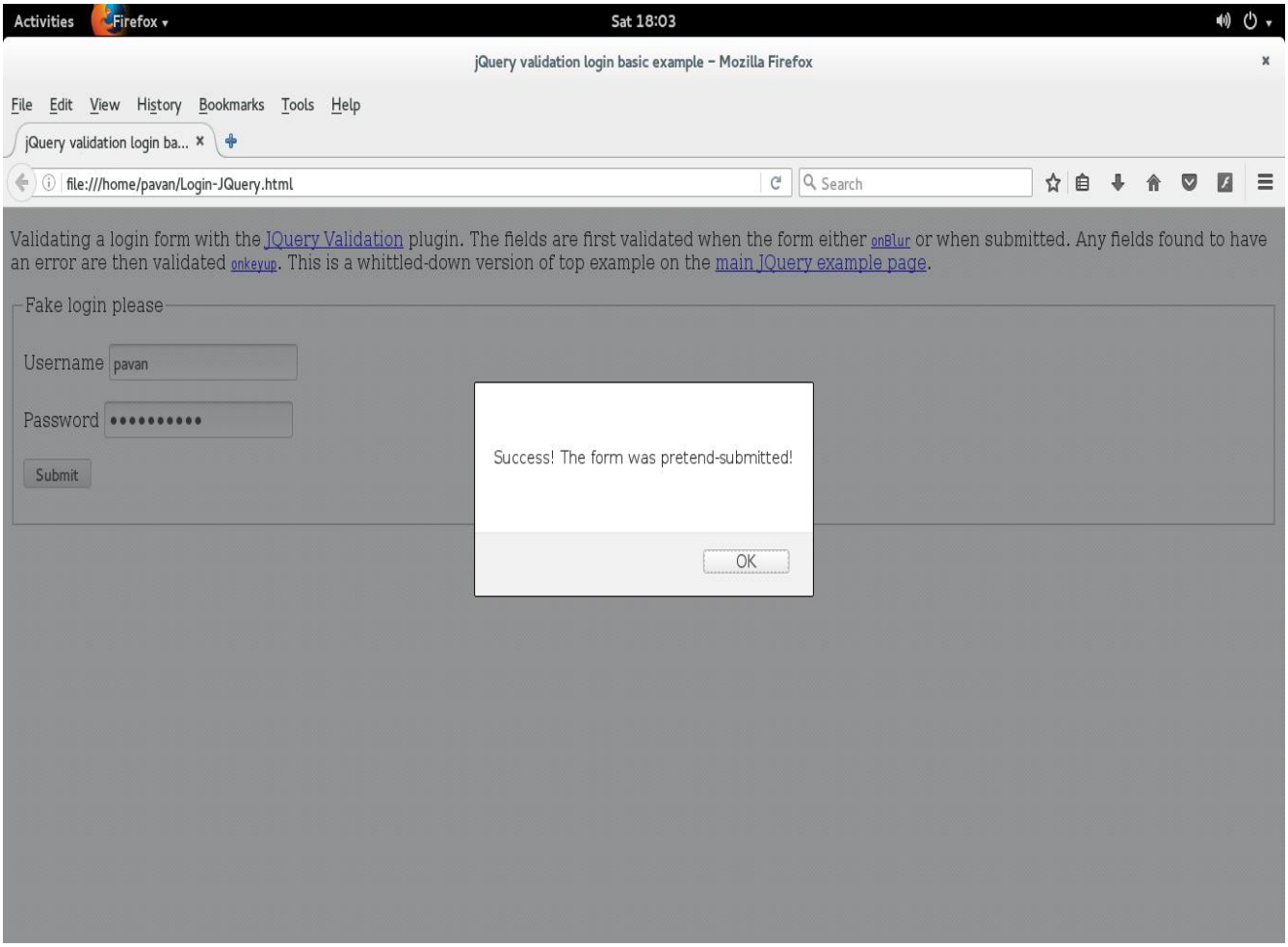
Login-JQuery.html code

```
<!DOCTYPE html>
<html lang="en">
<head>    <meta charset="utf-8">    <title>jQuery validation login basic example</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <meta name="viewport" content="width=device-width"/>
  <script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
  <script src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.13.0/jquery.validate.min.js"></script>
  <script>
    //Configuration
      var minUserLen = 5, maxUserLen = 30;
      var minPassLen = 8, maxPassLen = 4096;
      var usernameMsg = "Username must be between " + minUserLen + " and " +
                  maxUserLen + " characters, inclusive.";
      var passwordMsg = "Password must be between " + minPassLen + " and " +
                  maxPassLen + " characters, inclusive.";
      jQuery.validator.setDefaults({
        debug: true,      //Avoids form submit. Comment when in production.
        success: "valid",
        submitHandler: function() {
          alert("Success! The form was pretend-submitted!");
        }         });
    $(document).ready(function() {
      // validate signup form on keyup and submit
      $("#signupForm").validate({
        rules: {
          username: {
            required: true,
            minlength: minUserLen,
            maxlength: maxUserLen
           },
          password: {
            required: true,
```

```
            minlength: minPassLen,
            maxlength: maxPassLen
          },          },
        messages: {
          username: {
            required: "Username required",
            minlength: usernameMsg,
            maxlength: usernameMsg
          },
          password: {
            required: "Password required",
            minlength: passwordMsg,
            maxlength: passwordMsg
          }          }        });      });
  </script>
  </head>
<body>
  <p>Validating a login form with the <a href="http://jqueryvalidation.org/">JQuery Validation</a> plugin. The fields are first validated when the form either <a href="https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL/Attribute/textbox.onblur"><code>onBlur</code></a> or when submitted. Any fields found to have an error are then validated <a href="https://developer.mozilla.org/en-US/docs/Web/API/Window.onkeyup"><code>onkeyup</code></a>. This is a whittled-down version of top example on the <a href="http://jqueryvalidation.org/files/demo/">main JQuery example page</a>.</p>

  <form id="signupForm">
    <fieldset>
      <legend>Fake login please</legend>
      <p>
        <label for="username">Username</label>
        <input id="username" name="username" type="text">
      </p>
      <p>
        <label for="password">Password</label>
        <input id="password" name="password" type="password">
      </p>
      <p>
        <input class="submit" type="submit" value="Submit">
      </p>
    </fieldset>
  </form>
</body>
</html>
```


Activities
Firefox
Sat 18:03
jQuery validation login basic example - Mozilla Firefox
File Edit View History Bookmarks Tools Help
jQuery validation login ba...
file:///home/pavan/Login-JQuery.html
Search
Validating a login form with the JQuery Validation plugin. The fields are first validated when the form either onBlur or when submitted. Any fields found to have an error are then validated onkeyup. This is a whittled-down version of top example on the main JQuery example page.
Fake login please
Username
Username required
Password
Password required
Submit



References:

1. https://javascript.info/intro
2. https://www.tutorialspoint.com/javascript/index.htm
3. http://www.thedevelopertips.com/JavaScript/JS/Login-form-Validation-in-javascript.aspx?id=5
4. https://learn.jquery.com/about-jquery/
5. https://aliteralmind.wordpress.com/2014/10/28/login_jquery_validation/

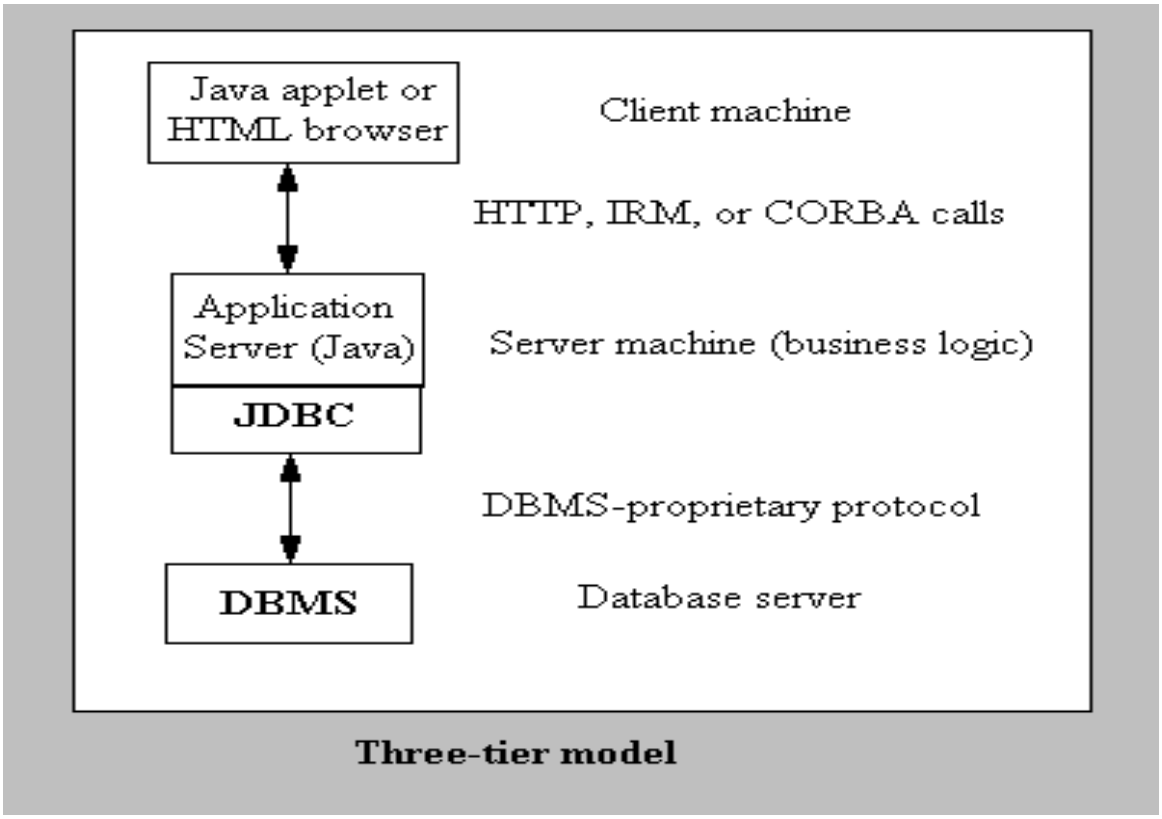
Oral Questions:

1. What is JavaScript?
2. What are JavaScript Data Types?
3. What is the use of isNaan function?
4. Which company developed JavaScript?
5. What are undeclared and undefined variables?
6. Why do we use JQuery?
7. How JavaScript and JQuery are different?
8. Is JQuery a library for client scripting or server scripting?
9. Is JQuery a W3C standard?
10. Which is the starting point of code execution in JQuery?

ASSIGNMENT NO: 4

TITLE	Add dynamic web application features to previously selected application using Servlet, JSP and backend (MySQL / MongoDB).
PROBLEM STATEMENT /DEFINITION	Add dynamic web application features using Servlet, JSP and backend (MySQL / MongoDB) to any one application from the given list : 1. Online pizza order application 2. Student information system for training & placement department 3. Leave management application 4. Blogging platform 5. Meeting room booking application 6. Exam cell automation application
OBJECTIVE	• To understand 3-tier applications • To explore the usage of Servlet, JSP • To understand database connectivity
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating System open source Fedora 20 Networked computer with internet access Editor : IDE : Netbeans 8.1 Web browser Mozilla Firefox, Google Chrome
REFERENCES	1. https://www.javatpoint.com/servlet-tutorial 2. https://www.javatpoint.com/jsp-tutorial 3. https://www.tutorialspoint.com/articles/implementing-jsp-on-netbeans-ide
STEPS	Refer to steps below
INSTRUCTIONS FOR WRITING JOURNAL	• Title • Problem Statement • Theory • Architecture diagrams (design part) • Troubleshooting (if any) • Conclusion • References

Theory:



Three-tier model

Database Server

MySQL version --- 5.5.37
Installed on Fedora 20
MySQL: Installation steps

1. On server machine open terminal.
2. change user to root user using su command.
3. type "yum install mysql mysql-server"
   It will take few mins to complete.
4. on server machine type service mysqld start. (one time start,on each reboot you have to execute the command)
5. To enable mysql service to execute in any run level type chkconfig mysqld on.(this step is not necessary)
6. Open port on server machine for mysql using command
   "iptables -I INPUT -p tcp --dport 3306 -j ACCEPT"(on every server reboot)
7. execute mysql_secure_installation command.
8. It will prompt for root password. As we have not set up root password just press enter.
9. It will ask whether you want to set root password, type yes and provide new password of your choice.

Follow the remaining instructions
10. Now installation is over.
11.Type mysql -u root -p on server machine command prompt. It will prompt you for the password.
12. Once you logged in mysql you have to create separate user account and database for each student(client).
13. To create user account
type command-- > create user 'user_name'@'%' identified by 'user_password';
14. Grant permissions on the databases to users.
commands for creating database and granting permissions are given in mysql doc
For example, >grant all on Student_db.* to 'user_name'@'%";
15. The newly created account can be used by students(clients).
Client can connect to the server from any machine using following command
mysql -h ip_address_of_server_machine -u user_name_for_that_student -p
It will prompt for password.
Enter the correct password.

MySQL Client Side Installation
1. On each client machine open terminal.
2. change user to root user using su command.
3. use command --
> yum install mysql
4. Once installed client can login and connect to the server using command--
>mysql -h ip_address_of_server_machine -u user_name_for_that_student -p

For jsp development

On developers machine install suitable version of java. (java 1.7 or 1.8)
Install Apache tomcat suitable version using the steps given next. (apache tomcat 8.0.9)

Tomcat Installation Steps

1.Download tomcat tar file.
2.create tomcat installation directory anywhere using command *mkdir*.
e.g. *mkdir /opt/tomcat_installation*
3.cp apache-tomcat-{version}.tar.gz /opt/tomcat_installation
4.cd /opt/tomcat_installation
5.tar -xvf apache-tomcat-{version}.tar.gz
6.It will extract apache-tomcat-{version}.tar.gz
7.It will also include bin directory where containing binaries for tomcat.
8. cd apache-tomcat-{version}
8.cd bin.
9.To start tomcat use- ./startup.sh
It will show Tomcat started.
10.Now copy html file that you want to host on tomcat server into tomcat_installation/webapps/ROOT/
11.Open browser and type http://localhost:8080/hello.html
if hello.html is in /opt/tomcat_installation/ webapps/ROOT/ , it will show contents of html.

To host jsp pages with database connectivity
12. Create lib directory using mkdir command inside tomcat_installation/webapps/ROOT/WEB-INF/

13.Now in order to host jsp pages which will connect to database, we have to copy *jstl-1.2.jar* and *mysql-connector.jar* to tomcat_installation/webaps/ROOT/WEB-INF/lib directory.
You can download this jar file from ftp server of the lab
14. Shutdown tomcat server using ./shutdown.sh command from bin directory.
15.cp jsp files to tomcat_installation/webapps/ROOT/.
16. Start tomcat using ./startup.sh
17. You can now try the sample program that follows.

Sample Application: Login

Step A: Creating table and inserting records

1. Create table users in your mysql database using :
```
>mysql -h ip_address_of_server_machine -u user_name_for_that_student -p
>use Student_db;  /* to use your database*/
>create table users(username varchar(10), password varchar(8));
```

/* create table in Student_db*/
2. Insert some records into users table.
```
>insert into users values ('abc' , '123');
>insert into users values ('xyz' , '456');
>insert into users values ('pqr' , '789');
```

3. To check
```
> select * from users;
```

Step B: Creating html page

.*html file* – to be saved in tomcat_installation/webaps/ROOT/
Open editor and create a file login.html
Paste following lines in the login.html and save.

login.html

```
<body>
<form action="login.jsp" method="post">
User name :<input type="text" name="usr" />
password :<input type="password" name="pwd" />
<input type="submit" />
</form>
</body>
```

Step C: Developing jsp with database connectivity

.jsp file – to be saved in tomcat_installation/webaps/ROOT/
Open editor and create a file login.jsp
Paste following lines in the login.jsp and save.
```
<%@ page import ="java.sql.*" %>
<%@ page import ="javax.sql.*" %>
<%
```

```
String userid=request.getParameter("usr");
//session.putValue("userid",userid);
String pwd=request.getParameter("pwd");
Class.forName("com.mysql.jdbc.Driver");
java.sql.Connection con =
DriverManager.getConnection("jdbc:mysql://mysql_server_ip:3306/Student_db","username",
"User_password");
Statement st= con.createStatement();
ResultSet rs=st.executeQuery("select * from users  where username='"+userid+"'");
 if(rs.next())
 {
 if(rs.getString(2).equals(password))
 {
 out.println("welcome   "+userid);
 }
 else
 {
 out.println("Invalid password try again");
 }
 }
 %>
```

To run the application

Open the browser
Type – http://localhost:login.html
(if you are accessing login application from the same machine---developer)
OR
Type-- http://ip_address_of developer_machine:login.html
(If you are accessing login application from any other machine--- client)
HTML page contents will be shown.
Enter username and password ,
Click on submit query button.
If you provide valid username and correct password,
the WELCOME page will be shown.

References:

1. https://www.javatpoint.com/servlet-tutorial
2. https://www.javatpoint.com/jsp-tutorial
3. https://www.tutorialspoint.com/articles/implementing-jsp-on-netbeans-ide

Oral Questions:

1. What is JSP?
2. What are the life cycle methods of JSP?
3. What is the difference between hide comment and output comment?
4. What are the JSP implicit objects?
5. What is difference between include directive and include action?

6. Is JSP technology extensible?
7. How is JSP used in MVC model?
8. What is difference between sendRedirect and forward in Servlet?
9. How do you get ServletContext reference inside Servlet?
10. What is difference between ServletContext and ServletConfig?
11. What is the difference between GET and POST method in HTTP protocol?
12. What does load-on-start-up element do in web.xml?

ASSIGNMENT NO: 5

TITLE	***Design web application using PHP, MySQL, AJAX***
PROBLEM STATEMENT /DEFINITION	Add dynamic web application features to previously selected application using PHP, MySQL database connectivity and AJAX controls.
OBJECTIVE	• To study PHP to design web pages and also study MySQL database connectivity in PHP. • To study how to use AJAX controls in web page.
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating System open source Fedora 20 Networked computer with internet access Editor : IDE : Netbeans 8.1 Web browser Mozilla Firefox, Google Chrome
REFERENCES	1. https://www.w3schools.com/php/default.asp 2. https://www.tutorialspoint.com/php/index.htm 3. https://www.tutorialspoint.com/ajax/index.htm 4. https://www.w3schools.com/xml/ajax_intro.asp
STEPS	Refer to steps below
INSTRUCTIONS FOR WRITING JOURNAL	• Title • Problem Statement • Theory • Design part • Troubleshooting (if any) • Conclusion • References

PHP

PHP stands for Hypertext Preprocessor. It is a programming language used to design dynamic web pages. PHP was started as Open Source project. Rasmus Lerdorf gave first version of PHP in 1994.

PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites. It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server. PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

Common uses of PHP:

PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.

- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible −

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

Sample Program in PHP

Find the 'Hello World' program in PHP as follows:

```
<html>
<head>
<title>Hello World Program in PHP</title>
</head>
<body>
<? php
echo “Hello World!!”;
?>
</body>
</html>
```

Output of the program will be as follows:

```
Hello World!!
```

Database MySQL connectivity using PHP

We need to setup a connection with database. For this, we require url, user id and password for that user id. Function mysql_connect() is used to create a MySQL connection while mysql_close() is used to close the database connection.

Following is the sample PHP code for MySQL database connectivity.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'guest';
   $dbpass = 'guest123';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);
    if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }
   echo 'Connected successfully';
   mysql_close($conn);
  ?>
```

Function mysql_query() is used to execute a SQL query. Following is the sample PHP code to execute a SQL query for creating a database.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);
    if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }
  echo 'Connected successfully';
   $sql = 'CREATE Database test_db';
   $retval = mysql_query( $sql, $conn );
   if(! $retval ) {
      die('Could not create database: ' . mysql_error());
   }
  echo "Database test_db created successfully\n";
   mysql_close($conn);
?>
```

AJAX

AJAX is a web development technique for creating interactive web applications. AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display. Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.

All the available browsers cannot support AJAX. Here is a list of major browsers that support AJAX.

- Mozilla Firefox 1.0 and above.
- Netscape version 7.1 and above.
- Apple Safari 1.2 and above.
- Microsoft Internet Explorer 5 and above.
- Konqueror.
- Opera 7.6 and above.

The whole AJAX operation is given in following steps:

1. A client event occurs.
2. An XMLHttpRequest object is created.
3. The XMLHttpRequest object is configured.
4. The XMLHttpRequest object makes an asynchronous request to the Webserver.
5. The Webserver returns the result containing XML document.
6. The XMLHttpRequest object calls the callback() function and processes the result.
7. The HTML DOM is updated.

Following is the sample program which makes use of AJAX. It reads from the text file ajax_info.txt.

```
<html>
<body>
<div id="demo">
<h1>The XMLHttpRequest Object</h1>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script> </body> </html>
```

References:

1. https://www.w3schools.com/php/default.asp
2. https://www.tutorialspoint.com/php/index.htm
3. https://www.tutorialspoint.com/ajax/index.htm
4. https://www.w3schools.com/xml/ajax_intro.asp

Oral Questions:

1. What is the use of echo PHP?
2. How to include a file to a PHP page?
3. What is difference between include and require?
4. Require_once(), require(), include(), what is difference between them?
5. How to declare an array in PHP?
6. What is use of print in PHP?
7. What is use of in_array() in PHP?
8. How to set cookies in PHP?
9. How to retrieve a cookie value?
10. How to create a session in PHP?

ASSIGNMENT NO: 6

TITLE	**Develop and deploy web application using Struts2**
PROBLEM STATEMENT /DEFINITION	Develop and deploy web application using Struts2 from the given list : 1. Online pizza order application 2. Student information system for training & placement department 3. Leave management application 4. Blogging platform 5. Meeting room booking application 6. Exam cell automation application
OBJECTIVE	• To understand MVC framework using Struts2 • To explore the usage of Struts2 framework
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating System open source Fedora 20 Networked computer with internet access Editor : IDE : Netbeans 8.1 Web browser Mozilla Firefox, Google Chrome
REFERENCES	1. https://www.javatpoint.com/struts-2-tutorial 2. https://dzone.com/tutorials/java/struts/struts-tutorial/struts-mvc-architecture-tutorial.html 3. https://www.quickprogrammingtips.com/struts2/struts-2-netbeans-tutorial.html 4. http://www.pavanjaiswal.com/2018/03/struts-2-hello-world-application-using.html
STEPS	Refer to steps below
INSTRUCTIONS FOR WRITING JOURNAL	• Title • Problem Statement • Theory • Architecture diagrams (design part) • Troubleshooting (if any) • Conclusion • References

Struts 2 Framework:
The **struts 2 framework** is used to develop **MVC-based web application**.

The struts framework was initially created by **Craig McClanahan** and donated to Apache Foundation in May, 2000 and Struts 1.0 was released in June 2001.

The current stable release of Struts is Struts 2.3.16.1 in March 2, 2014.

This struts 2 tutorial covers all the topics of Struts 2 Framework with simplified examples for beginners and experienced persons.

The Struts 2 framework is used to develop MVC (Model View Controller) based web applications. Struts 2 is the combination of **webwork framework** of opensymphony and **struts 1**.

struts2 = webwork + struts1

Struts 2 Features:
Struts 2 provides many features that were not in struts 1. The **important features** of struts 2 framework are as follows:

1. Configurable MVC components
2. POJO based actions
3. AJAX support
4. Integration support
5. Various Result Types
6. Various Tag support
7. Theme and Template support

1. Configurable MVC components
In struts 2 framework, we provide all the components (view components and action) information in struts.xml file. If we need to change any information, we can simply change it in the xml file.

2. POJO based actions
In struts 2 framework, we provide all the components (view components and action) information in struts.xml file. If we need to change any information, we can simply change it in the xml file.

3. AJAX supports
Struts 2 provides support to ajax technology. It is used to make asynchronous request i.e. it doesn't block the user. It sends only required field data to the server side not all. So it makes the performance fast.

4. Integration support
We can simply integrate the struts 2 application with hibernate, spring, tiles etc. frameworks.

5. Various result types
We can use JSP, freemarker, velocity etc. technologies as the result in struts 2.

6. Various tag support

Struts 2 provides various types of tags such as UI tags, Data tags, control tags etc to ease the development of struts 2 application.

7. Theme and template support
Struts 2 provides three types of theme support: xhtml, simple and css_xhtml. The xhtml is default theme of struts 2. Themes and templates can be used for common look and feel.

Struts 2 Architecture and Flow:
The **architecture and flow of struts 2 application**, is combined with many components such as Controller, ActionProxy, ActionMapper, Configuration Manager, ActionInvocation, Inerceptor, Action, Result etc.

Here, we are going to understand the struts flow by 2 ways:

1. struts 2 basic flow
2. struts 2 standard architecture and flow provided by apache struts

Struts 2 basic flow
Let's try to understand the basic flow of struts 2 application by this simple figure 1:

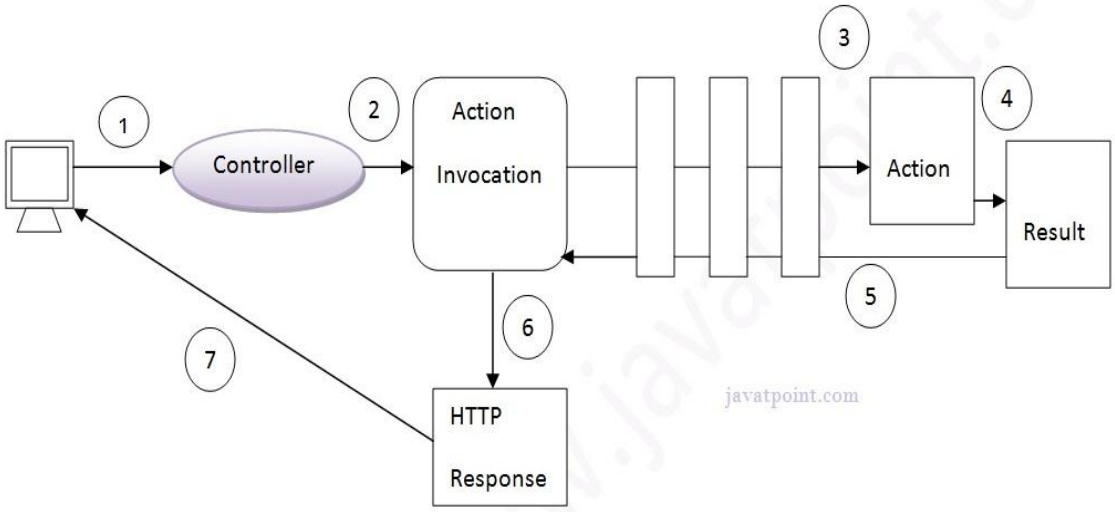
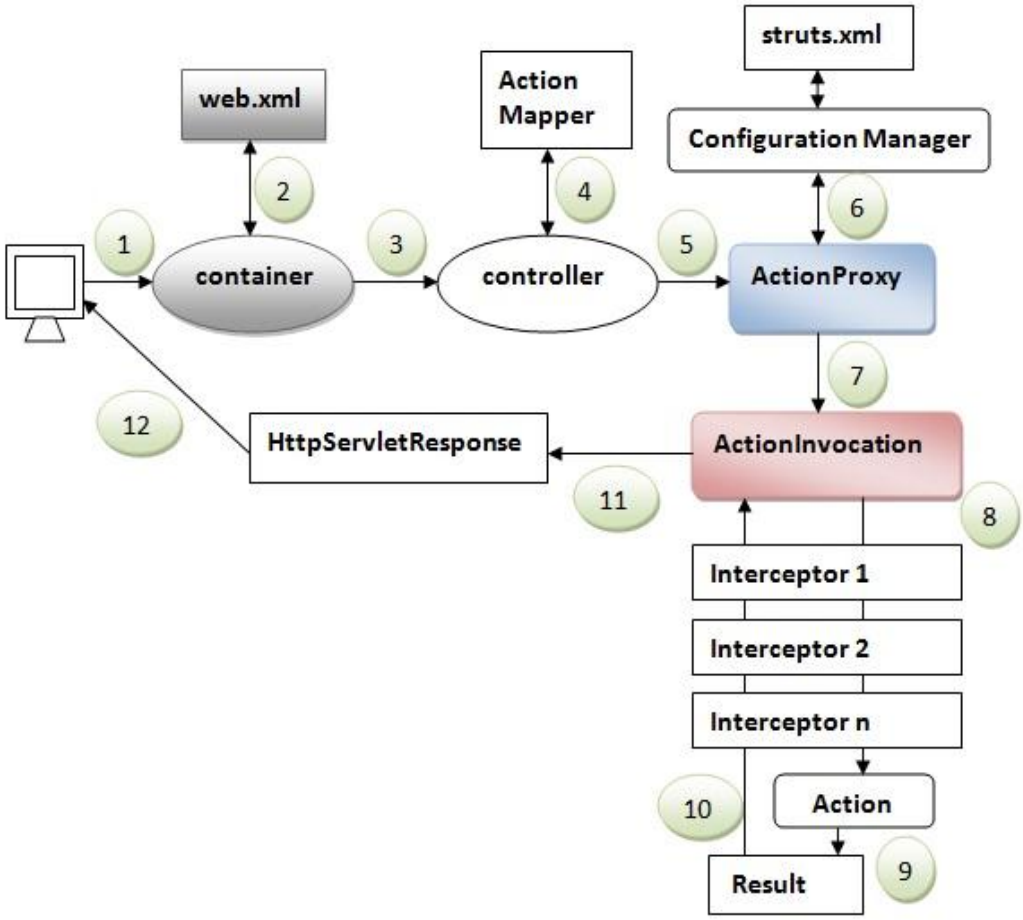


Figure 1: Struts 2 basic flow

1. User sends a request for the action
2. Controller invokes the ActionInvocation
3. ActionInvocation invokes each interceptors and action
4. A result is generated
5. The result is sent back to the ActionInvocation
6. A HttpServletResponse is generated
7. Response is sent to the user

Struts 2 standard flow (Struts 2 architecture)
Let's try to understand the standard architecture of struts 2 application by this simple figure 2:



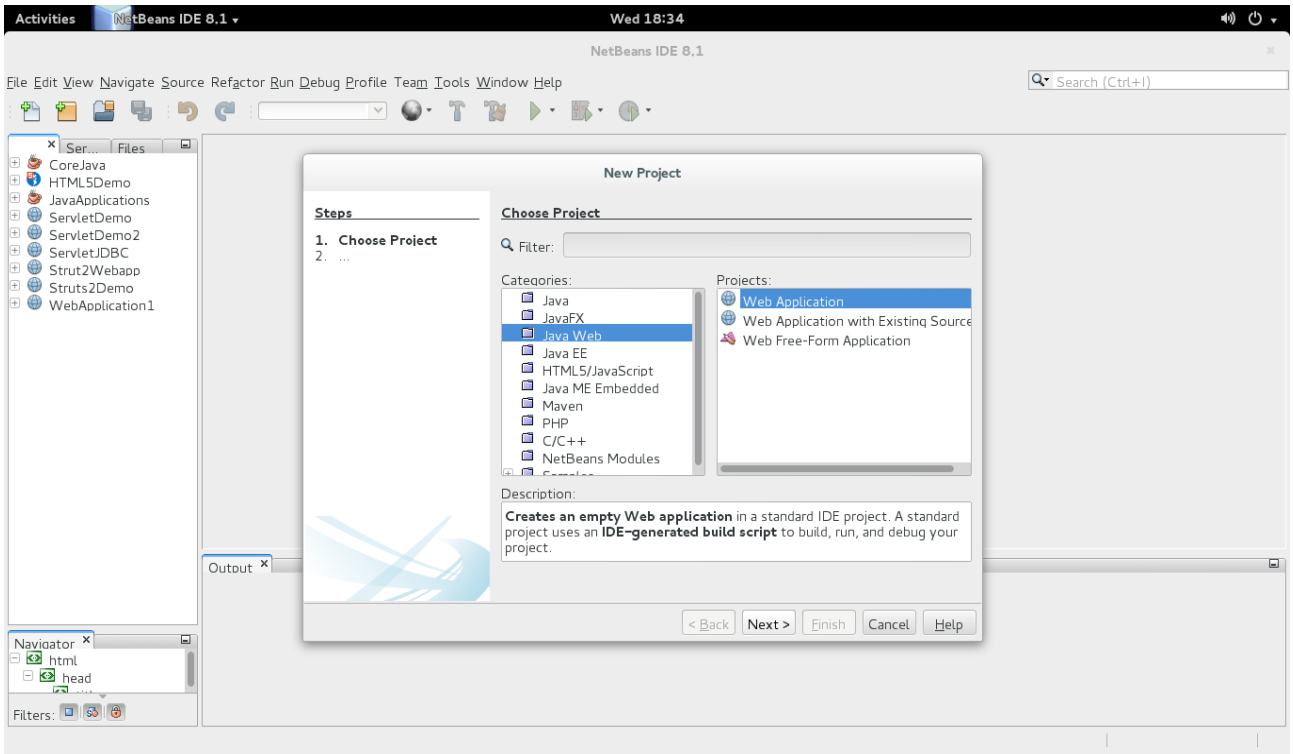
1. User sends a request for the act
2. Container maps the request in the web.xml file and gets the class name of controller.
3. Container invokes the controller (StrutsPrepareAndExecuteFilter or FilterDispatcher). Since struts2.1, it is StrutsPrepareAndExecuteFilter. Before 2.1 it was FilterDispatcher.
4. Controller gets the information for the action from the ActionMapper
5. Controller invokes the ActionProxy
6. ActionProxy gets the information of action and interceptor stack from the configuration manager which gets the information from the struts.xml file.
7. ActionProxy forwards the request to the ActionInvocation
8. ActionInvocation invokes each interceptors and action
9. A result is generated
10. The result is sent back to the ActionInvocation
11. A HttpServletResponse is generated
12. Response is sent to the user

Hello World Program Using Struts 2

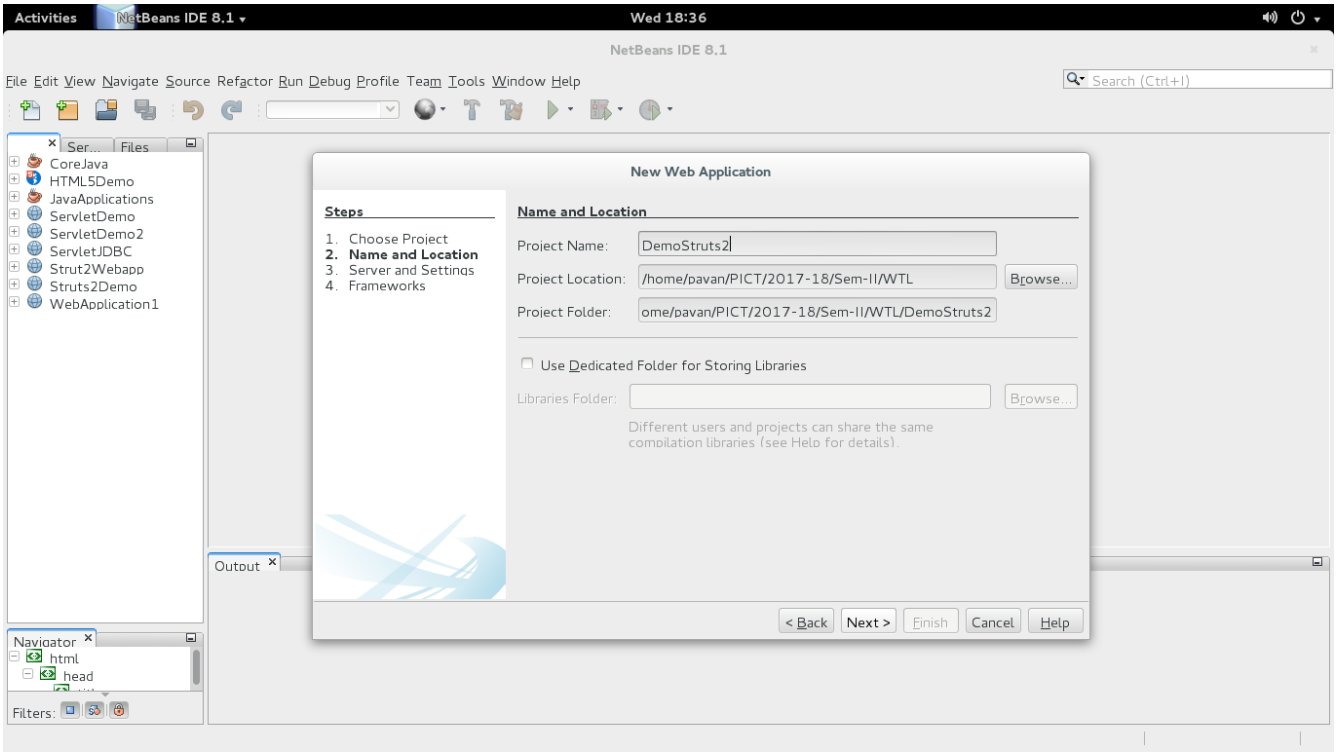
Prerequisite:
Following tools are required to start Struts 2 web application development using NetBeans,

- **Java SDK** – Download and install Java SDK from Oracle site. For this tutorial, I used Java 6, update 26.
- **NetBeans** – Download and install NetBeans IDE from here. Download either the "Java EE" edition or the "All" edition since we need the bundled Tomcat server for running Struts 2 applications. For this tutorial, I used NetBeans 8.1.
- **Struts 2 Library Files** – Download and extract Struts 2 binary distribution files from here. I recommend downloading the "Full Distribution" zip file containing all dependency jar files. However the "Essential Dependencies Only" zip file (which is much smaller than full distribution) is also sufficient for the sample application we are going to build. For this tutorial, I used Struts 2.3.1.2.

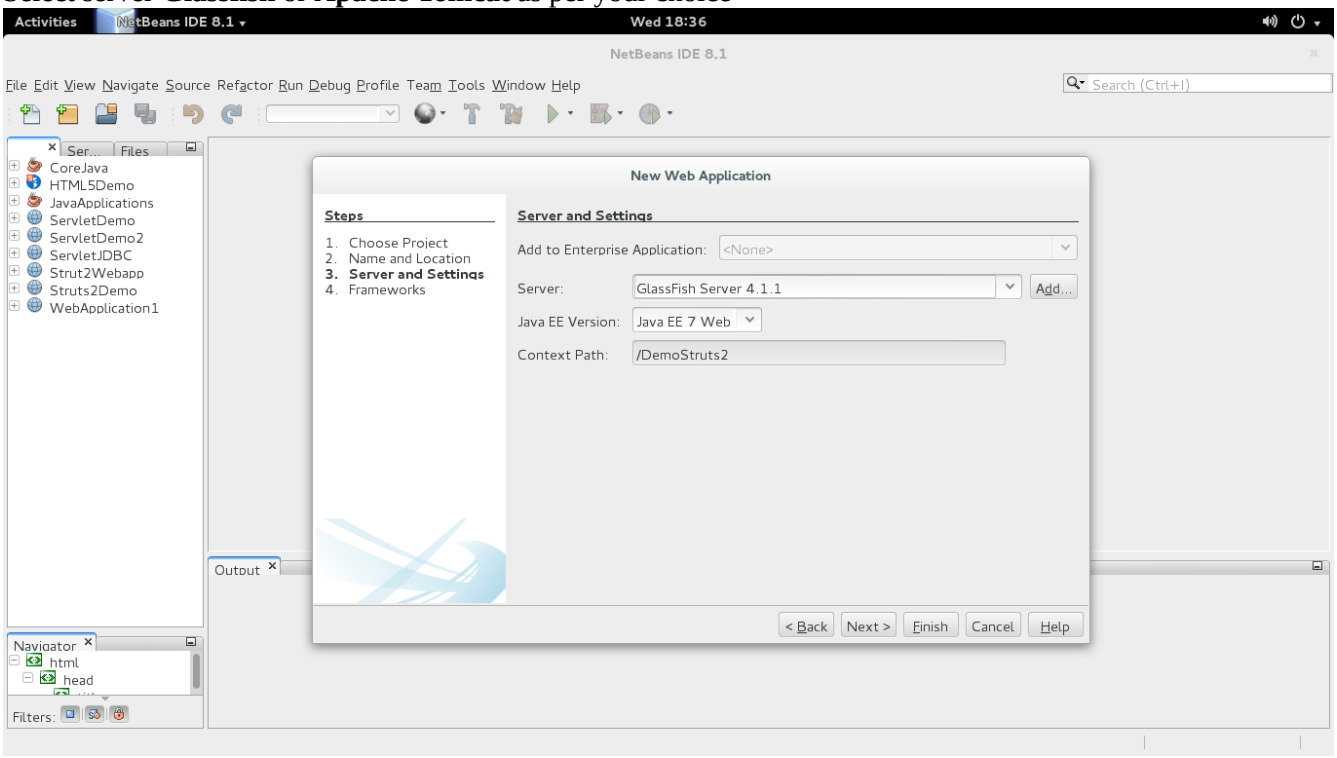
Creating Struts 2 application
Create Java web application as shown below.



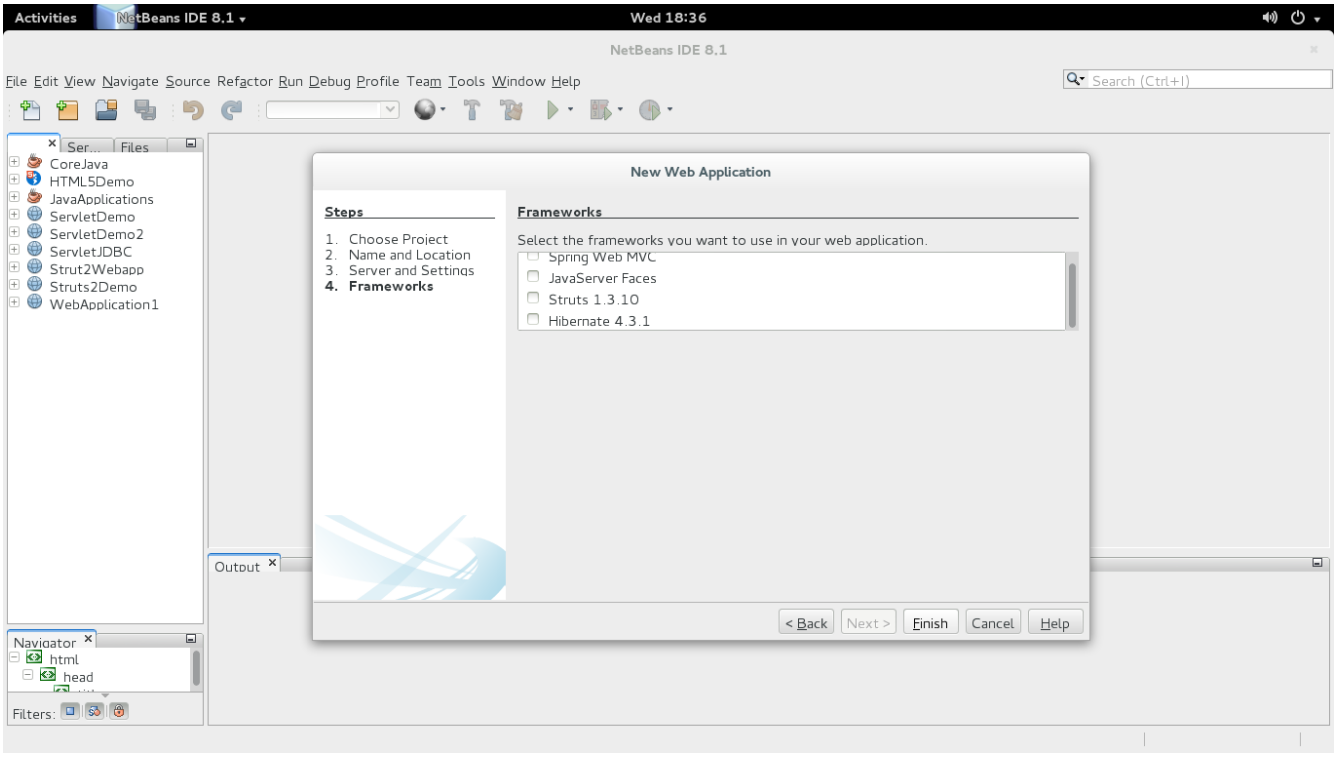
Provide project name: **DemoStruts2**



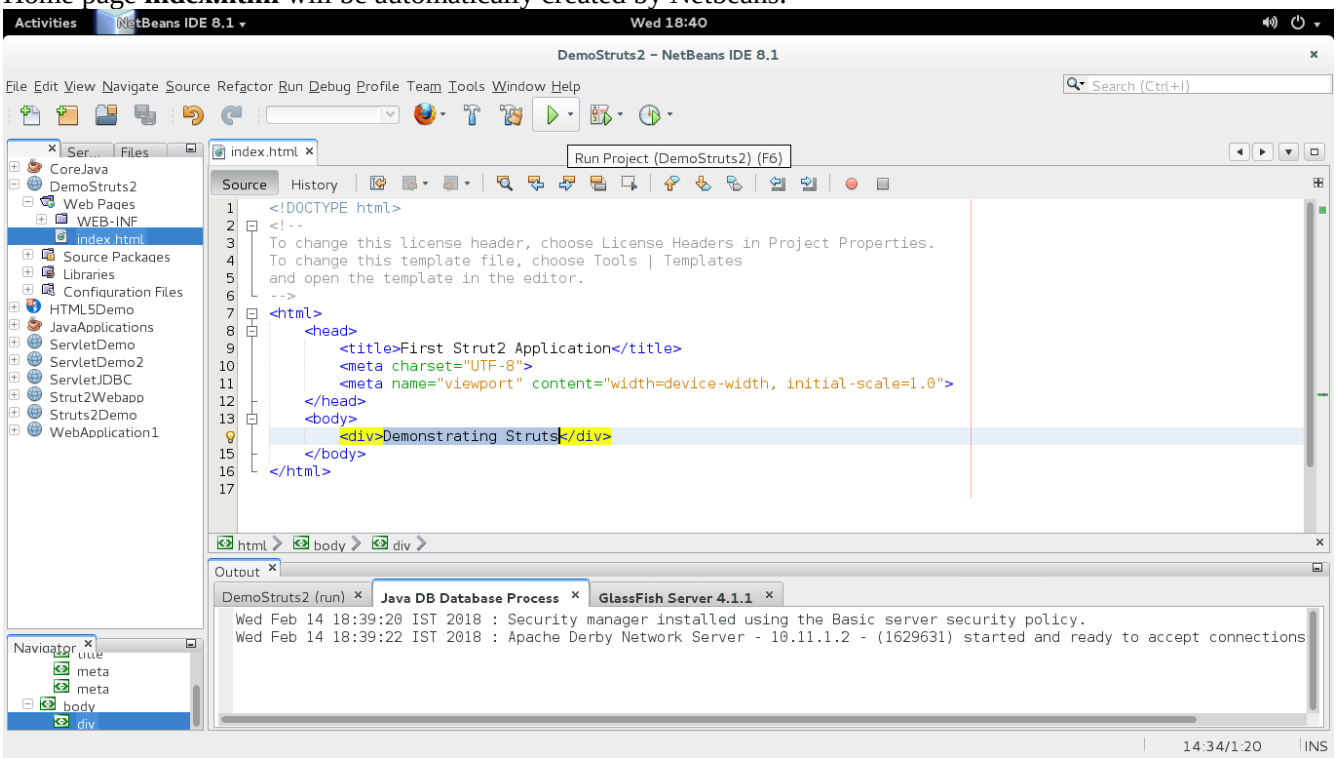
Select server **Glassfish** or **Apache Tomcat** as per your choice



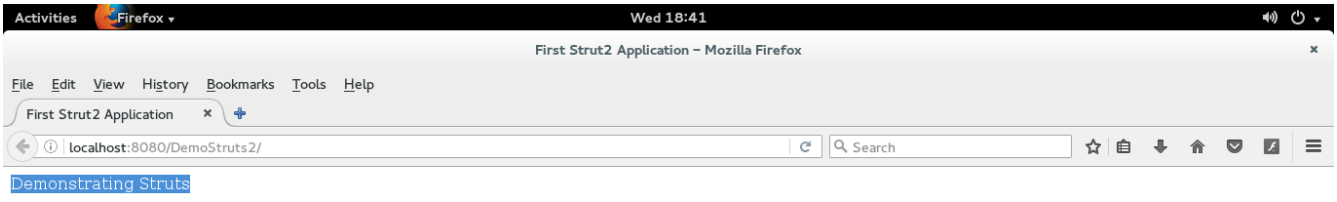
Click on **finish** button.



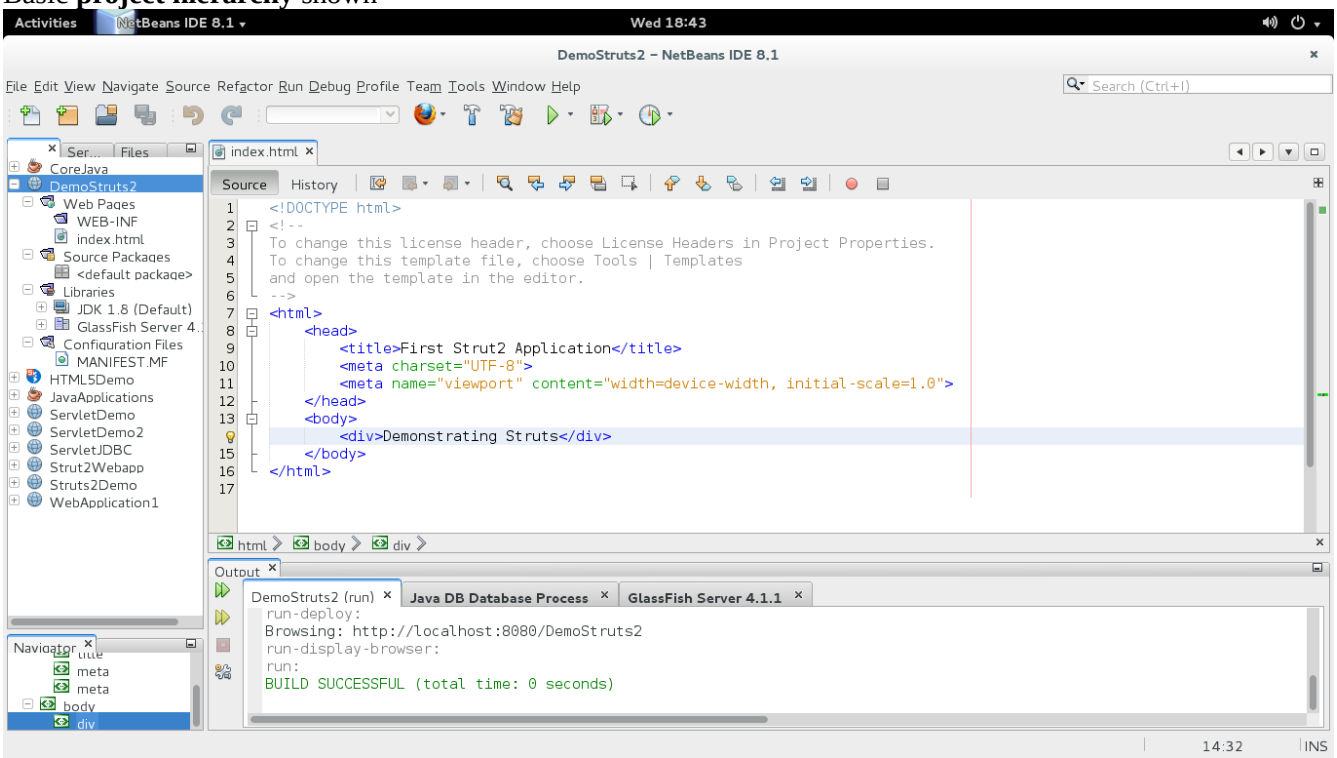
Home page **index.html** will be automatically created by Netbeans.



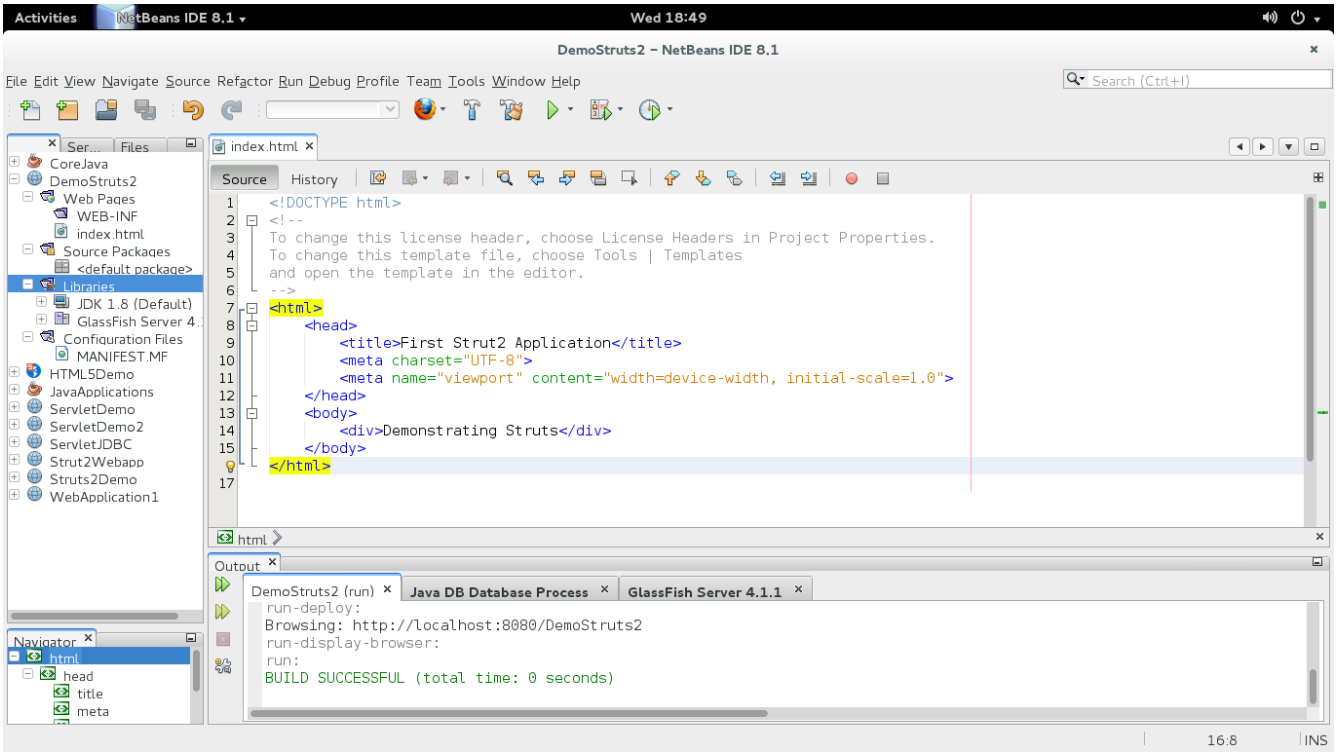
Run the project. Use url: localhost:8080/DemoStruts2



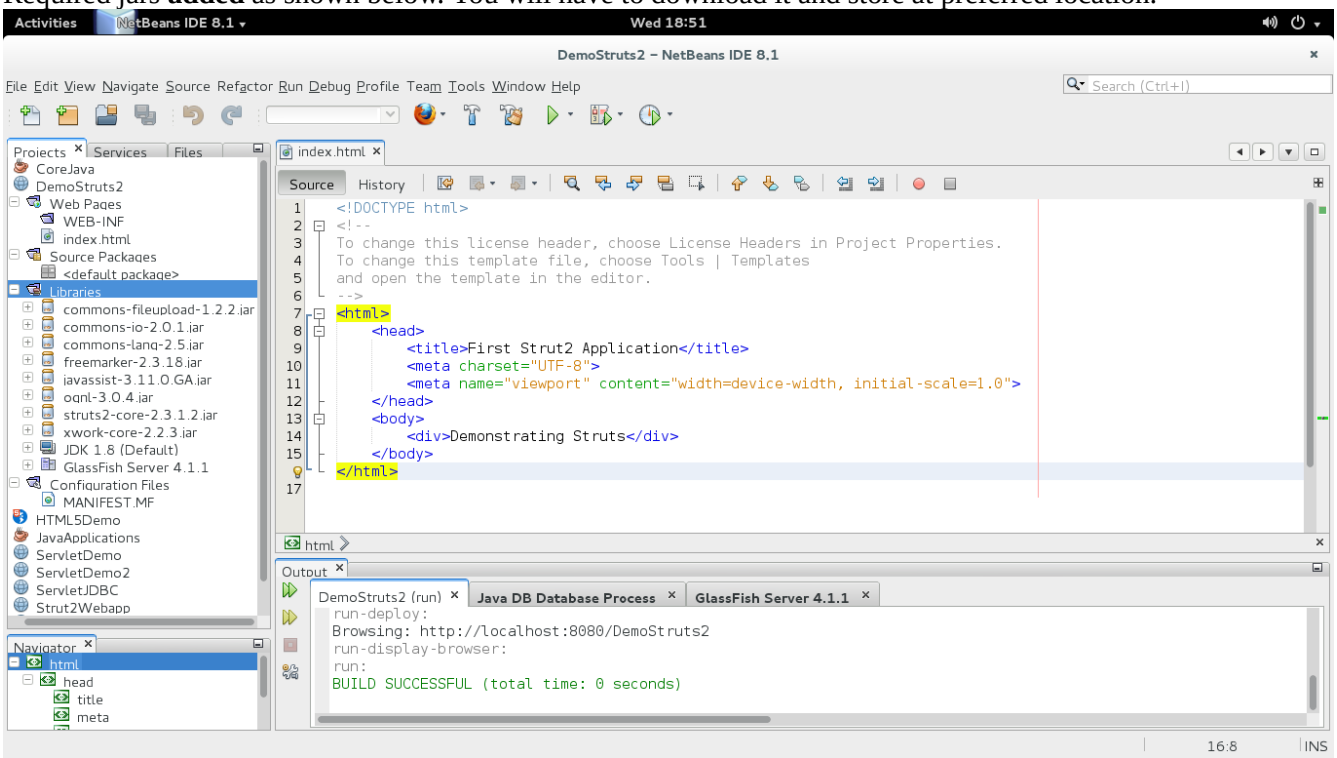
Basic **project hierarchy** shown



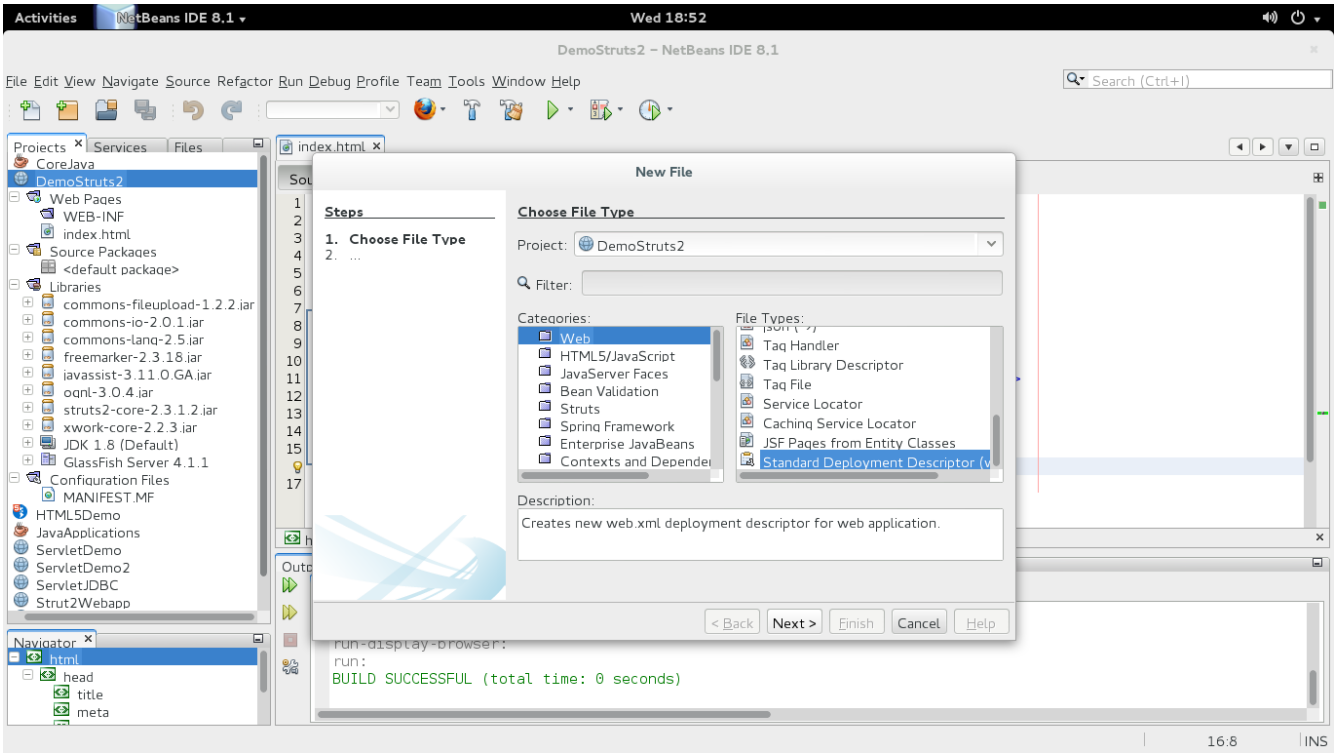
Right click on **libraries** folder to add required **jars**.



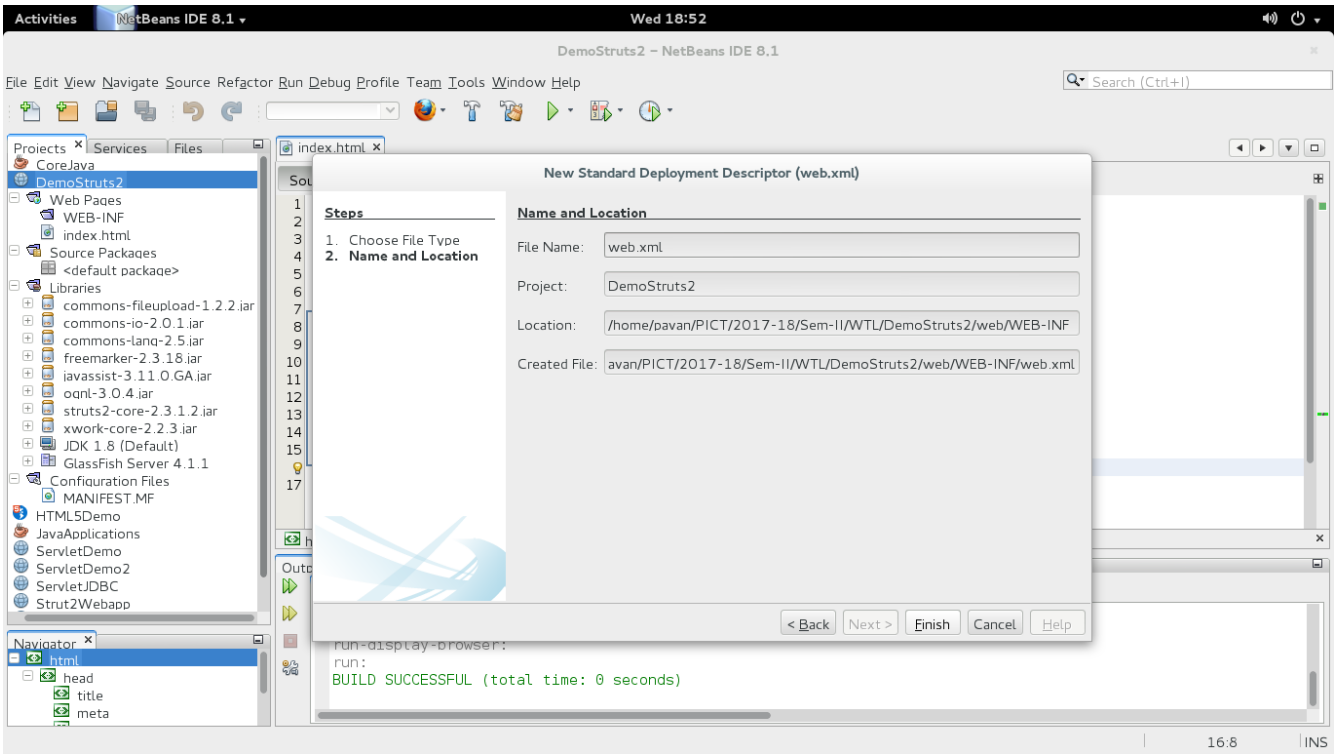
Required jars **added** as shown below. You will have to download it and store at preferred location.



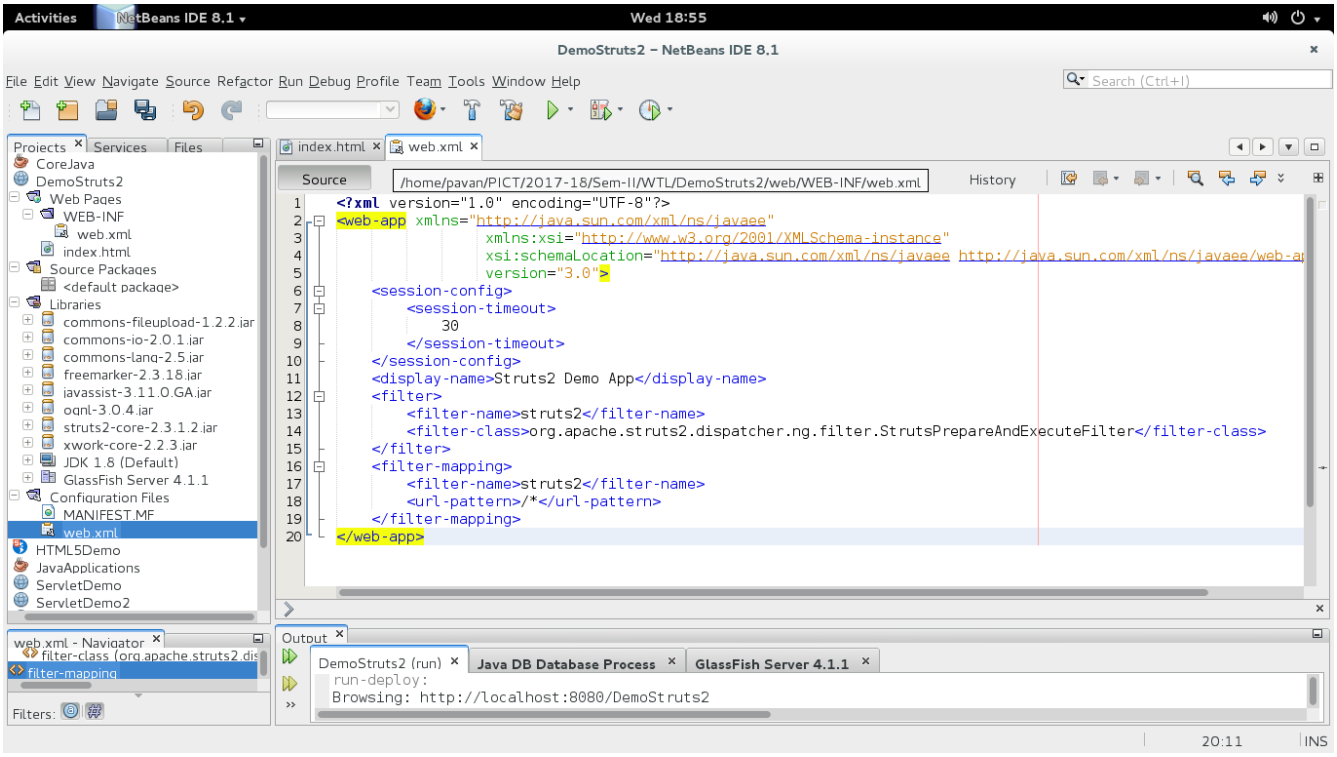
Add **deployment descriptor**



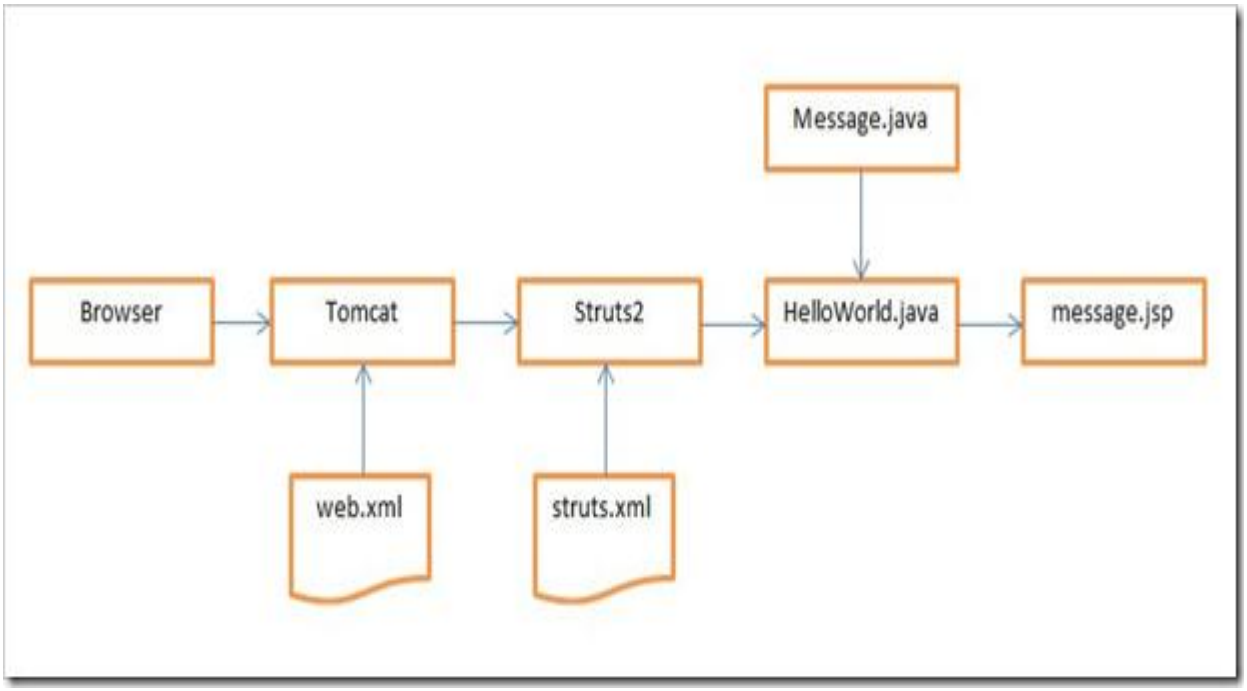
Provide name as "**web.xml**"



Modify web.xml as shown below



Overall architecture of HelloWorld application as shown below



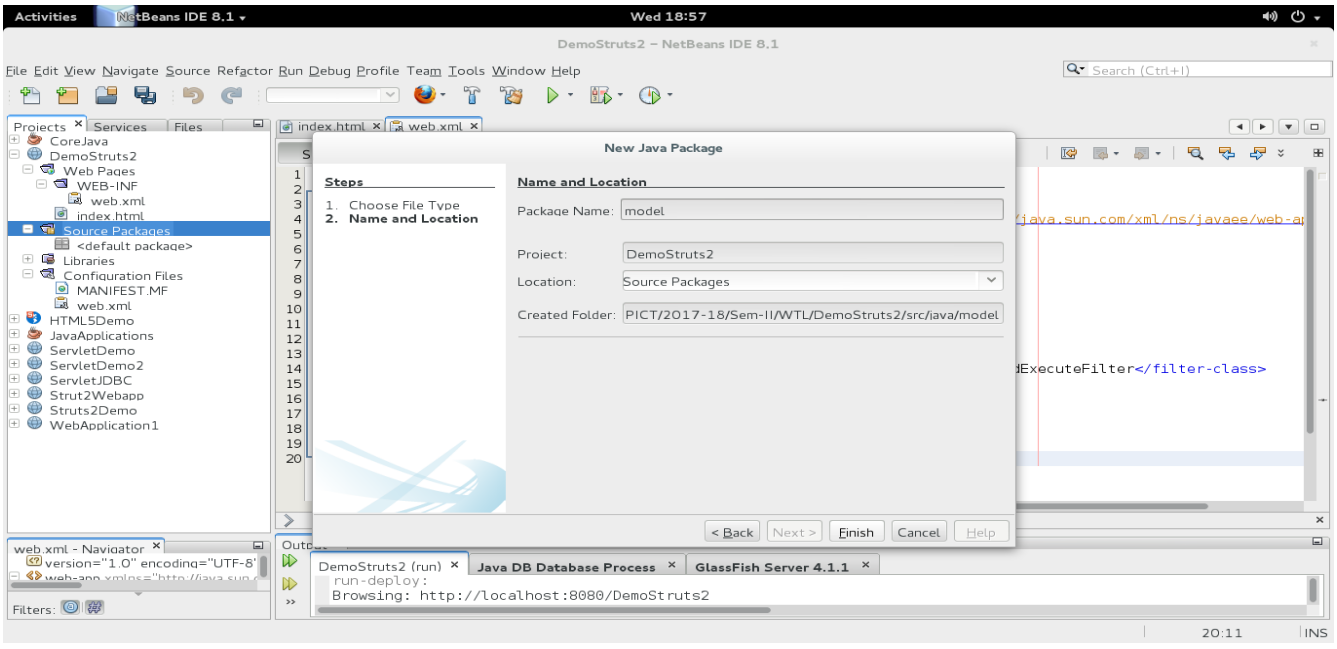
This means that we need the following components,

- **Model** – A class which supplies the data needed for the application. In this demo app, the data supplied is the "Hello World!" message. We will write a simple Message class (**Message.java**) to return the message.
- **View** – A view element which displays the data. In this demo app, we will write a JSP file (**message.jsp**) which will display the message returned by the Model class.
- **Controller** – A controller class which will respond to user URL request. The controller will coordinate the actions of Model and the View. We will write a Struts 2 action class (**HelloWorld.java**) for this.
- **Struts 2 Configuration** – Struts 2 needs to be told about various components of the application. This can be configured in an xml file (**struts.xml**) or we can use a combination of standard naming conventions and annotations. This tutorial will stick to the traditional way of configuring Struts 2 using struts.xml. For the first time user, conventions and annotations may seem like magic!

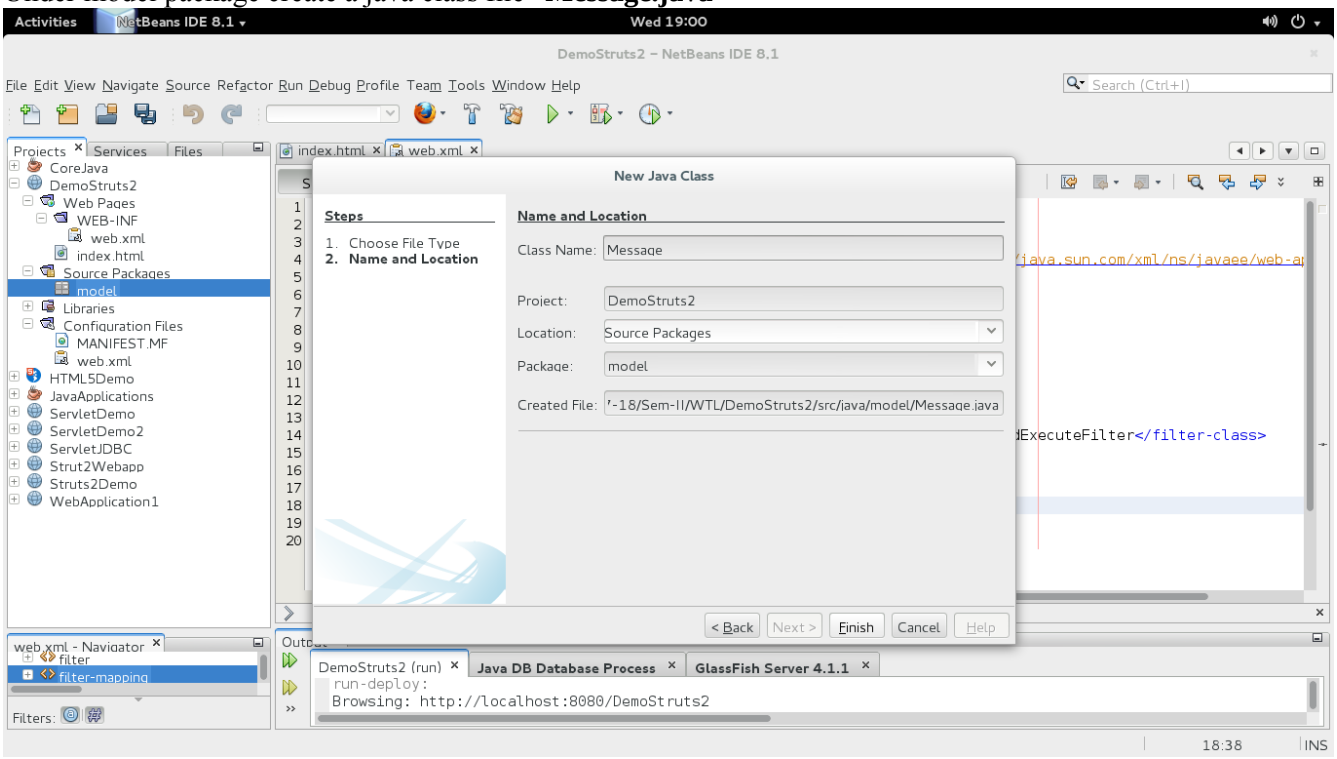
For a simple application like "Hello World!" this may seem like an overkill. However for large enterprise Web applications such separation of concerns is very essential for the robustness, reliability, extensibility and maintainability of the application.

Let us first create our model class which holds the "Hello World!" message. In real applications, these model objects will correspond to persistent data. In this simple example we will be directly using the model object from the controller. However in real applications, we will create a business layer which will return the data. Note that the Message class is organized under model java package.

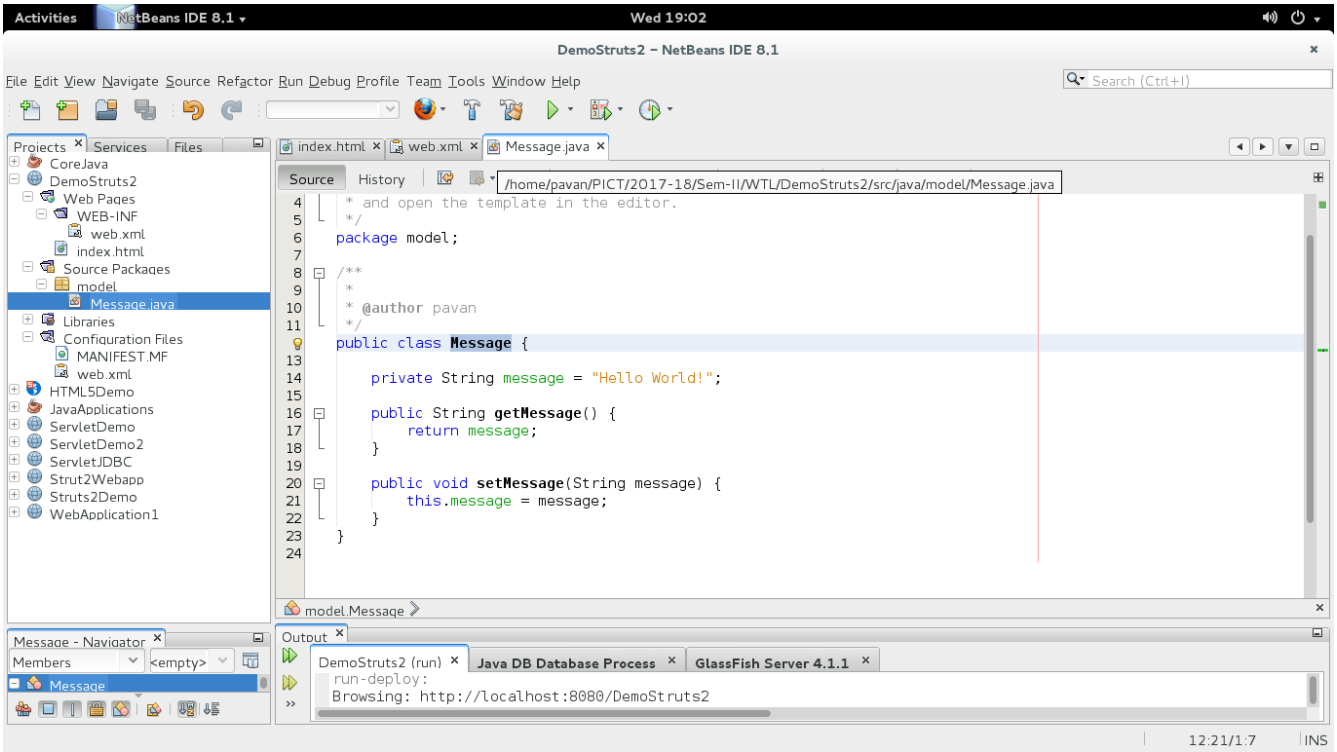
Create **model** package under source package



Under model package create a java class file "**Message.java**"

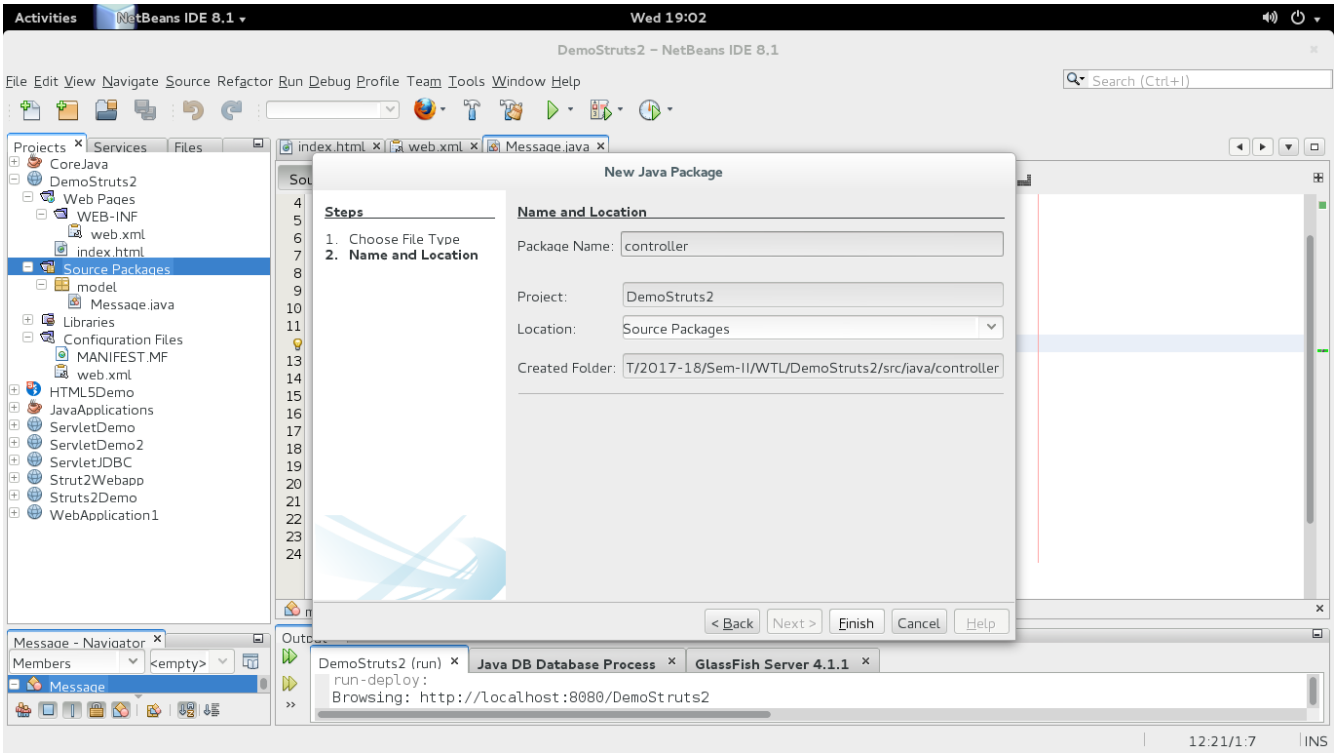


Add contents to Message.java as show below

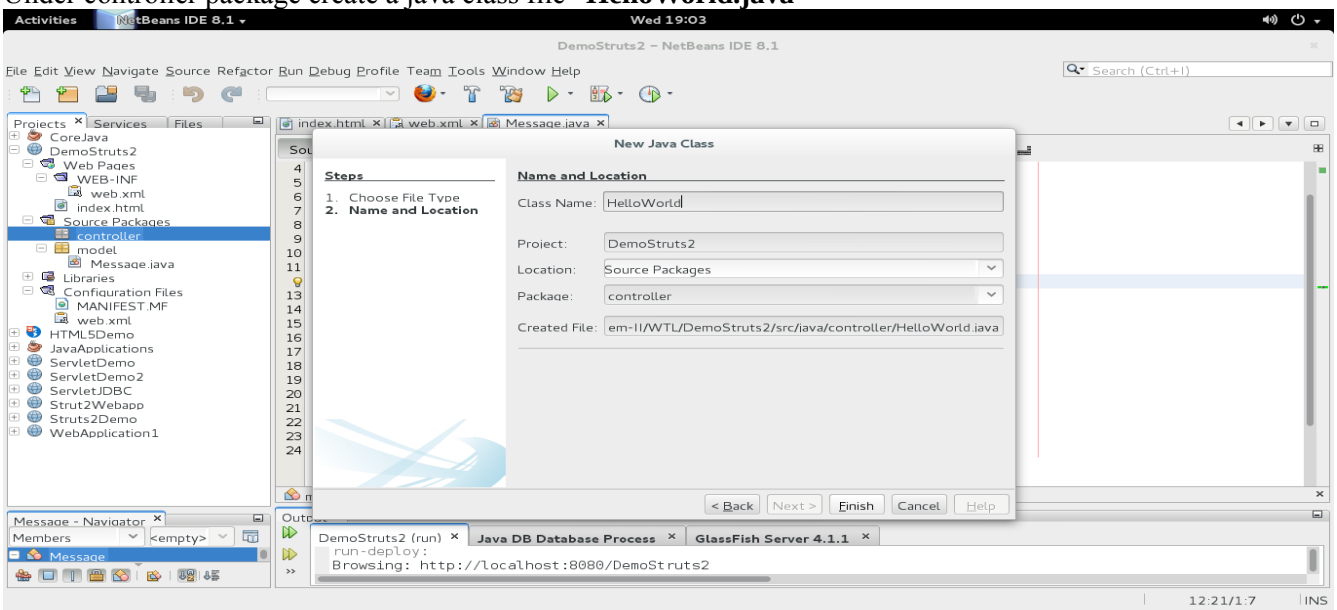


Next we will create an action class (controller) which will respond to the user request. The controller in this case will create/retrieve the message and would pass it along to the view. Typically action classes handling user requests extend from ActionSupport class. The request processing will be done by the overridden execute() method which is automatically called by the framework. Note that the HelloWorld.java (action class) is organized under the controller java package.

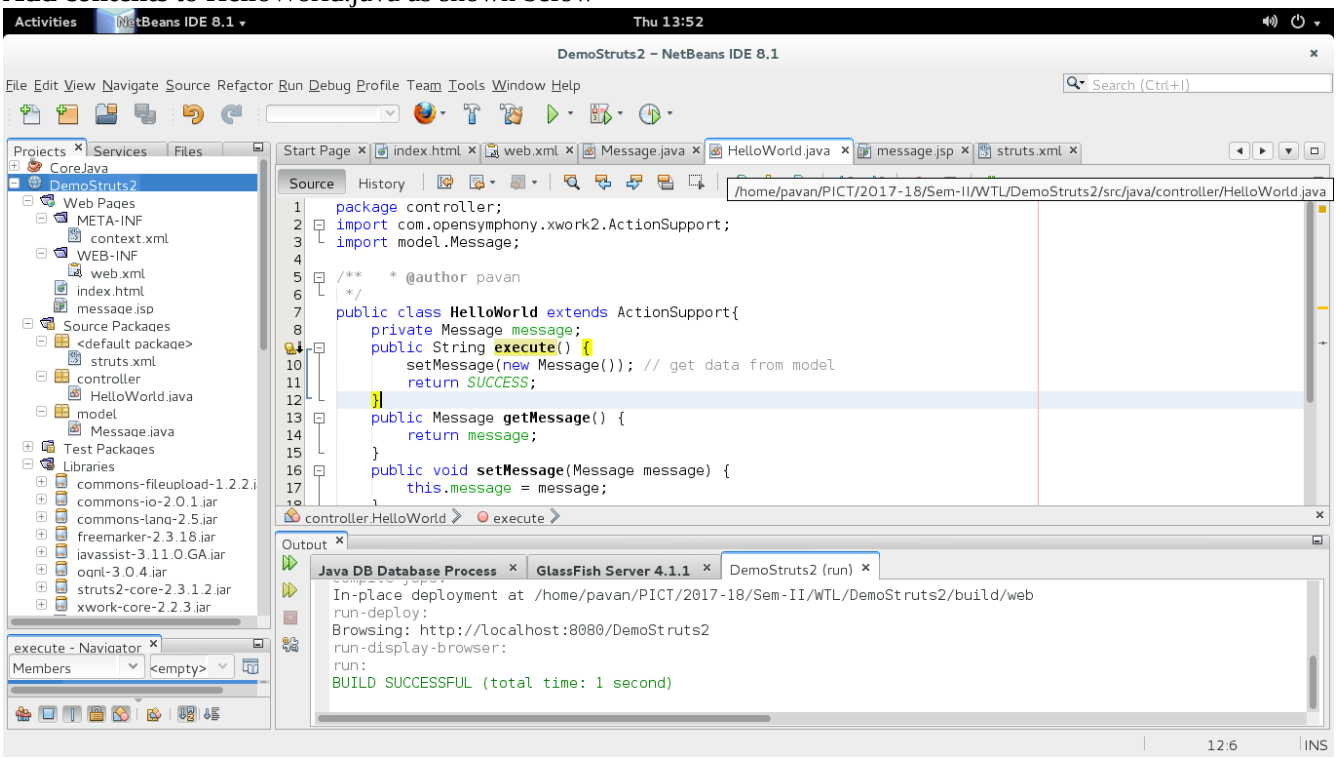
Create **controller** package under source package



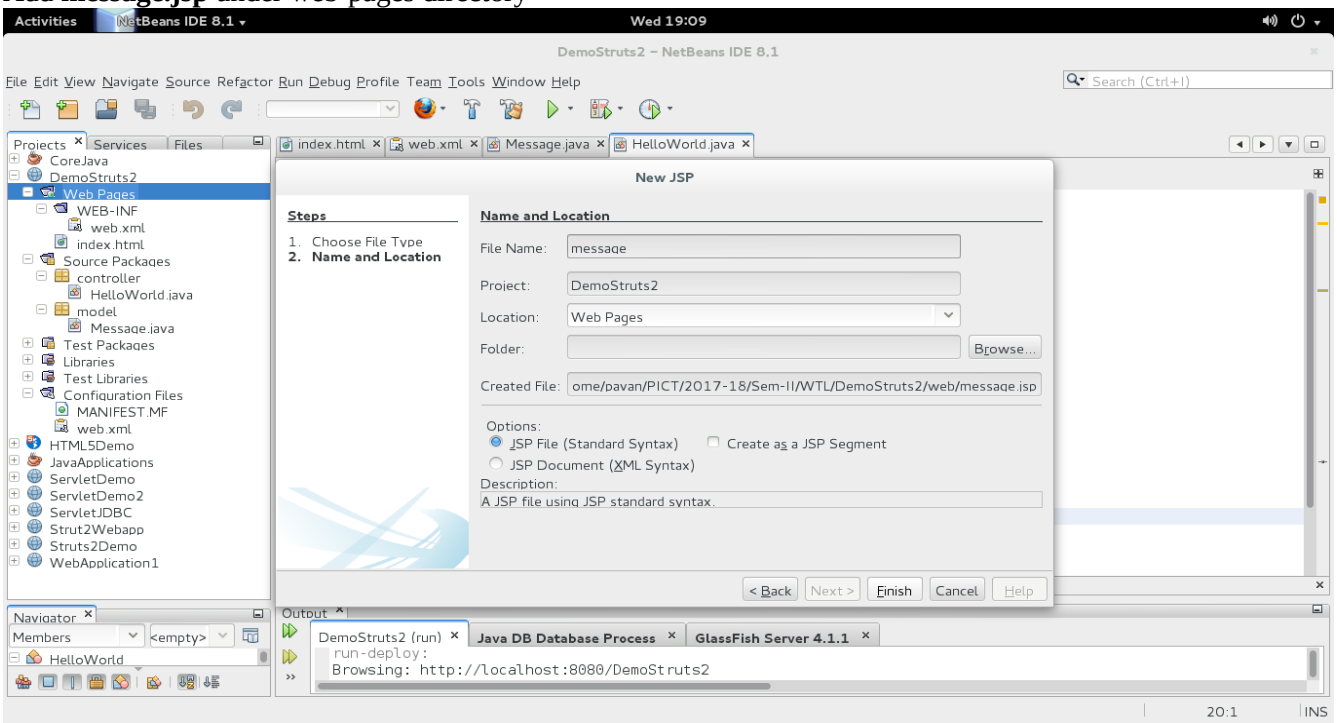
Under controller package create a java class file "**HelloWorld.java**"



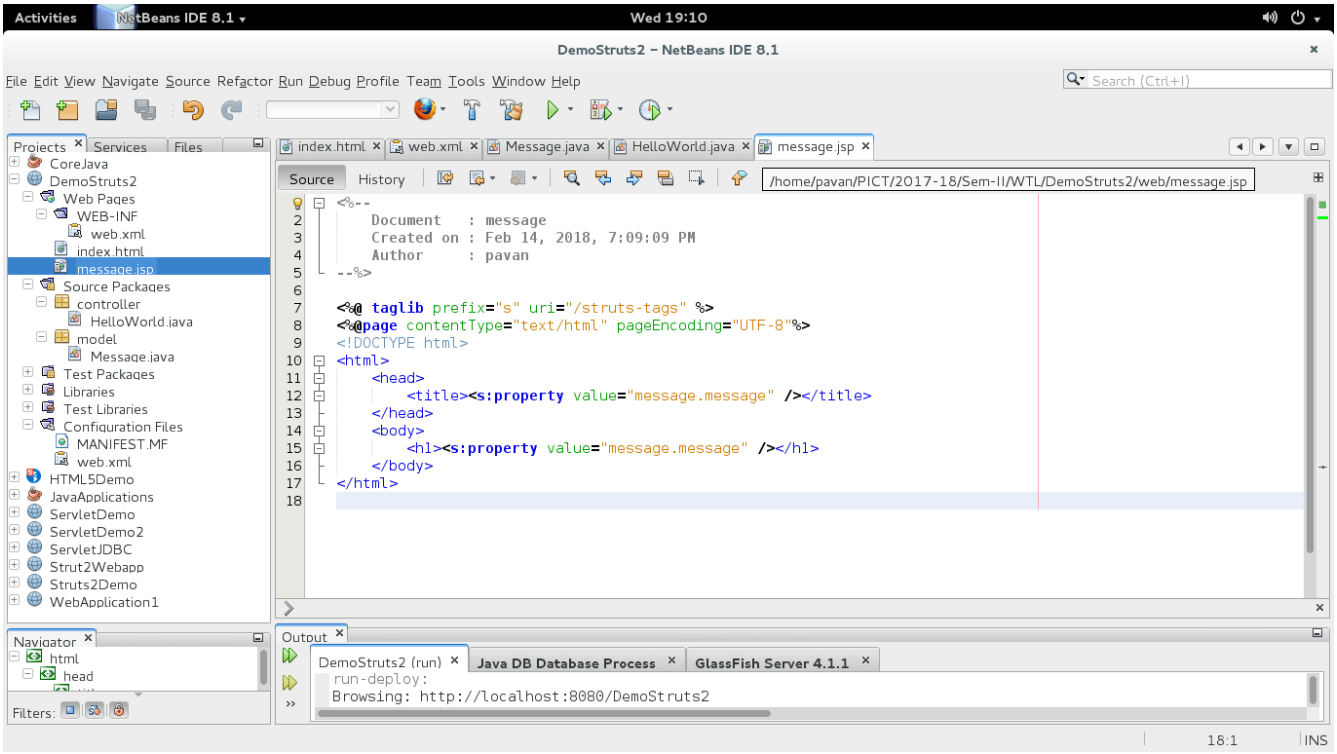
Add contents to HelloWorld.java as shown below



We will now create the user interface. In this demo example we will use the JSP technology for rendering the user interface. Struts 2 supports other view technologies such as freemarker templates and velocity templates.
Add **message.jsp** under web-pages directory



Add contents to message.jsp as shown below



Note the use of custom tags in the JSP. These custom tags are provided by Struts 2. The Struts 2 custom tags use the OGNL expressions to refer to the data exposed by the controller class. This is achieved by the Struts 2 framework by automatically exposing the member variables of the controller class to the tag libraries.

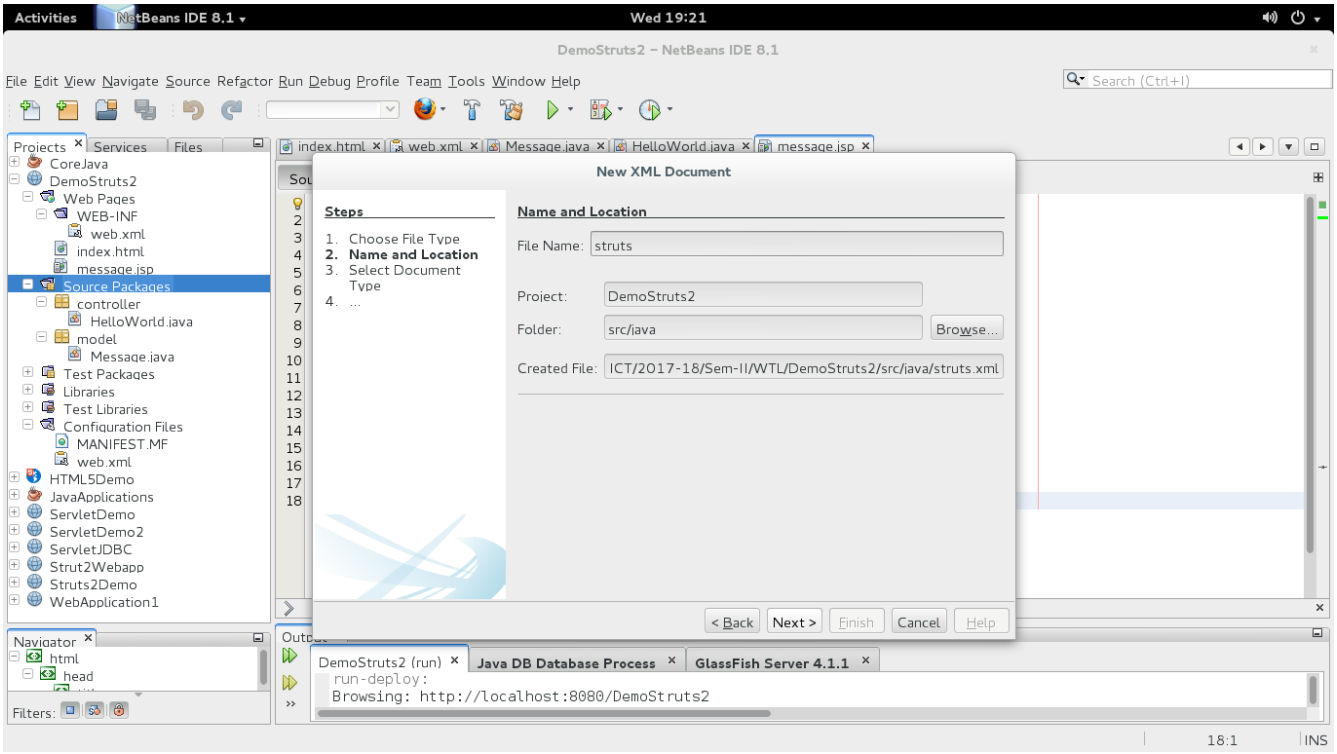
In the above example, Struts 2 property tag refers to the value **message.message**. The framework automatically translates this to the following call on the controller (action class) by convention.

HelloWorld.getMessage().getMessage()

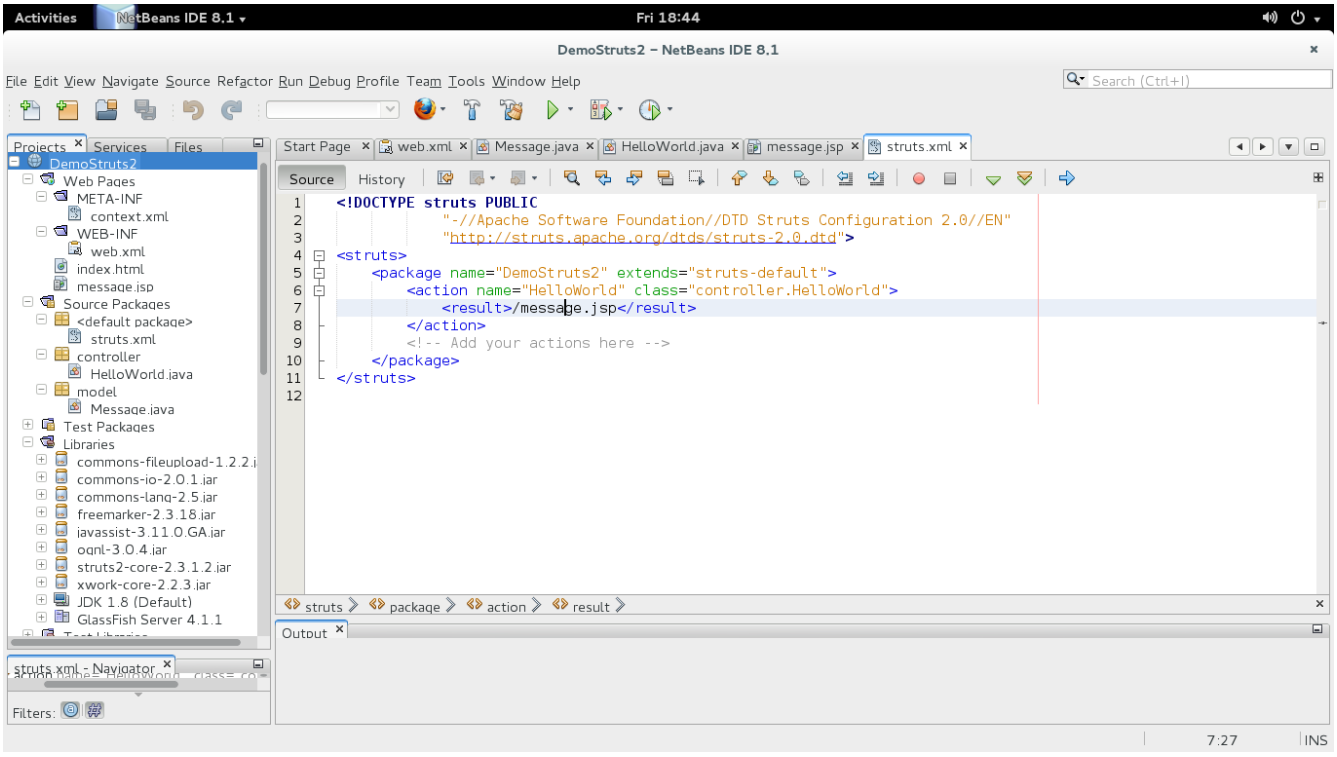
Since we have already prepared the Message object in the HelloWorld controller, we get the "Hello World!" message printed on the screen.

Finally we need to inform Struts 2 framework about our controllers, action classes and view files. This can be configured in struts.xml. This file needs to be located at the root of the source tree (it needs to be in the classpath

Add **struts.xml** to source package directory

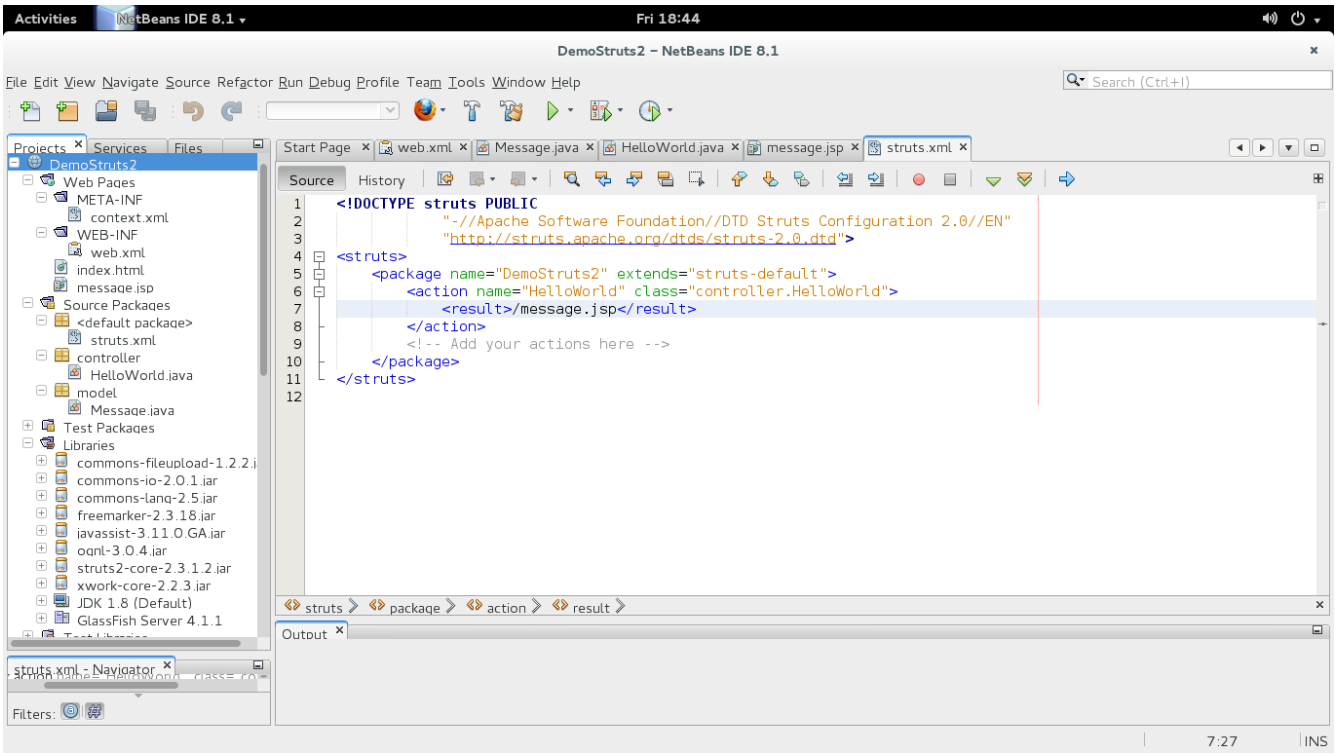


Add contents to struts.xml as shown below

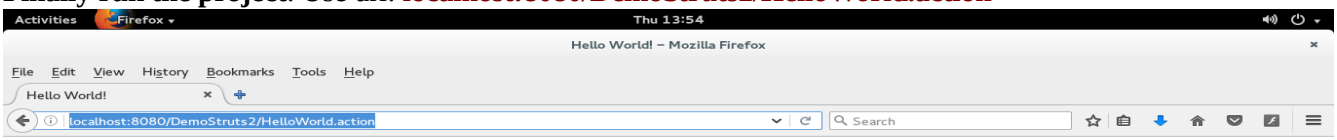


We have specified an action named HelloWorld and indicated which class file needs to be invoked for the action request. By convention, a URL request to HelloWorld.action is mapped to the action name HelloWorld. Also we have configured a result which points to our view component (message.jsp). Struts 2 framework will forward the request to message.jsp when it encounters return SUCCESS in the controller class (HelloWorld.java).

Complete directory structure at the end



Finally run the project. Use url: **localhost:8080/DemoStruts2/HelloWorld.action**



Hello World!

References:

1. https://www.javatpoint.com/struts-2-tutorial
2. https://dzone.com/tutorials/java/struts/struts-tutorial/struts-mvc-architecture-tutorial.html
3. https://www.quickprogrammingtips.com/struts2/struts-2-netbeans-tutorial.html
4. http://www.pavanjaiswal.com/2018/03/struts-2-hello-world-application-using.html

Oral Questions:

1. Explain Struts framework in brief.
2. What is Struts validator framework?
3. What is need of Struts?
4. What is includeAction?
5. What is Action class?
6. How expectations are handled in Struts application?
7. What are the classes used in Struts?
8. What is MVC?
9. Differentiate Struts1 and Struts2.
10. What are the features of Struts?

ASSIGNMENT NO: 7

TITLE	**Develop and deploy web application using AngularJS.**
PROBLEM STATEMENT /DEFINITION	Develop and deploy one of the following web applications: 1. Online pizza order application 2. Student information system for training & placement department 3. Leave management application 4. Blogging platform 5. Meeting room booking application 6. Exam cell automation application
OBJECTIVE	• To understand in depth working of AngularJS • To use AngularJS to develop any web application.
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating System: open source Fedora 20 Networked computer with Internet access Editor : IDE : Netbeans 8.1 Web browser: Mozilla Firefox, Google Chrome
REFERENCES	1. https://angular.io/ 2. https://angular.io/tutorial 3. https://www.w3schools.com/angular/angular_intro.asp 4. https://www.tutorialspoint.com/angularjs/index.htm 5. https://www.javatpoint.com/angularjs-tutorial
STEPS	Refer to steps below
INSTRUCTIONS FOR WRITING JOURNAL	• Title • Problem Statement • Theory • Design part • Troubleshooting (if any) • Conclusion • References

AngularJS

AngularJS (commonly referred to as "Angular.js" or "AngularJS 1.X") is a JavaScript-based open-source front-end web application framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications. The JavaScript components complement Apache Cordova, the framework used for developing cross-platform mobile apps. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller (MVC) and model–view–viewmodel (MVVM) architectures, along with components commonly used in rich Internet applications.

Angular (commonly referred to as "Angular 4" or "Angular 2") is a TypeScript-based open-source front-end web application platform led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.

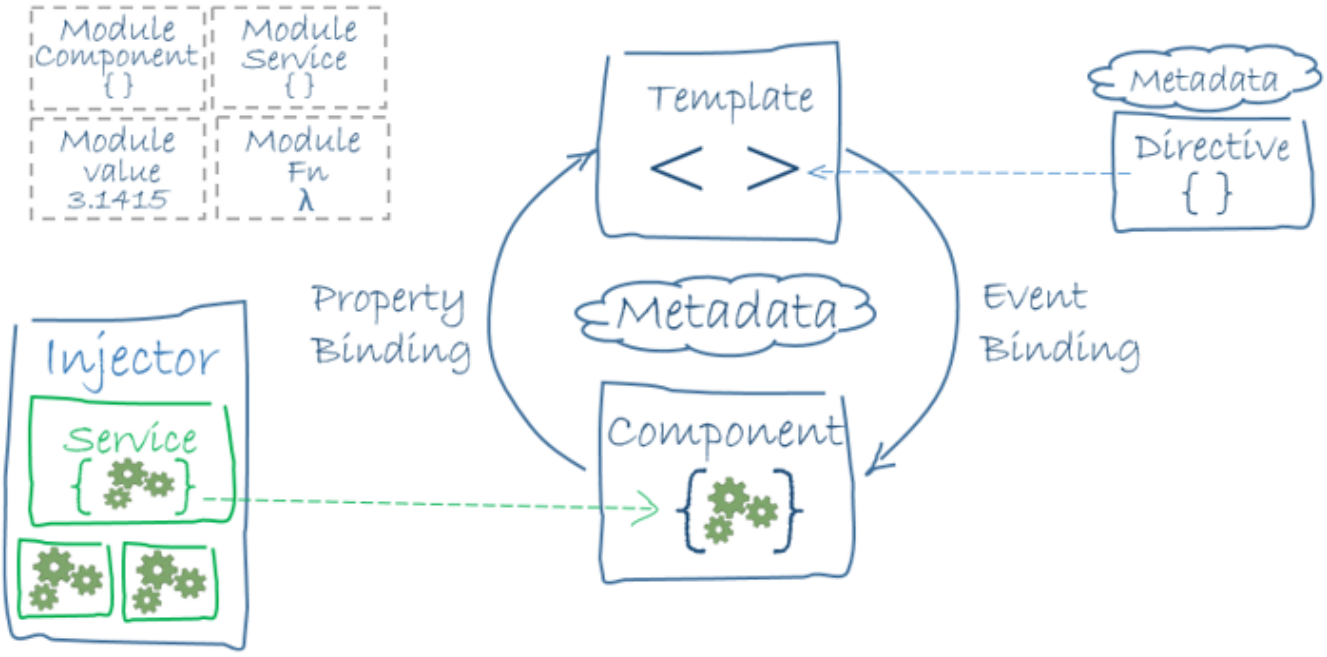
Architecture Overview

Angular is a framework for building client applications in HTML and either JavaScript or a language like TypeScript that compiles to JavaScript.

The framework consists of several libraries, some of them core and some optional.

You write Angular applications by composing HTML *templates* with Angularized markup, writing *component* classes to manage those templates, adding application logic in *services*, and boxing components and services in *modules*.

Then you launch the app by *bootstrapping* the *root module*. Angular takes over, presenting your application content in a browser and responding to user interactions according to the instructions you've provided.



Modules

Angular apps are modular and Angular has its own modularity system called *NgModules*. NgModules are a big deal. Every Angular app has at least one NgModule class, the *root module*, conventionally named AppModule. While the *root module* may be the only module in a small application, most apps have many more *feature modules*, each a cohesive block of code dedicated to an application domain, a workflow, or a closely related set of capabilities. An NgModule, whether a *root* or *feature*, is a class with an @NgModule decorator.

NgModule is a decorator function that takes a single metadata object whose properties describe the module. The most important properties are:

- declarations - the *view classes* that belong to this module. Angular has three kinds of view classes: components, directives, and pipes.
- exports - the subset of declarations that should be visible and usable in the component templates of other modules.
- imports - other modules whose exported classes are needed by component templates declared in *this* module.
- providers - creators of services that this module contributes to the global collection of services; they become accessible in all parts of the app.
- bootstrap - the main application view, called the *root component*, that hosts all other app views. Only the *root module* should set this bootstrap property.

Components

A *component* controls a patch of screen called a *view*.

For example, the following views are controlled by components:

- The app root with the navigation links.
- The list of heroes.
- The hero editor.

You define a component's application logic—what it does to support the view—inside a class. The class interacts with the view through an API of properties and methods.

```
export class HeroListComponent implements OnInit {
  heroes: Hero[];
  selectedHero: Hero;

  constructor(private service: HeroService) { }

  ngOnInit() {
    this.heroes = this.service.getHeroes();
  }

  selectHero(hero: Hero) { this.selectedHero = hero; }
}
```

Templates

You define a component's view with its companion **template**. A template is a form of HTML that tells Angular how to render the component.

```
<h2>Hero List</h2>

<p><i>Pick a hero from the list</i></p>
<ul>
  <li *ngFor="let hero of heroes" (click)="selectHero(hero)">
    {{hero.name}}
  </li>
</ul>

<app-hero-detail *ngIf="selectedHero" [hero]="selectedHero"></app-hero-detail>
```

Although this template uses typical HTML elements like <h2> and <p>, it also has some differences. Code like *ngFor, {{hero.name}}, (click), [hero], and <app-hero-detail> uses Angular's template syntax.
In the last line of the template, the <app-hero-detail> tag is a custom element that represents a new component, HeroDetailComponent.
The HeroDetailComponent is a *different* component than the HeroListComponent you've been reviewing. The HeroDetailComponent (code not shown) presents facts about a particular hero, the hero that the user selects from the list presented by the HeroListComponent. The HeroDetailComponent is a **child** of the HeroListComponent.

Metadata
Metadata tells Angular how to process a class. Looking back at the code for HeroListComponent, you can see that it's just a class. There is no evidence of a framework, no "Angular" in it at all. In fact, HeroListComponent really is *just a class*. It's not a component until you *tell Angular about it*. To tell Angular that HeroListComponent is a component, attach **metadata** to the class. In TypeScript, you attach metadata by using a **decorator**. Here's some metadata for HeroListComponent

```
@Component({
  selector:    'app-hero-list',
  templateUrl: './hero-list.component.html',
  providers:  [ HeroService ]
})
export class HeroListComponent implements OnInit {
/* . . . */
}
```

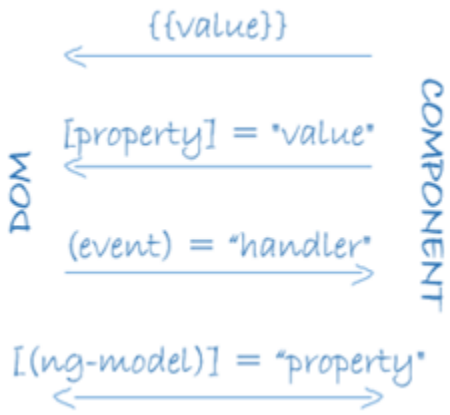
Here is the @Component decorator, which identifies the class immediately below it as a component class. The @Component decorator takes a required configuration object with the information Angular needs to create and present the component and its view.
Here are a few of the most useful @Component configuration options:

- selector: CSS selector that tells Angular to create and insert an instance of this component where it finds a <app-hero-list> tag in *parent* HTML. For example, if an app's HTML contains <app-hero-list></app-hero-list>, then Angular inserts an instance of the HeroListComponent view between those tags.
- templateUrl: module-relative address of this component's HTML template, shown above.
- providers: array of **dependency injection providers** for services that the component requires. This is one way to tell Angular that the component's constructor requires a HeroService so it can get the list of heroes to display.

The metadata in the @Component tells Angular where to get the major building blocks you specify for the component. The template, metadata, and component together describe a view. @Injectable, @Input, and @Output are a few of the more popular decorators.

Data binding

Without a framework, you would be responsible for pushing data values into the HTML controls and turning user responses into actions and value updates. Writing such push/pull logic by hand is tedious, error-prone, and a nightmare to read as any experienced jQuery programmer can attest. Angular supports **data binding**, a mechanism for coordinating parts of a template with parts of a component. Add binding markup to the template HTML to tell Angular how to connect both sides.
As the diagram shows, there are four forms of data binding syntax. Each form has a direction — to the DOM, from the DOM, or in both directions.



Directives

Angular templates are *dynamic*. When Angular renders them, it transforms the DOM according to the instructions given by **directives**. A directive is a class with a @Directive decorator. A component is a *directive-with-a-template*; a @Component decorator is actually a @Directive decorator extended with template-oriented features.
Two *other* kinds of directives exist: *structural* and *attribute* directives. They tend to appear within an element tag as attributes do, sometimes by name but more often as the target of an assignment or a binding.
Structural directives alter layout by adding, removing, and replacing elements in DOM.

Services

Service is a broad category encompassing any value, function, or feature that your application needs. Almost anything can be a service. A service is typically a class with a narrow, well-defined purpose. It should do something specific and do it well.

Examples include:

- logging service
- data service
- message bus
- tax calculator
- application configuration

There is nothing specifically *Angular* about services. Angular has no definition of a service. There is no service base class, and no place to register a service. Yet services are fundamental to any Angular application. Components are big consumers of services.

Dependency injection

Dependency injection is a way to supply a new instance of a class with the fully-formed dependencies it requires. Most dependencies are services. Angular uses dependency injection to provide new components with the services they need.

Angular can tell which services a component needs by looking at the types of its constructor parameters.

References:

1. https://angular.io/
2. https://angular.io/tutorial
3. https://www.w3schools.com/angular/angular_intro.asp
4. https://www.tutorialspoint.com/angularjs/index.htm
5. https://www.javatpoint.com/angularjs-tutorial

Oral Questions:

1. What is data binding in AngularJS?
2. What is scope in AngularJS?
3. What are the controllers in AngularJS?
4. What are the services in AngularJS?
5. What are the filters in AngularJS?
6. Explain directives in AngularJS.
7. Explain templates in AngularJS.
8. What is routing in AngularJS?
9. What is deep linking in AngularJS?
10. What are the advantages of AngularJS?
11. What are the disadvantages of AngularJS?
12. Which are the core directives of AngularJS?
13. Explain AngularJS boot process.
14. What is MVC?
15. Explain ng-app directive.
16. Explain ng-model directive.
17. Explain ng-bind directive.
18. Explain ng-controller directive.
19. How AngularJS integrates with HTML?
20. Explain ng-init directive.

21. Explain ng-repeat directive.
22. What are AngularJS expressions?
23. Explain uppercase filter.
24. Explain lowercase filter.
25. Explain currency filter.
26. Explain filter filter.
27. Explain orderby filter.
28. Explain ng-disabled directive.
29. Explain ng-show directive.
30. Explain ng-hide directive.
31. Explain ng-click directive.
32. How angular.module works?
33. How to validate data in AngularJS?
34. Explain ng-include directive.
35. How to make an ajax call using Angular JS?
36. What is use of $routeProvider in AngularJS?
37. What is $rootScope?
38. What is scope hierarchy in AngularJS?
39. What is a service?
40. What is service method?
41. What is factory method?
42. What are the differences between service and factory methods?
43. Which components can be injected as a dependency in AngularJS?
44. What is provider?
45. What is constant?
46. Is AngularJS extensible?
47. On which types of component can we create a custom directive?
48. What is internationalization? How to implement internationalization in AngularJS?

Assignment No. 8

TITLE	**Design, develop and deploy web application using EJB**
PROBLEM STATEMENT /DEFINITION	Design, develop and deploy web application using EJB, JSP & Servlet from the given list : 1. Online pizza order application 2. Student information system for training & placement department 3. Leave management application 4. Blogging platform 5. Meeting room booking application 6. Exam cell automation application
OBJECTIVE	• To understand working of EJB • To explore the usage of EJB with JSP and Servlet
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating System open source Fedora 20 Networked computer with internet access Editor : IDE : Netbeans 8.1 Web browser Mozilla Firefox, Google Chrome
REFERENCES	• http://www.ejbtutorial.com/category/ejb • https://www.javatpoint.com/ejb-tutorial • https://netbeans.org/kb/docs/javaee/entappclient.html • https://examples.javacodegeeks.com/enterprise-java/ejb3/ejb-tutorial-beginners-example/ • https://netbeans.org/kb/docs/javaee/javaee-entapp-ejb.html • https://wowjava.wordpress.com/2011/01/14/a-simple-ejb-3-0-example-with-jsp-and-servlet/ • https://www.youtube.com/watch?v=uI8TGqv-5hk&feature=related • http://www.pavanjaiswal.com/2018/03/simple-ejb-jsp-servlet-application.html
STEPS	Refer to steps below
INSTRUCTIONS FOR WRITING JOURNAL	• Title • Problem Statement • Theory • Architecture diagrams (design part) • Troubleshooting (if any) • Conclusion • References

EJB

EJB (Enterprise Java Bean) is used to develop scalable, robust and secured enterprise applications in java.

Unlike RMI, middleware services such as security, transaction management etc. are provided by **EJB Container** to all EJB applications.

The current version of EJB is EJB 3.2. The development of EJB 3 is faster than EJB 2 because of simplicity and annotations such as @EJB, @Stateless, @Stateful, @ModelDriven, @PreDestroy, @PostConstruct etc.

To run EJB application, you need an application server (EJB Container) such as Jboss, Glassfish, Weblogic, Websphere etc. It performs:

a. life cycle management,
b. security,
c. transaction management, and
d. object pooling.

EJB application is deployed on the server, so it is called server side component also.

EJB is like COM (Component Object Model) provided by Microsoft. But, it is different from Java Bean, RMI and Web Services.

When to use EJB?

1. **Application needs Remote Access**. In other words, it is distributed.
2. **Application needs to be scalable**. EJB applications supports load balancing, clustering and fail-over.
3. **Application needs encapsulated business logic**. EJB application is separated from presentation and persistent layer.

Types of EJB?

There are 3 types of enterprise bean in java.

Session Bean: Session bean contains business logic that can be invoked by local, remote or webservice client.

Message Driven Bean: Like Session Bean, it contains the business logic but it is invoked by passing message.

Entity Bean: It encapsulates the state that can be persisted in the database. It is deprecated. Now, it is replaced with JPA (Java Persistent API).

Difference between RMI and EJB

Both RMI and EJB, provides services to access an object running in another JVM (known as remote object) from another JVM. The differences between RMI and EJB are given below:

RMI	**EJB**
In RMI, middleware services such as security, transaction management, object pooling etc. need to be done by the java programmer.	In EJB, middleware services are provided by EJB Container automatically.
RMI is not a server-side component. It is not required to be deployed on the server.	EJB is a server-side component, it is required to be deployed on the server.
RMI is built on the top of socket programming.	EJB technology is built on the top of RMI.

EJB and Webservice

In EJB, bean component and bean client both must be written in java language.

If bean client need to be written in other language such as **.net**, **php** etc, we need to go with **webservices** (SOAP or REST). So EJB with web service will be better option.

Disadvantages of EJB

1. Requires application server
2. Requires only java client. For other language client, you need to go for webservice.
3. Complex to understand and develop ejb applications

Session Bean

Session bean encapsulates business logic only, it can be invoked by local, remote and webservice client.

It can be used for calculations, database access etc.

The life cycle of session bean is maintained by the application server (EJB Container).

Types of Session Bean

1. **Stateless Session Bean**: It doesn't maintain state of a client between multiple method calls.
2. **Stateful Session Bean**: It maintains state of a client across multiple requests.
3. **Singleton Session Bean**: One instance per application, it is shared between clients and supports concurrent access.

Stateless Session Bean

Stateless Session bean *is a business object that represents business logic* only. It doesn't have state (data).

In other words, *conversational state* between multiple method calls is not maintained by the container in case of stateless session bean.

The stateless bean objects are pooled by the EJB container to service the request on demand.

It can be accessed by one client at a time. In case of concurrent access, EJB container routes each request to different instance.

Annotations used in Stateless Session Bean

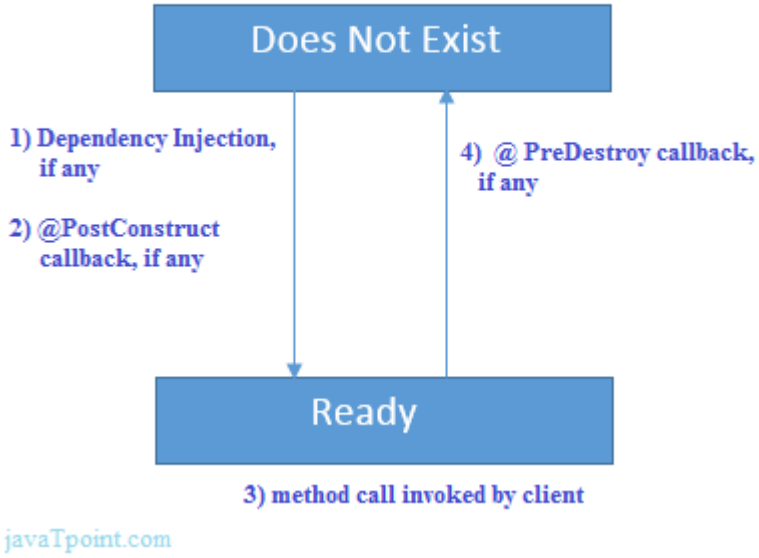
There are 3 important annotations used in stateless session bean:

1. @Stateless
2. @PostConstruct
3. @PreDestroy

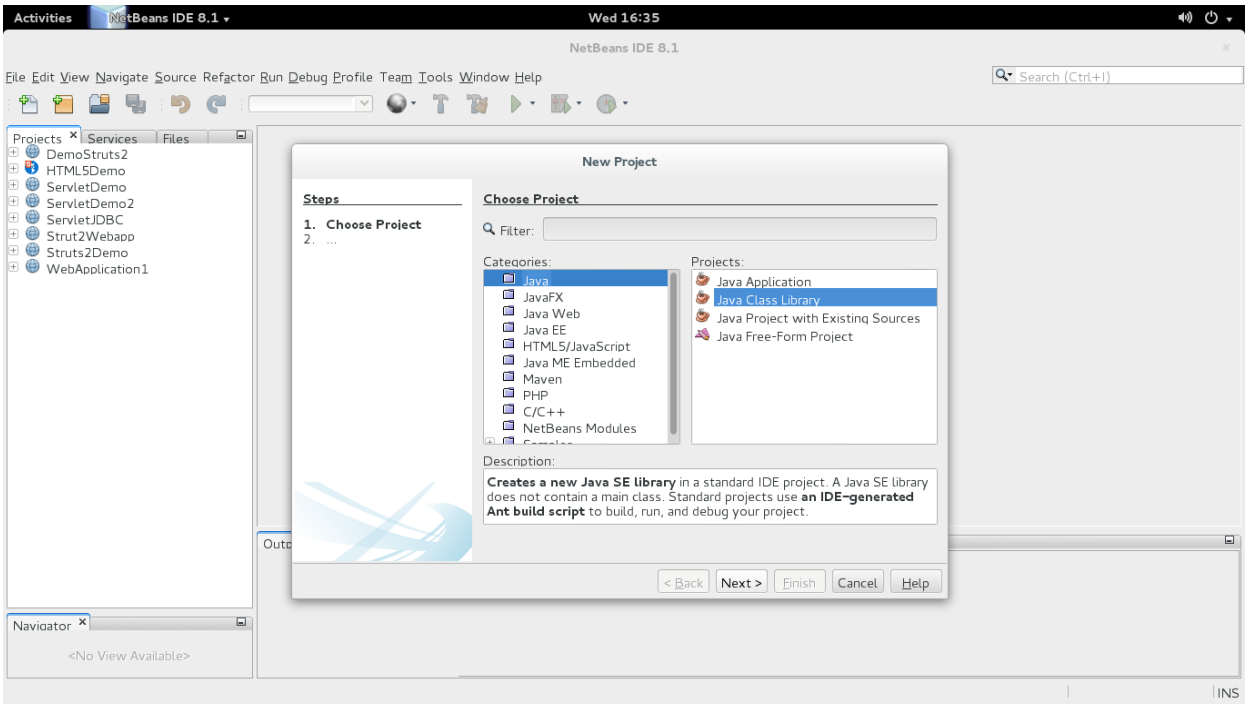
Life cycle of Stateless Session Bean

There is only two states of stateless session bean: does not exist and ready. It is explained by the figure EJB Container creates and maintains a pool of session bean first. It injects the dependency if then calls the

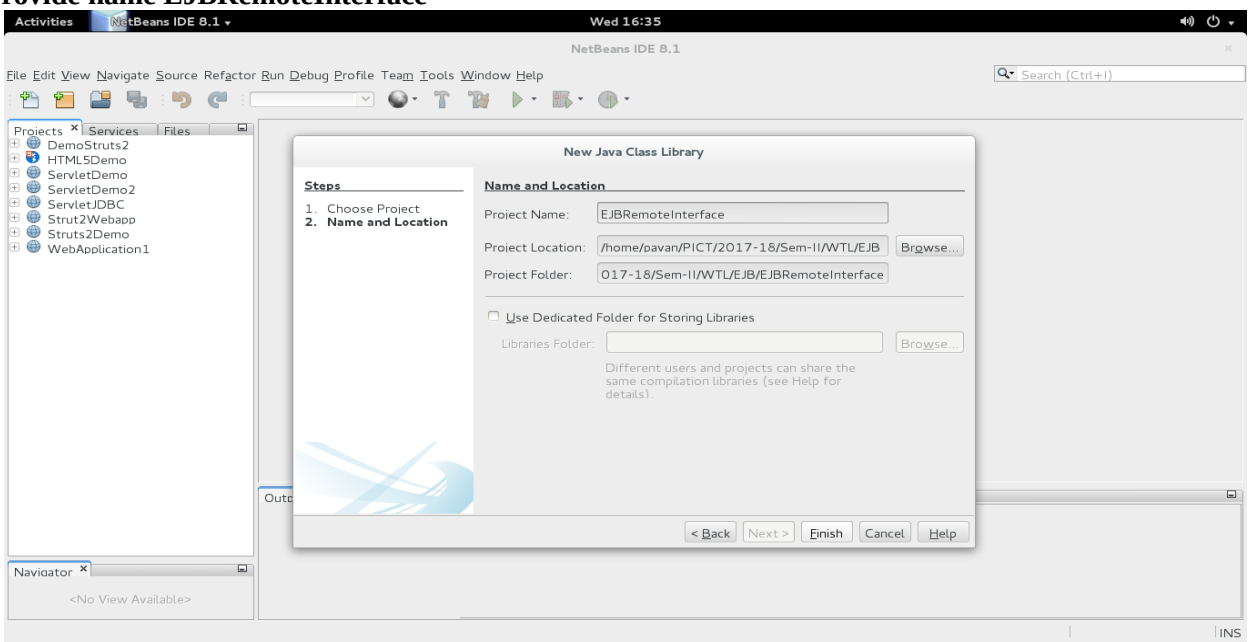
@PostConstruct method if any. Now actual business logic method is invoked by the client. Then, container calls @PreDestory method if any. Now bean is ready for garbage collection.



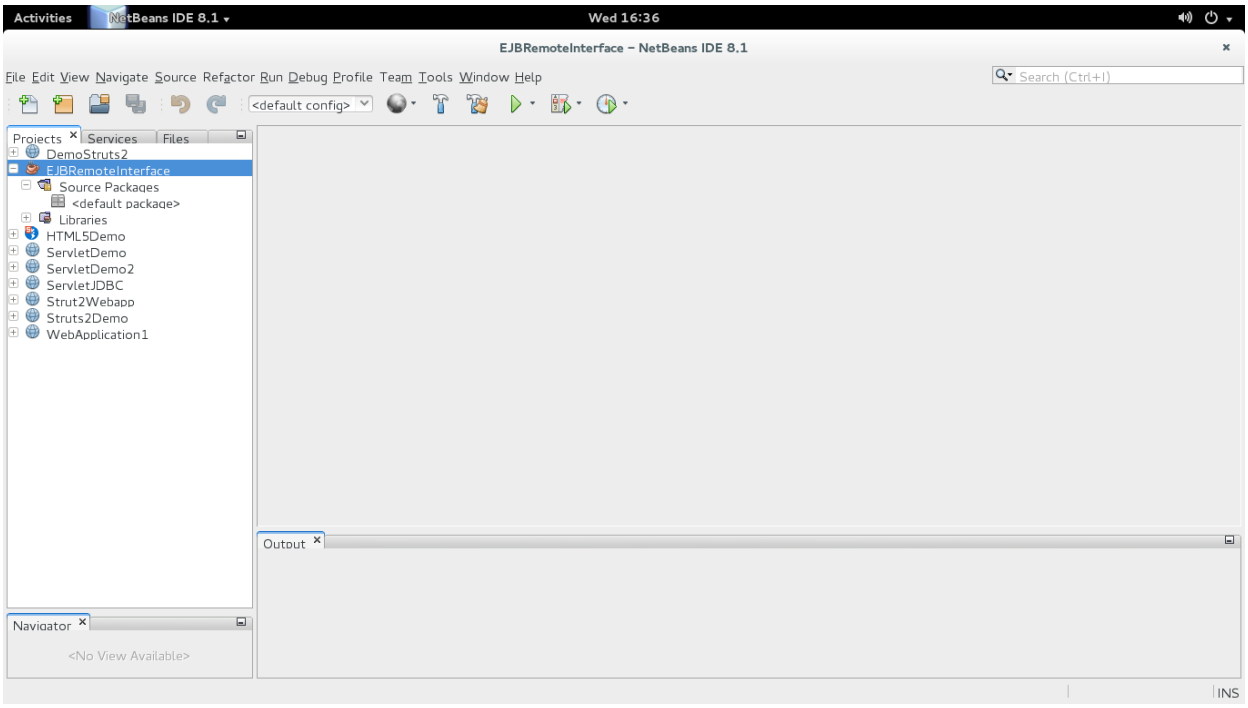
Example of Stateless Session Beans
IDE used: Netbeans 8.1
Application server: GlassFish-4.1.1
Jdk: 1.8
Operating System:Fedora
Follow the below mentioned steps. Its like **Hello World application** in EJB that take help of JSP and Servlet.
1. Create Java class library project



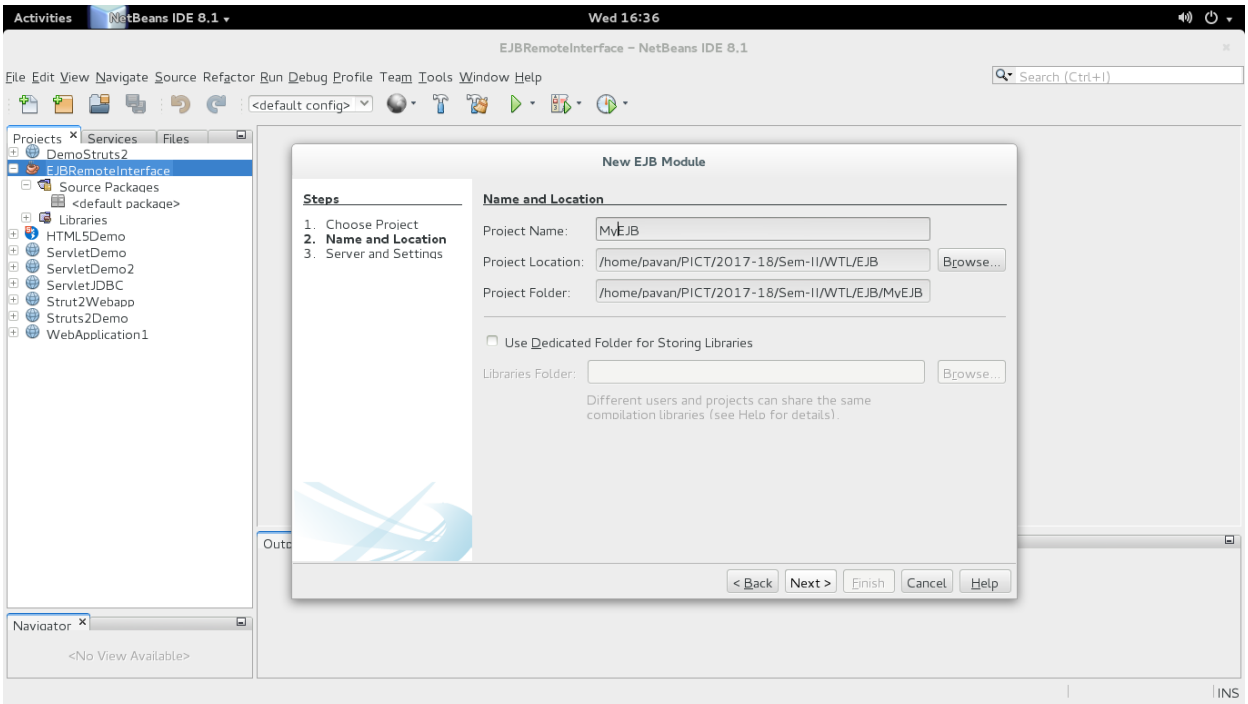
2.Provide name EJBRemoteInterface



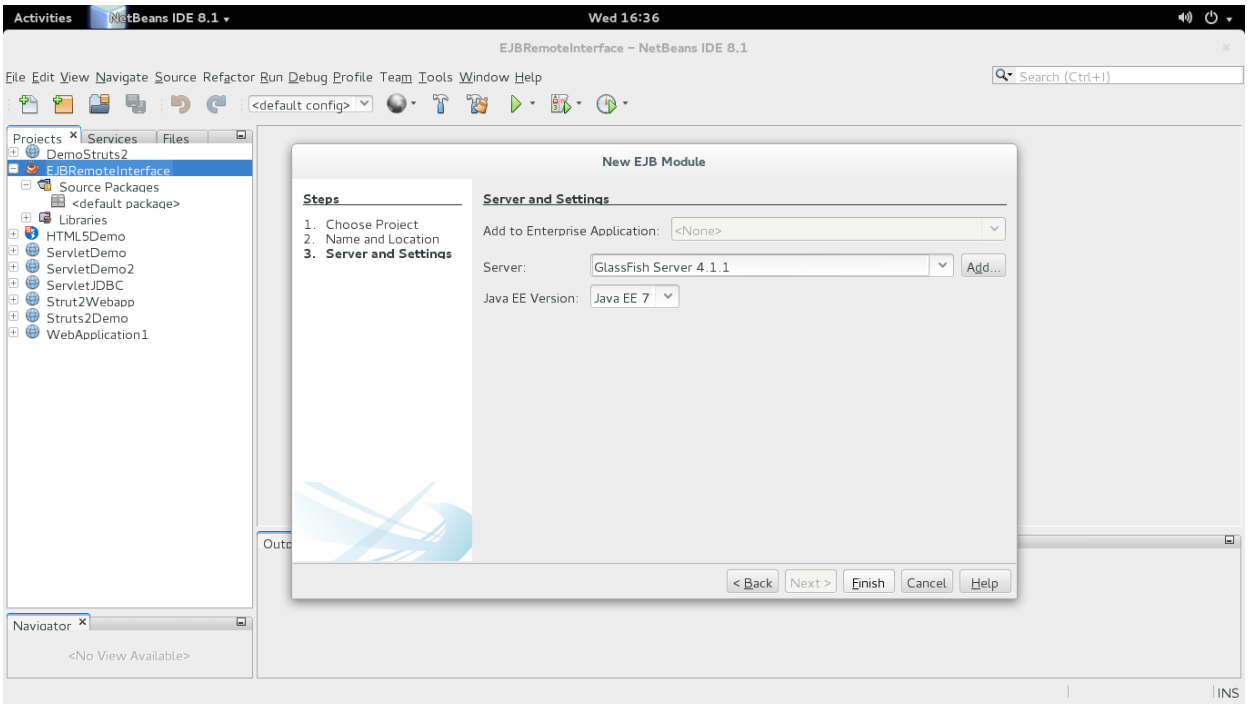
3. EJBRemoteInterface directory structure



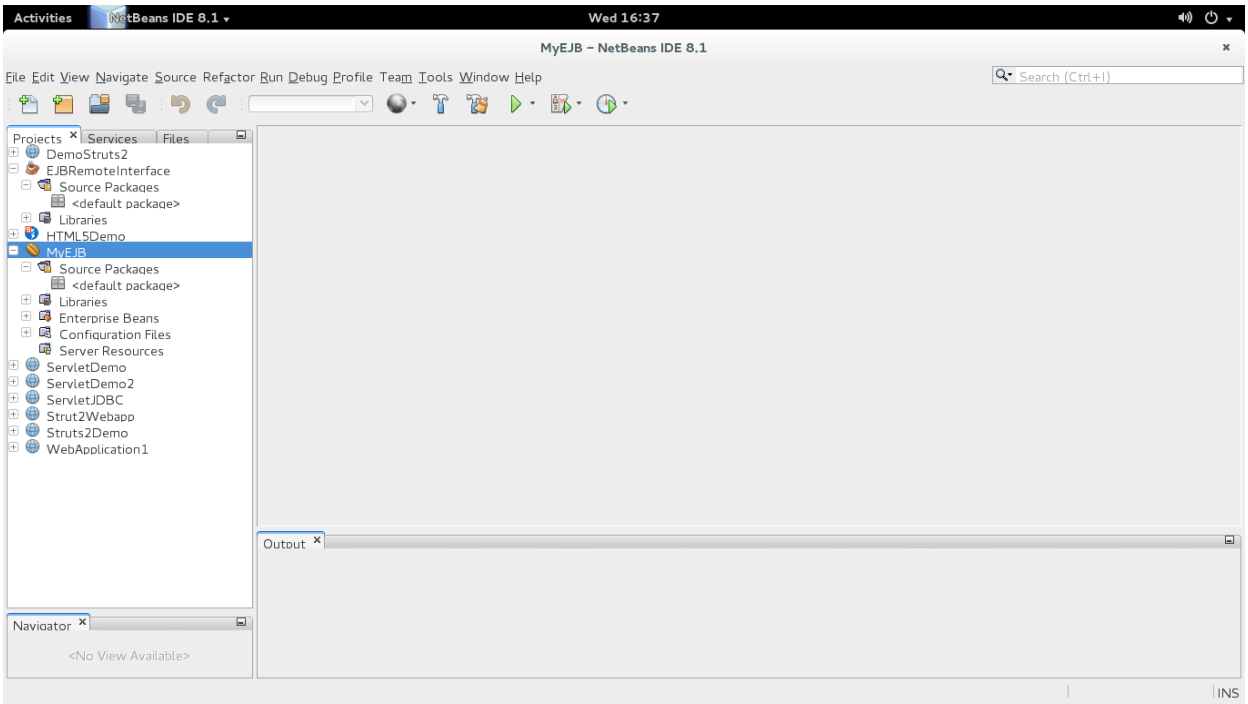
4. Create new JavaEE EJB module and provide name as MyEJB



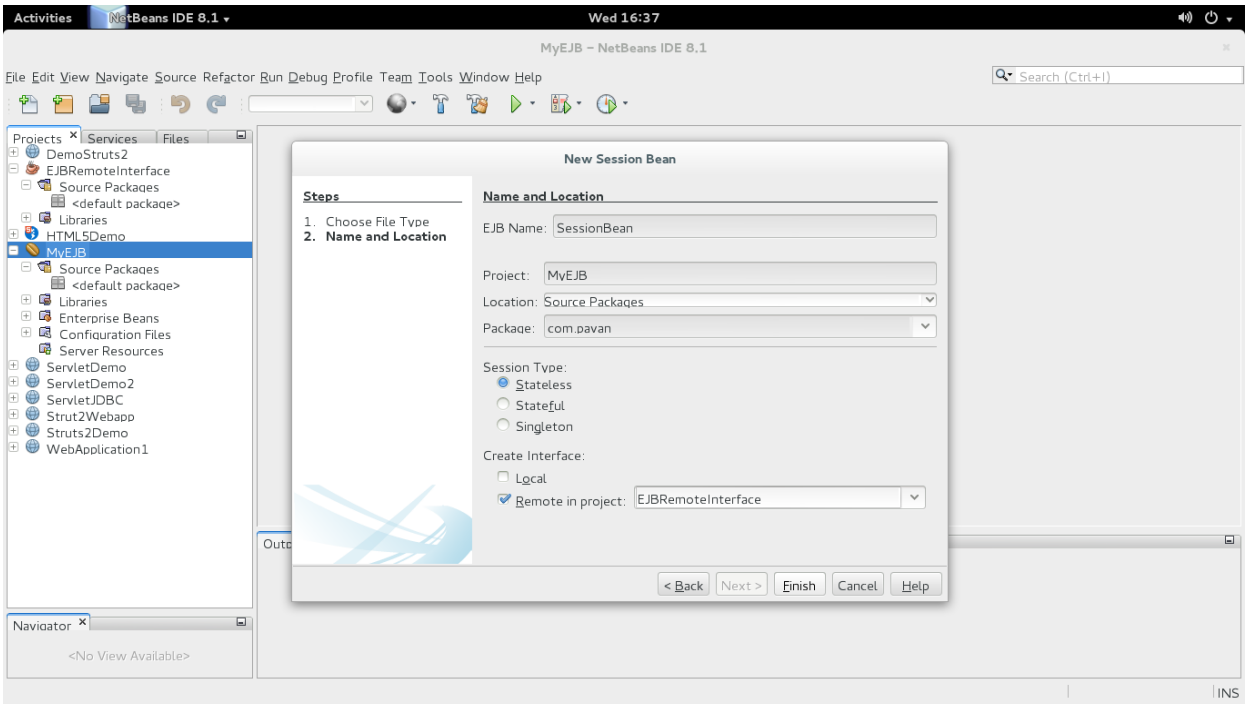
5. Select GlassFish Server



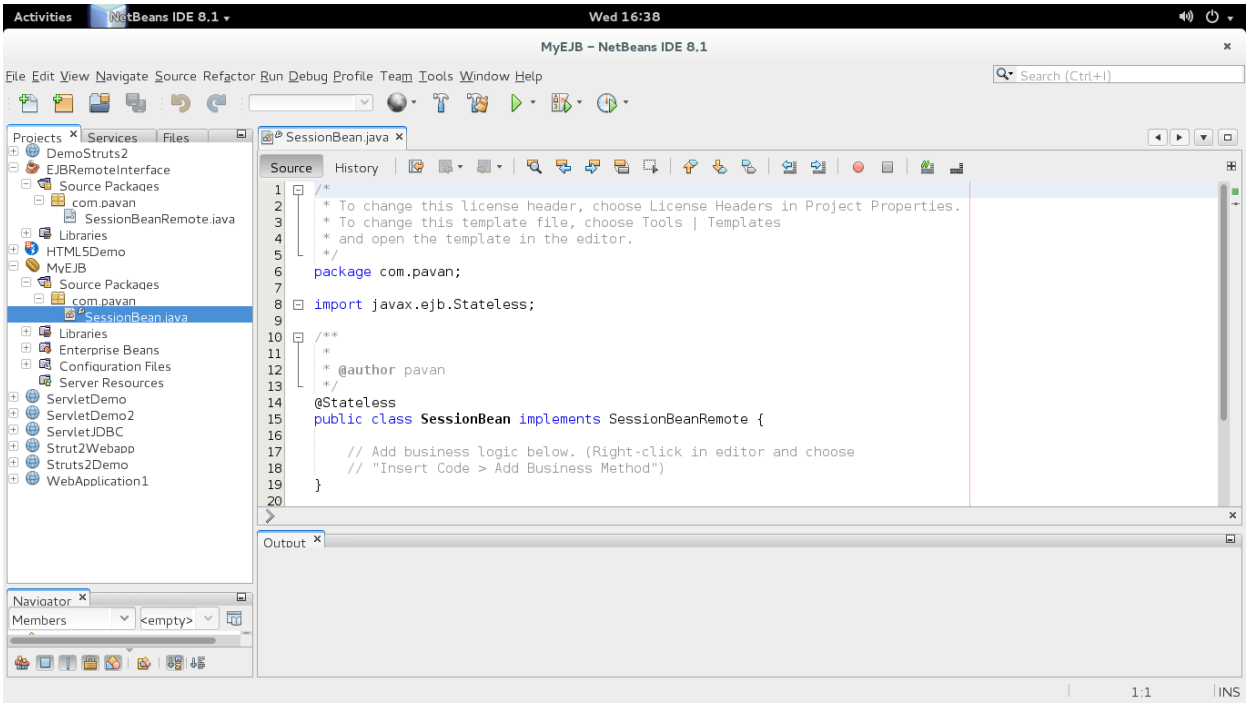
6. MyEJB directory structure



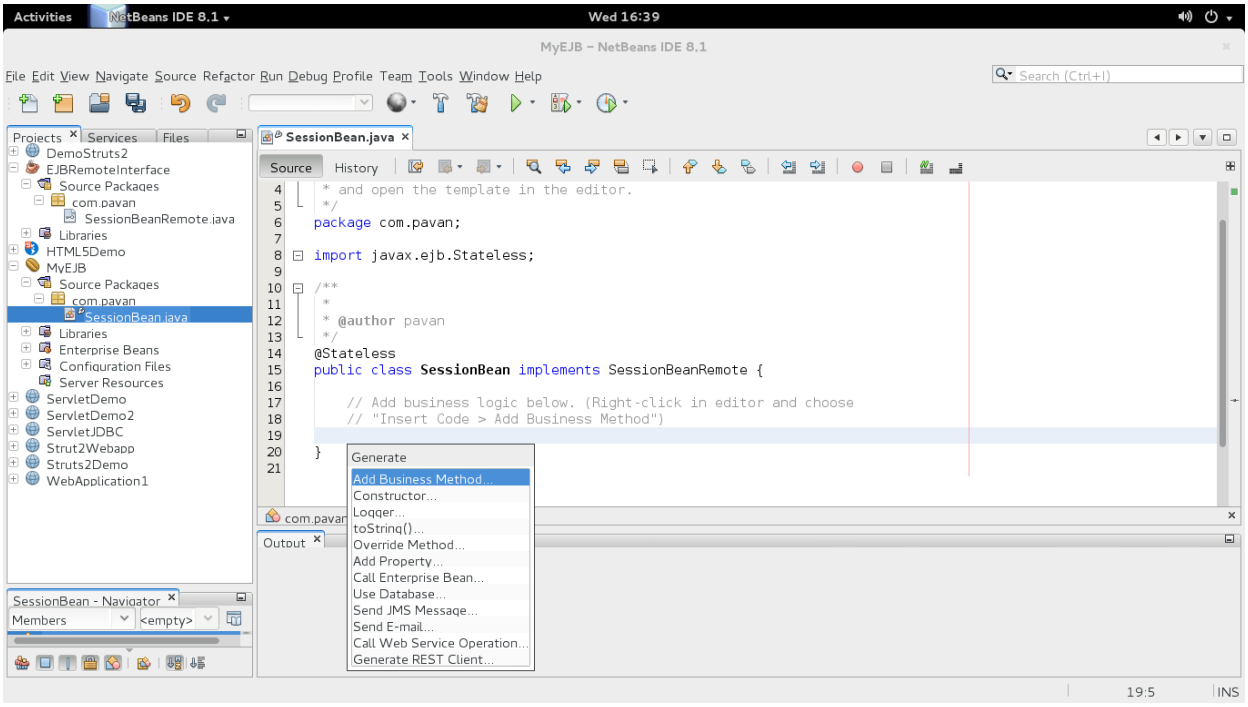
7. Right click on MyEJB and create new SessionBean having name: SessionBean



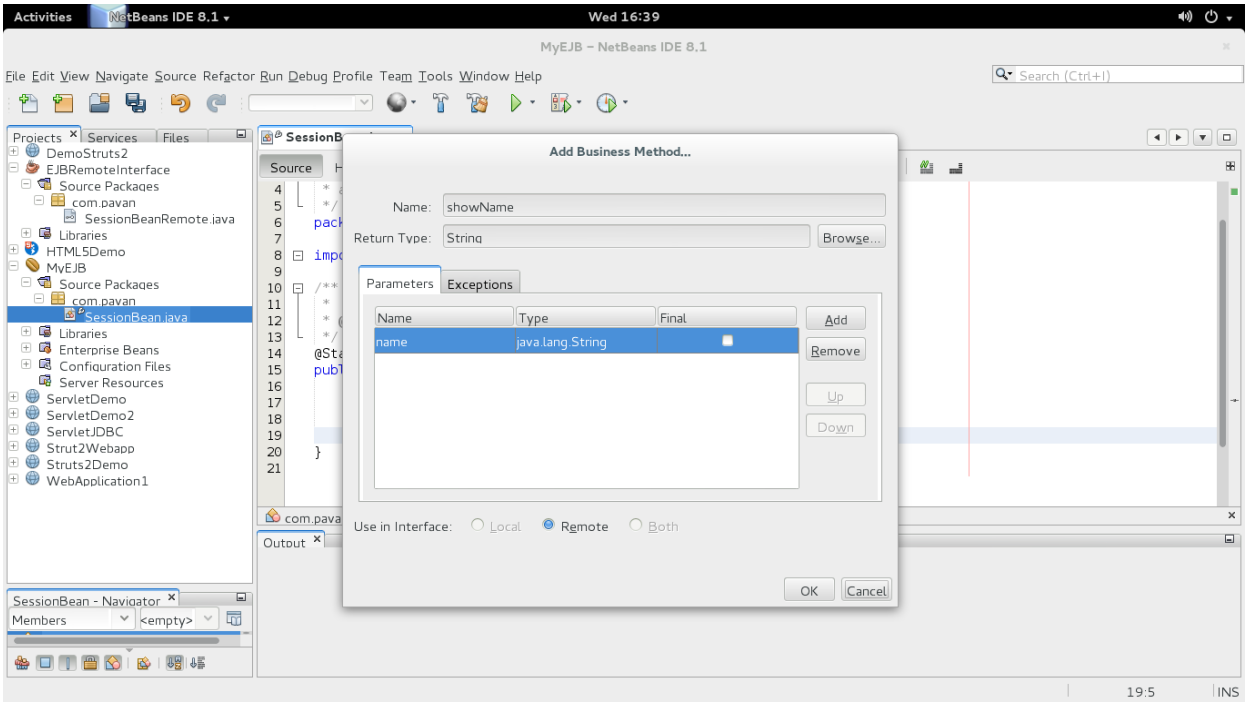
8. Empty SessionBean will look like as below



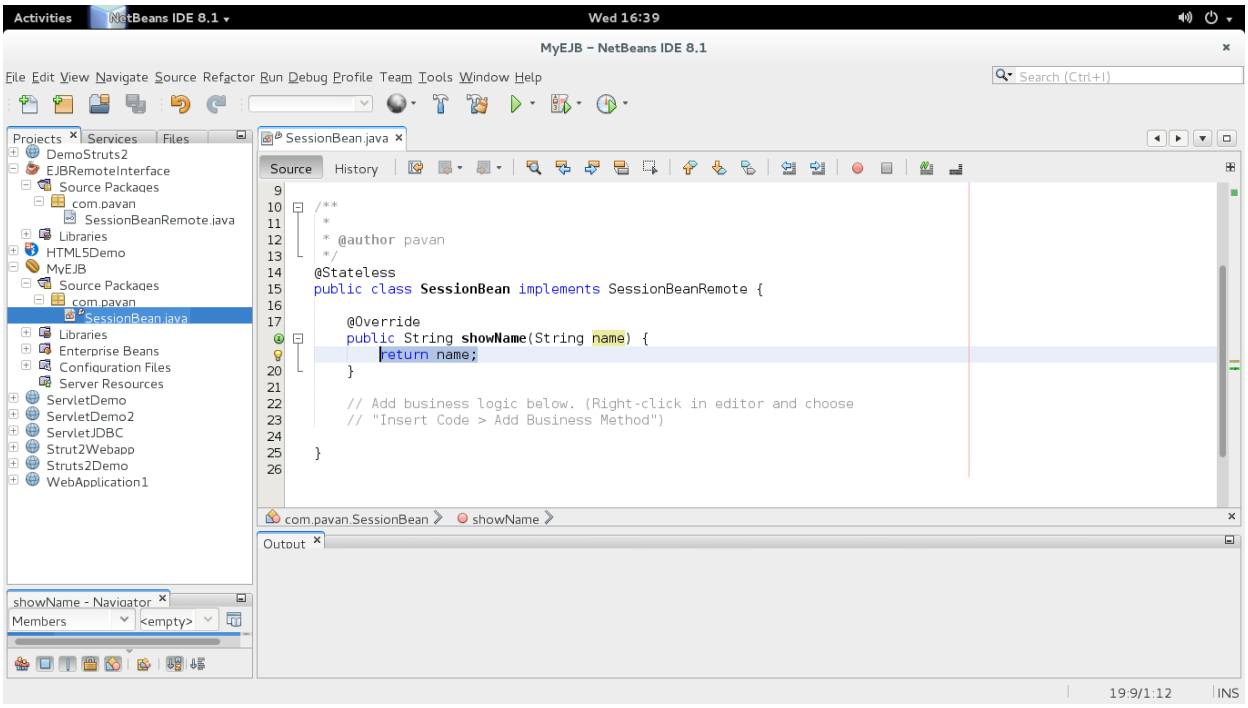
9. Add business method to newly created sessionbean by right clicking on file – insert code – business method



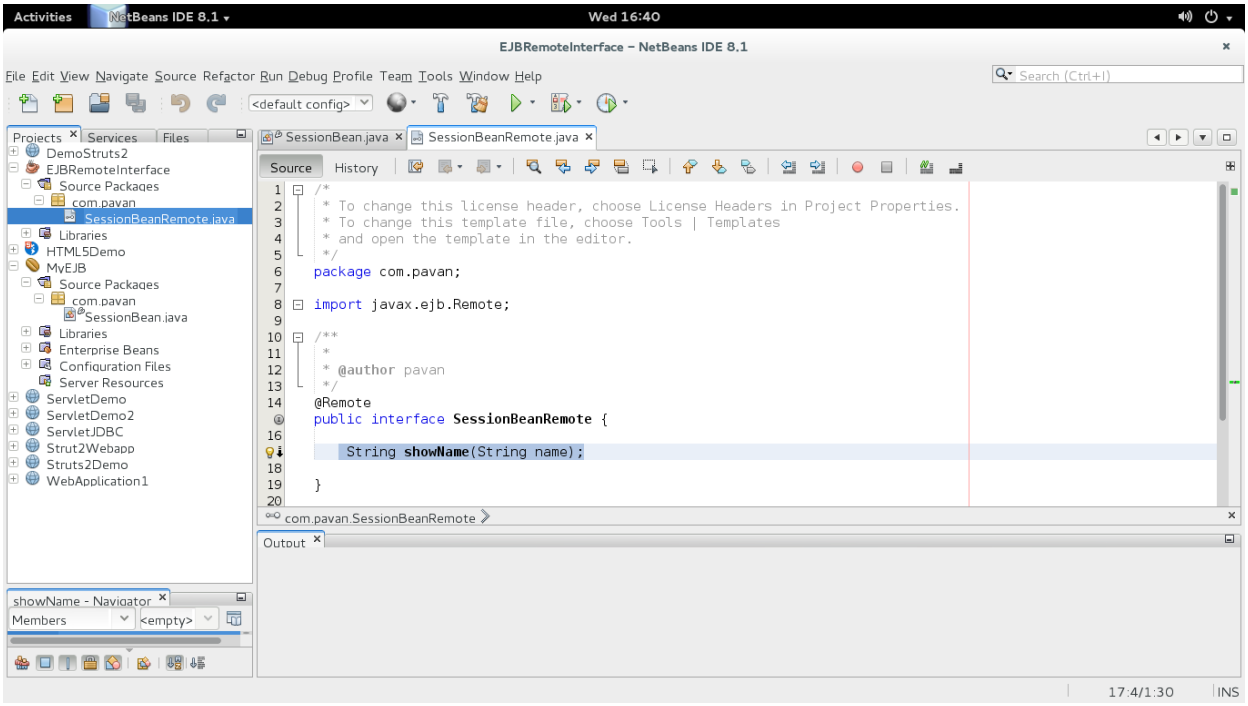
10. Providing name and parameters to business method: String showName(String)



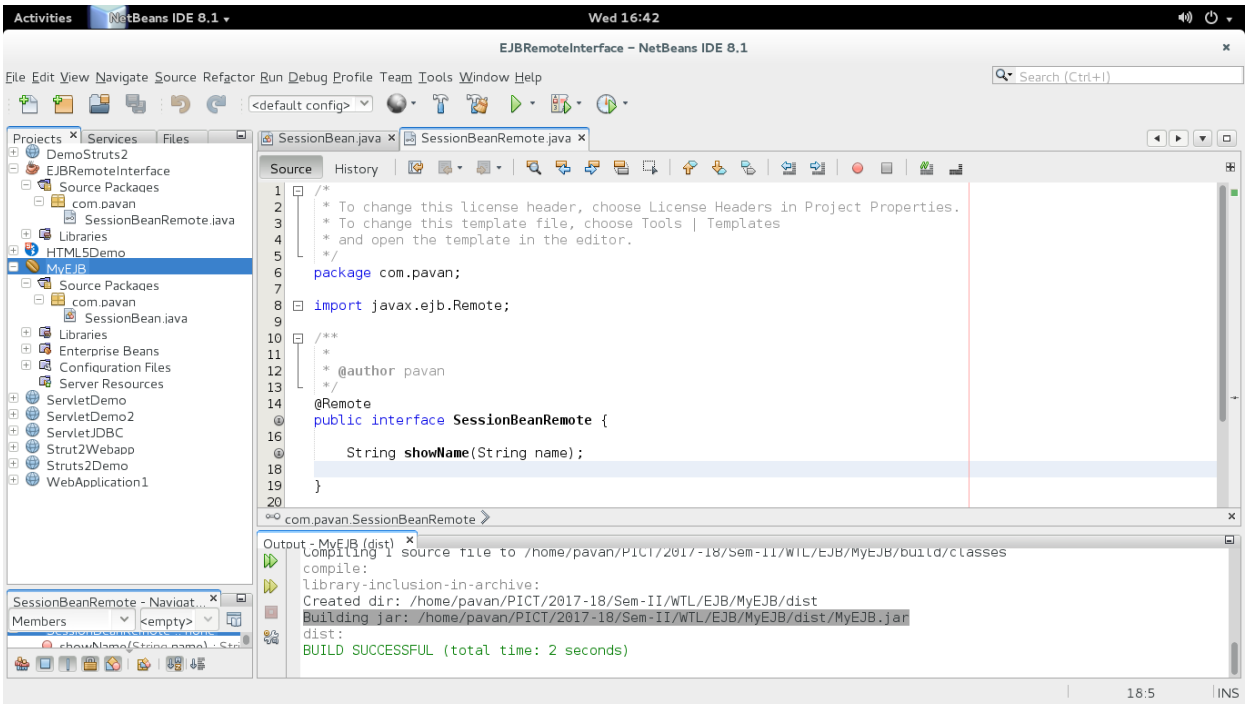
11. Final contents of SessionBean



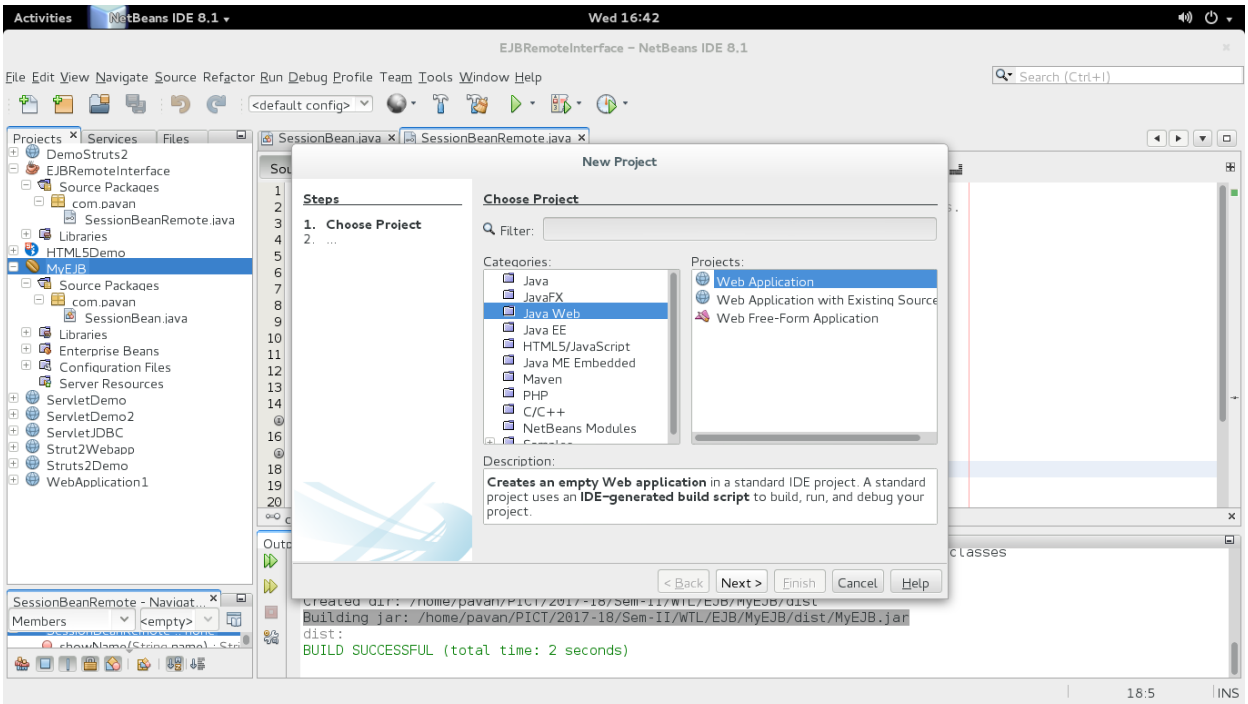
12. Contents of interface: SessionBeanRemote which has got created automatically in project: EJBRemoteInterface



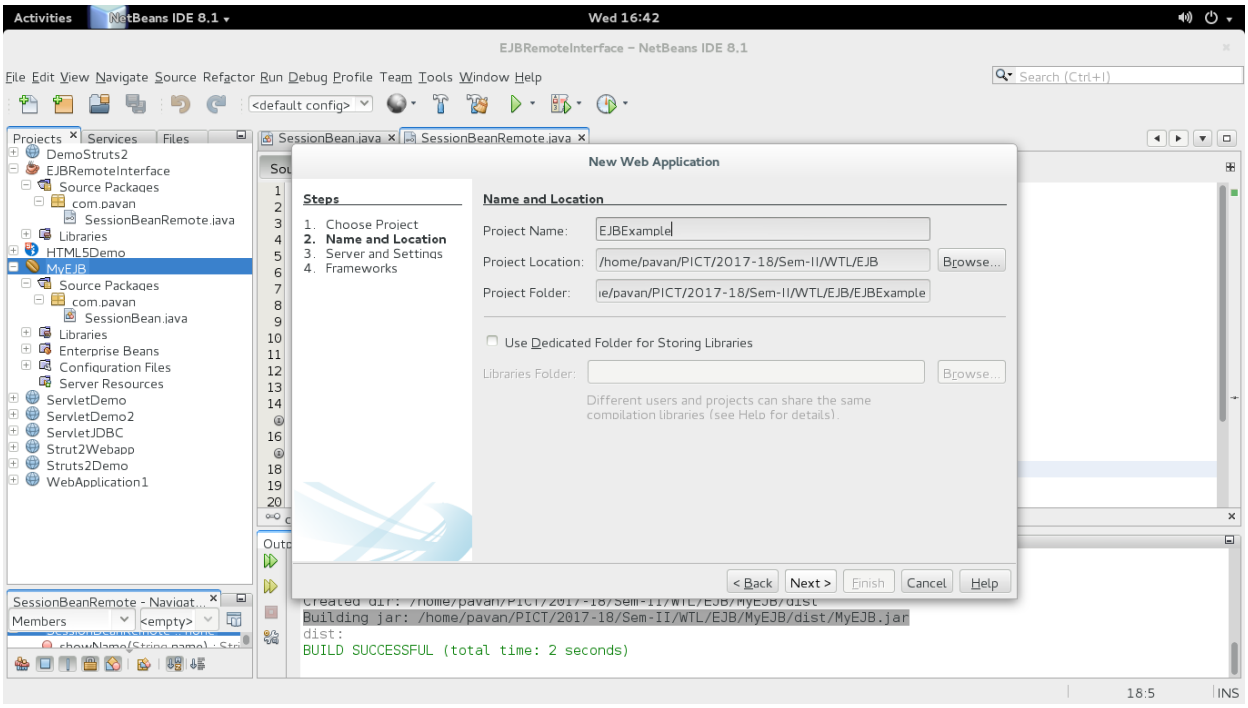
13. Build MyEJB. Right click on MyEJB and select Build option



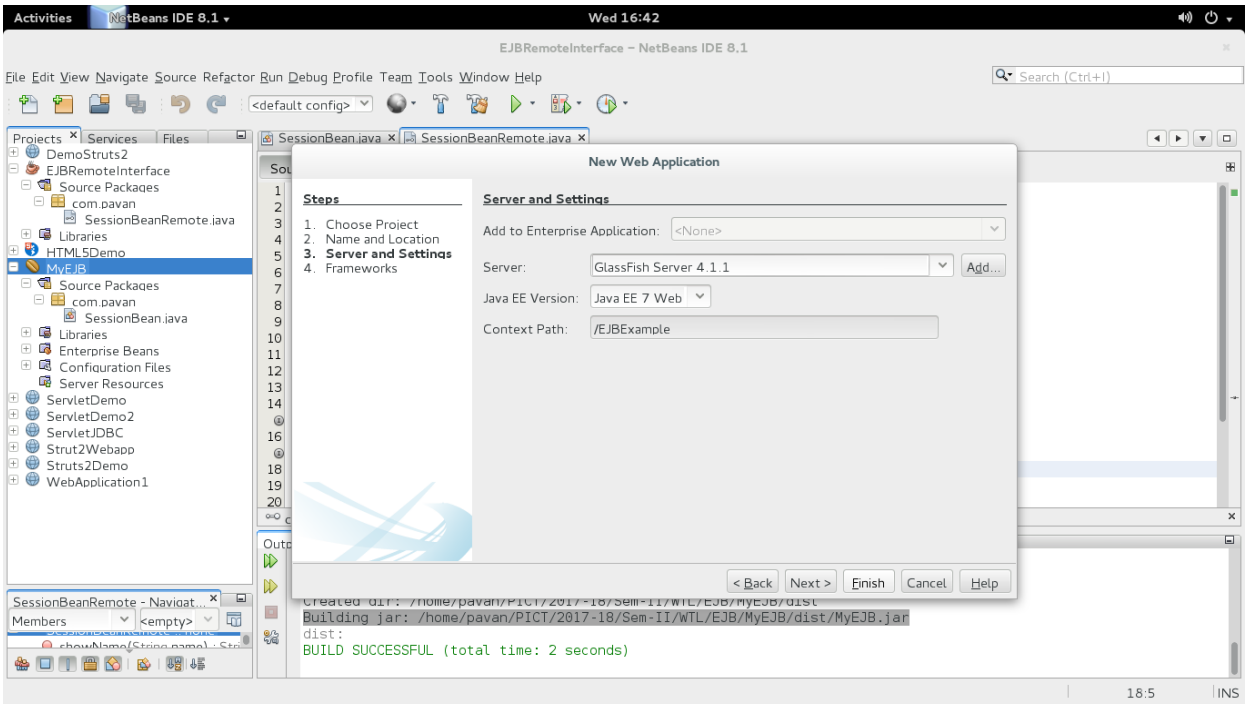
14. Create new Java Web application



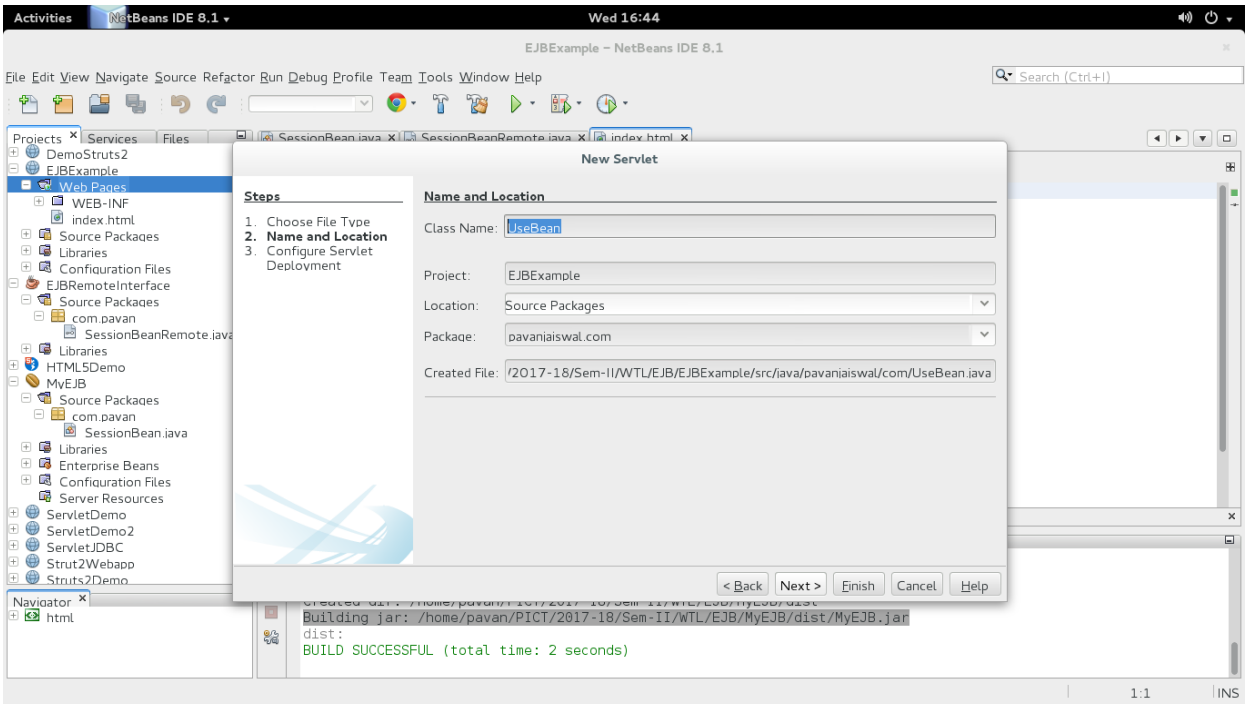
15. Provide name: EJBExample



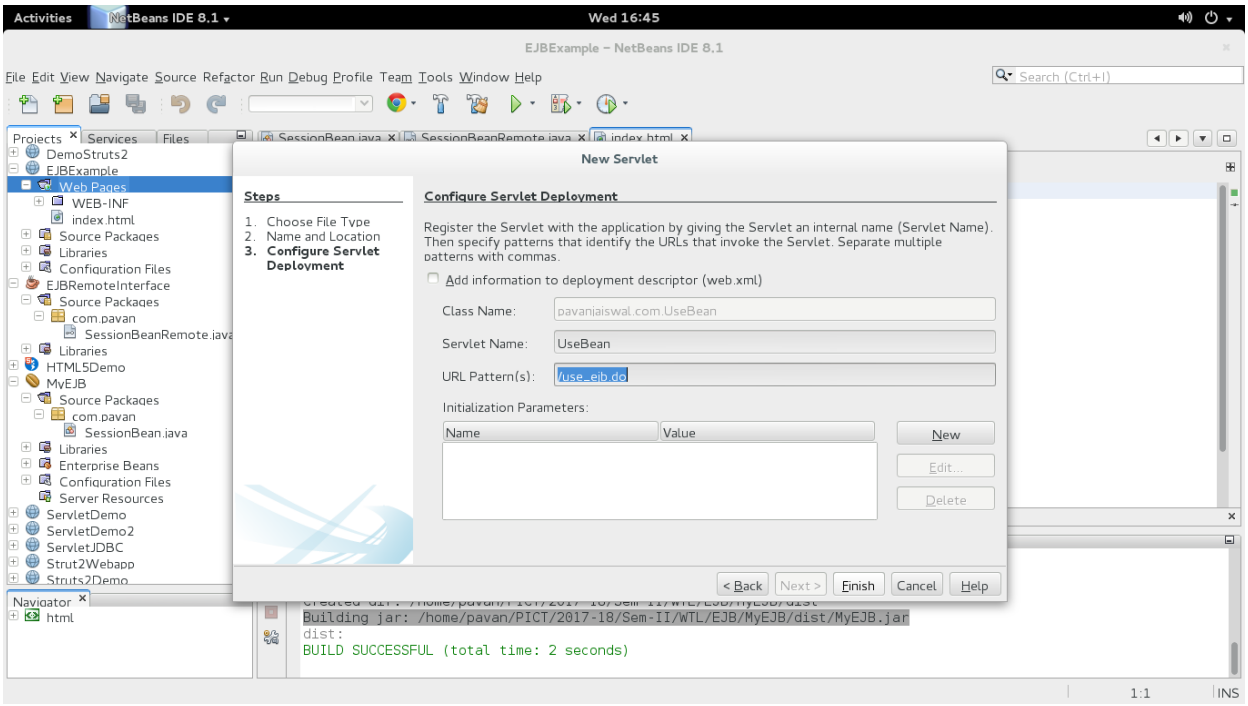
16. Glassfish server selection



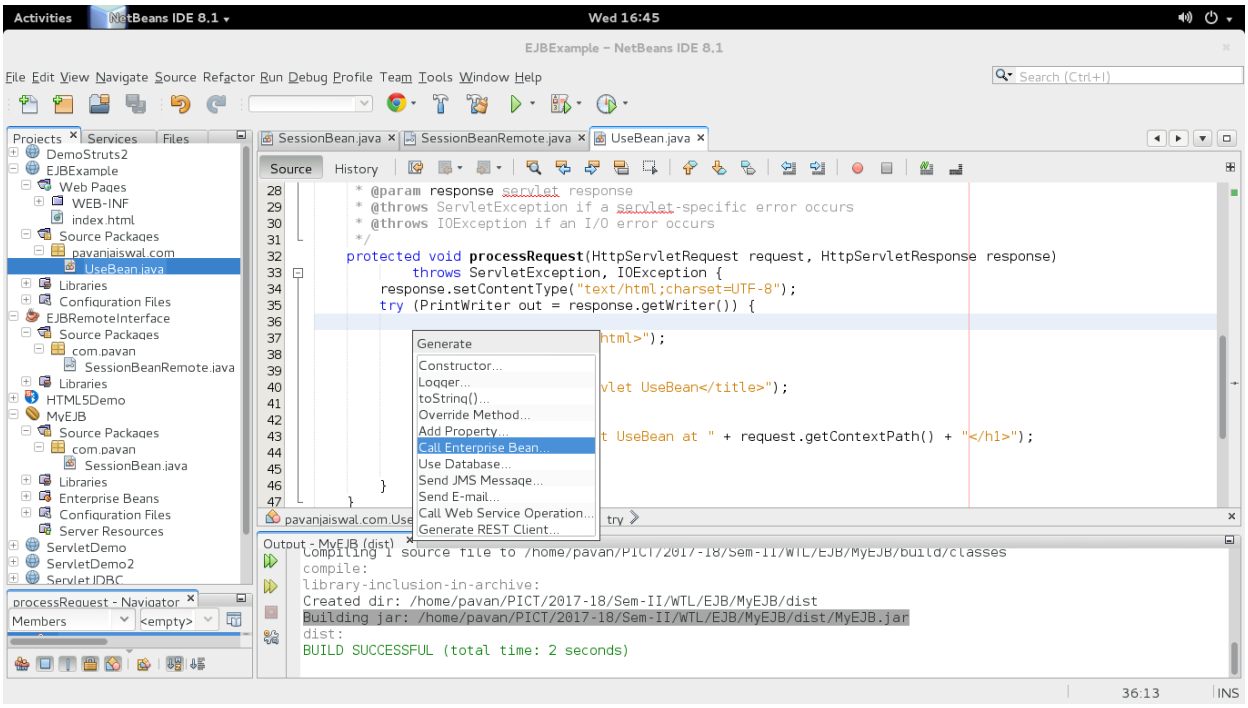
17. Create Servlet: UseBean



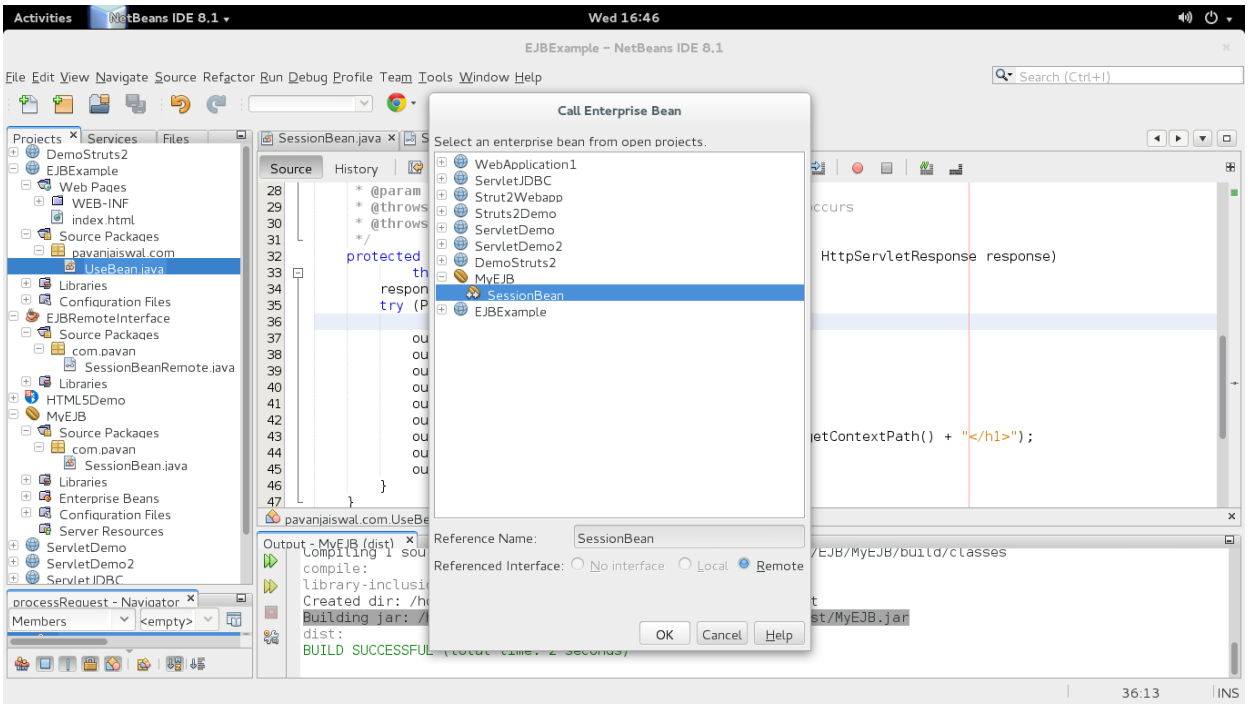
18. Provide url-pattern as: /use_ejb.do



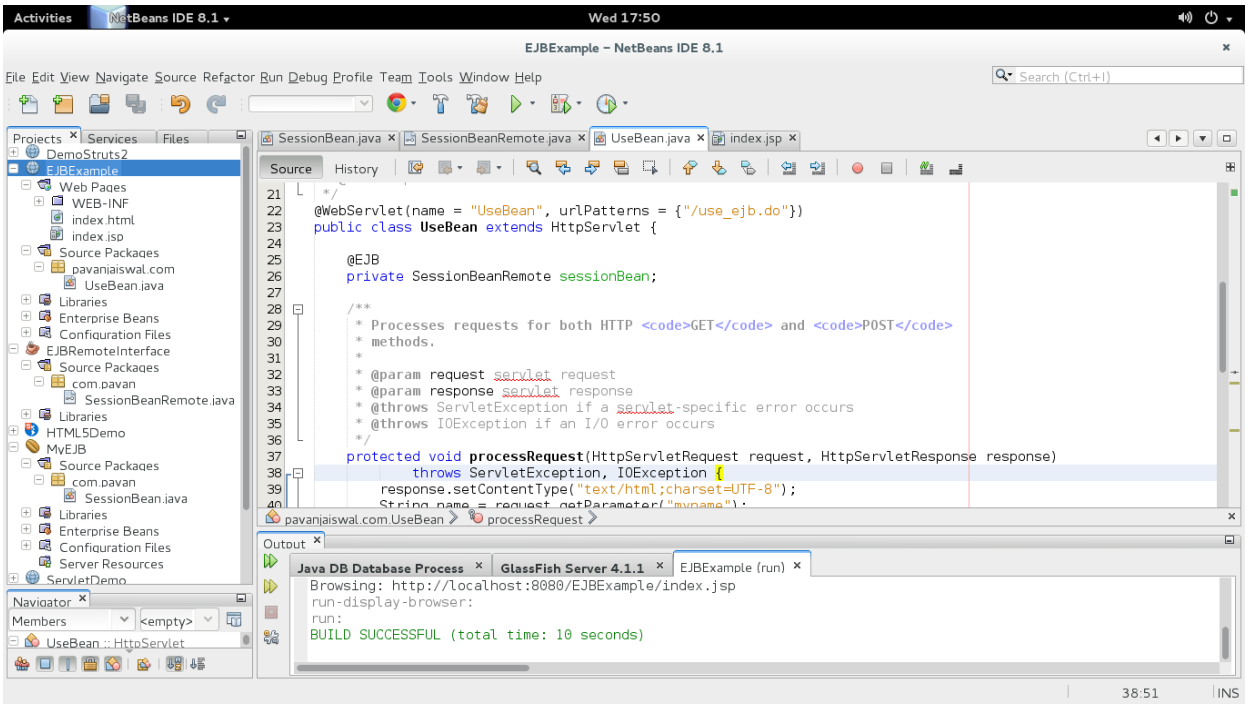
19. Call Enterprise Bean from Servlet ProcessRequest() method



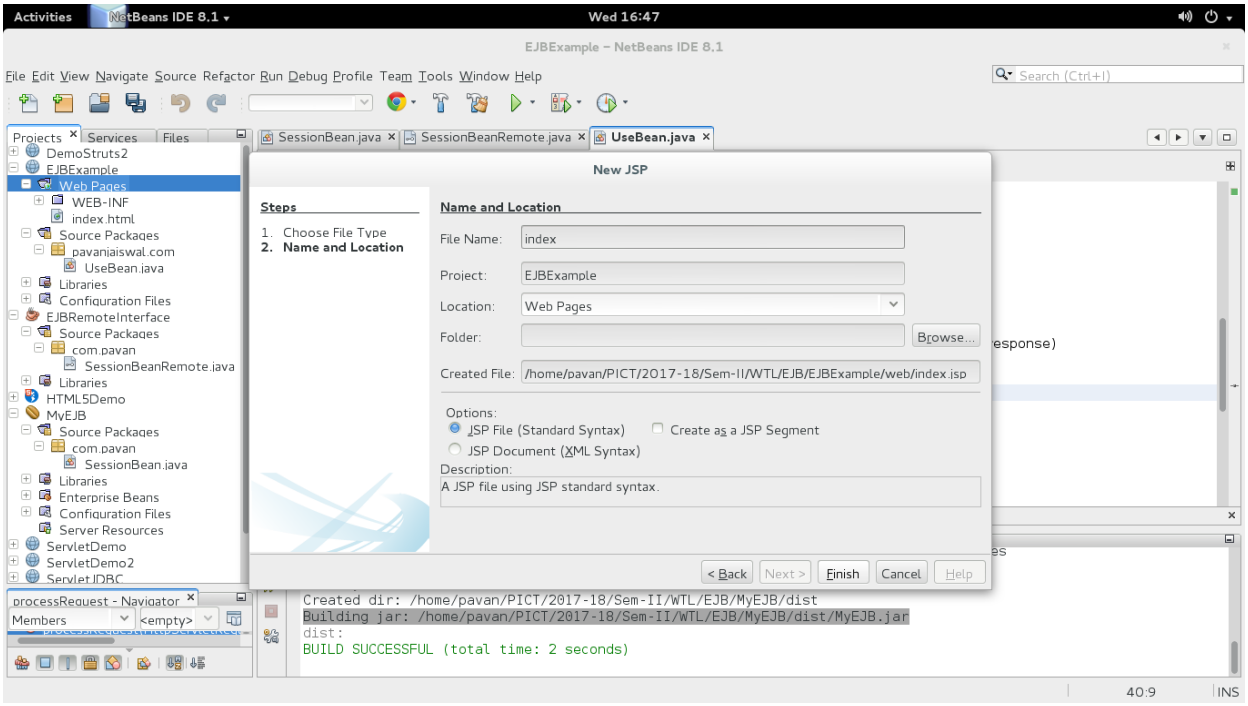
20. Select SessionBean from MyEJB drop down list



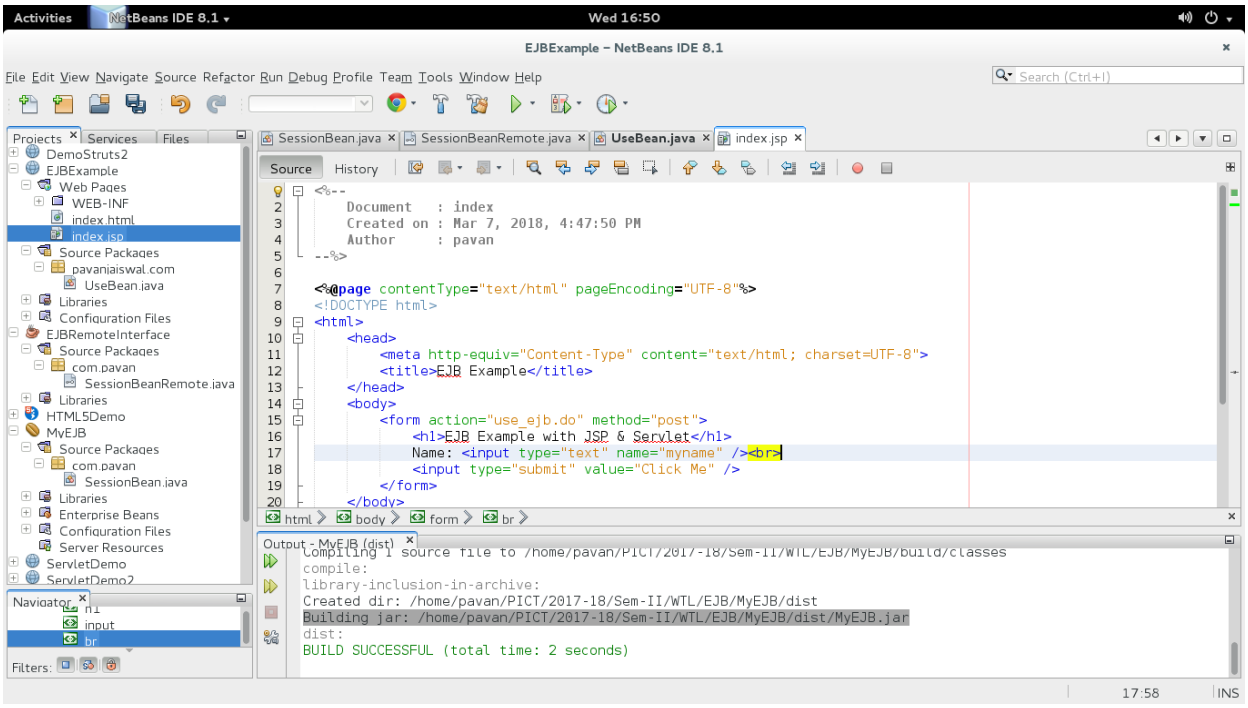
21. Instance of SessionBeanRemote created in servlet



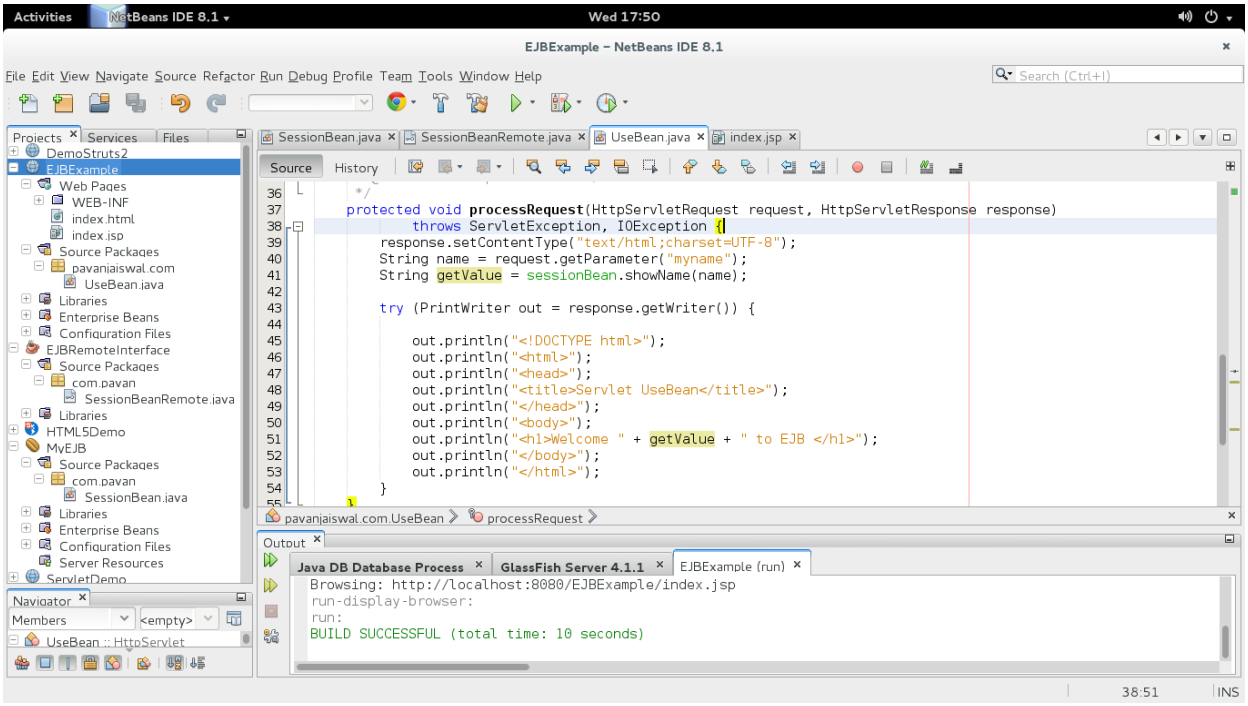
22. In EJBExample application create index.jsp which will have form to accept your name and submit event to: use_ejb.do servlet url-pattern



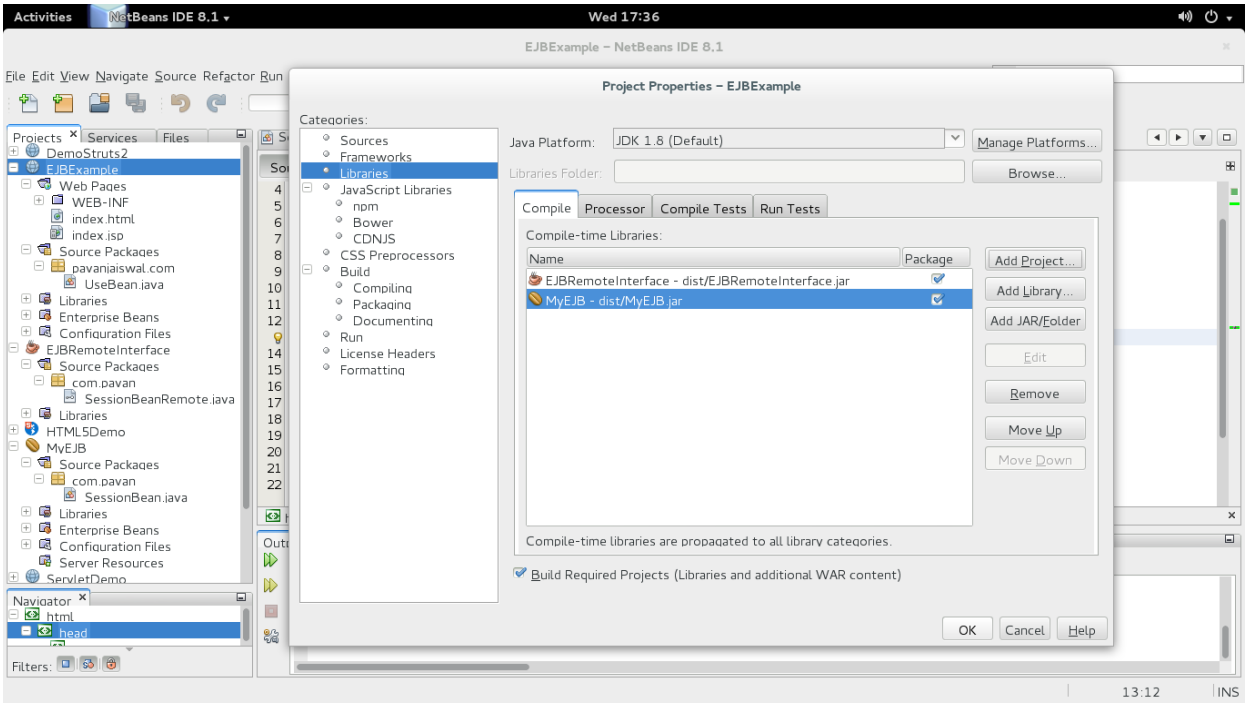
23. Contents of index.jsp



24. Contents of processRequest() method of UseBean servlet which in turns calling SessionBean method showName()



25. Right click to EJBExample – properties – library – add projects MyEJB and EJBRemoteInterface



26. Clean and build EJBExample

27. Deploy MyEJB. Just right click on it and select option Deploy

28. If everything goes fine, then run your project EJBExample. You will see index.jsp ready to accept your name in browser.

Congratulations!!! You are done with your First EJB-JSP-Servlet application.

References:

1. http://www.ejbtutorial.com/category/ejb
2. https://www.javatpoint.com/ejb-tutorial
3. https://netbeans.org/kb/docs/javaee/entappclient.html
4. https://examples.javacodegeeks.com/enterprise-java/ejb3/ejb-tutorial-beginners-example/
5. https://netbeans.org/kb/docs/javaee/javaee-entapp-ejb.html
6. https://wowjava.wordpress.com/2011/01/14/a-simple-ejb-3-0-example-with-jsp-and-servlet/
7. https://www.youtube.com/watch?v=uI8TGqv-5hk&feature=related
8. http://www.pavanjaiswal.com/2018/03/simple-ejb-jsp-servlet-application.html

Oral questions:

1. What is EJB?
2. What are the types of Enterprise Beans
3. What is session bean?
4. What is stateless session bean?
5. What is stateful session bean?
6. What is entity bean?
7. What are the benefits of EJB?
8. When to use local session bean and remote session bean?
9. Is Message Driven bean a stateless bean?
10. Explain @ annotations used in beans.