

LAB WORKBOOK

22CA1208 Mobile Application Development

Team EP

K L UNIVERSITY | Mobile Application Development



**LABORATORY WORKBOOK**

|  |  |
| --- | --- |
| STUDENT  NAME | Muthyala Madhuri |
| REG. NO | 2200520293 |
| YEAR | 1st |
| SEMESTER | 2-1 |
| SECTION | c |
| FACULTY |  |

Table of Contents

1. [Organization of the STUDENT LAB WORKBOOK 3](#_bookmark0)
2. [HTML-CSS-JS #1 6](#_bookmark1)
3. [JAVASCRIPT #2 18](#_bookmark2)
4. [JS DOM and Validations #3 25](#_bookmark3)

5. [JDBC #4 32](#_bookmark4)

1. [Servlets - 1 #5 53](#_bookmark5)
2. [Servlets - 2 #6 62](#_bookmark6)
3. [JSP(Java Server Pages) -1 #7 75](#_bookmark7)
4. JSP(Java Server Pages)-2 #8 66
5. Session Handlilng#9… 74
6. Hibernate-1#10… 83
7. HQL, HCQL #11 92
8. Inheritance Mapping #12 103
9. Spring Framework #13 118
10. Spring-Hibernate Integration #14 133
11. Spring-DAO,MVC #15 146

# Organization of the STUDENT LAB WORKBOOK

The laboratory framework includes a creative element but shifts the time-intensive aspects outside of the Two-Hour closed laboratory period. Within this structure, each laboratory includes three parts: Prelab, In-lab, and Post-lab.

* 1. **Pre-Lab**

The Prelab exercise is a homework assignment that links the lecture with the laboratory period - typically takes 2 hours to complete. The goal is to synthesize the information they learn in lecture with material from their textbook to produce a working piece of software. Prelab Students attending a two-hour closed laboratory are expected to make a good-faith effort to complete the Prelab exercise before coming to the lab. Their work need not be perfect, but their effort must be real (roughly 80 percent correct).

* 1. **In-Lab**

The In-lab section takes place during the actual laboratory period. The First hour of the laboratory period can be used to resolve any problems the students might have experienced in completing the Prelab exercises. The intent is to give constructive feedback so that students leave the lab with working Prelab software - a significant accomplishment on their part. During the second hour, students complete the In-lab exercise to reinforce the concepts learned in the Prelab. Students leave the lab having received feedback on their Prelab and In-lab work.

* 1. **Post-Lab**

The last phase of each laboratory is a homework assignment that is done following the laboratory period. In the Post-lab, students analyse the efficiency or utility of a given system call. Each Post-lab exercise should take roughly 120 minutes to complete.

2018-19 EVEN SEMESTER LAB CONTINUOUS EVALUATION

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sl No | Date | Experiment Name | Pre-Lab (5M) | In-Lab | | | | Post Lab (5M) | Viva Voce (5M) | Total (50M) | Faculty Signature |
| LOGIC (10M) | EXECUTIO N (10M) | RESUL T (10M) | ANALYS IS (5M) |
| 1 |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |

2018-19 EVEN SEMESTER LAB CONTINUOUS EVALUATION

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sl No | Date | Experiment Name | Pre-Lab (5M) | In-Lab | | | | Post Lab (5M) | Viva Voce (5M) | Total (50M) | Faculty Signature |
| LOGIC (10M) | EXECUTIO N (10M) | RESUL  T (10M) | ANALYS IS (5M) |
| 9 |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |  |
| 11 |  |  |  |  |  |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  |  |  |  |  |  |
| 13 |  |  |  |  |  |  |  |  |  |  |  |
| 14 |  |  |  |  |  |  |  |  |  |  |  |

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# HTML-CSS-JS #1

**Date of the Session: / / Time of the Session: to**

**Prerequisite:**

* **General idea of how webpages are built.**
* **General Idea on the main language or platform used for web-design.**
* **Idea on tags of HTML, CSS and JavaScript.**

## Pre-Lab Task:

1. What is the full form of HTML?

Hypertext Markup language

1. What if the full form of CSS?

Cascading Style Sheets

1. Write down the functionalities and syntax of the below mentioned tags in HTML.
   1. <!DOCTYPE>

Function : This tag is used to aspecify the version of html.

Syntax : <!DOCTYPE html>

* 1. <div>

Function: It defines a division or section within HTML document.

Syntax : <div style = “ background-color: light blue”>

* 1. <head>

Function: It defines the header of a section or webpage.

Syntax : <head> <title>Document Title</title> </head>

* 1. <body>

Funtion :It is used to define the body section of an HTML document.

Syntax : <body> Place your Content here........</body>

* 1. <h1> to <h6>

Funtion : It defines headings for an HTML document from level 1 to level 6.

Syntax : <h1>This is heading 1</h1><h2>This is heading 2</h2><h3>This is heading 3</h3><h4>This is heading 4</h4><h5>This is heading 5</h5><h6>This is heading 6</h6>

* 1. <li>

Function :It is used to represents items in list.

Syntax : <li> coffee </li>

* 1. <div>

Function : It defines a division or section within HTML document.

Syntax : <div style="border:1px solid pink;padding:20px;font-size:20px">

<p>Welcome to Javatpoint.com, Here you get tutorials on latest technologies.</p>

<p>This is second paragraph</p>

</div>

* 1. <form>

Function : It is used to define an HTML form.

Syntax : <form action="server url" method="get|post">

  //input controls e.g. textfield, textarea, radiobutton, button

</form>

* 1. <img>

Function : it is used to insert an image within an HTML document.

Syntax : <h2>HTML Image Example</h2>

<img src="good\_morning.jpg" alt="Good Morning Friends"/>

* 1. <title>

Function : It defines the title or name of an HTML document.

Syntax : <title>Document Title</title>

1. Write down the functionalities of the below mentioned in CSS.
   1. @import

The @import rule allows you to import a style sheet into another style sheet.

The @import rule must be at the top of the document (but after any [@charset](https://www.w3schools.com/cssref/pr_charset_rule.php) declaration)

The @import rule also supports media queries, so you can allow the import to be media- dependent.

* 1. padding

An element's padding is the space between its content and its border.

Padding creates extra space within an element, while margin creates extra space around an element.

* 1. background

It does not matter if one of the values above are missing, e.g. background:#ff0000 url(smiley.gif); is allowed.

If using multiple background-image sources but also want a background-color, the background-color parameter needs to be last in the list.

* 1. border

If border-color is omitted, the color applied will be the color of the text.

* 1. border-radius

The border-radius property defines the radius of the element's corners.

* 1. position

The position property specifies the type of positioning method used for an element (static, relative, absolute, fixed, or sticky).

* 1. font-family

The font-family property specifies the font for an element.

The font-family property can hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font.

Start with the font you want, and always end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

1. As you’re in the very beginning stage of learning how to build websites. Let’s begin by writing a small code in HTML. Please follow the below instructions in order to complete the task.
   1. Write the basic required syntax for the HTML document and save the file with .html format.
   2. Then, write the heading as “Basic HTML Program” into the document.
   3. Write the paragraph as “This is the trail program for building my skills in HTML”.
   4. Insert an image of your choice and specify the width to be 200px.

**Writing space for the Problem:(For Student’s use only)**

<!DOCTYPE html>

<html>

<head>

<title>Page Title</title>

</head>

<h1>Basic HTML Program</h1>

<p>This is a trail program for building my skills in HTML.</p>

<img src="1.jpg" alt="Image" style="width:200px">

</body>

</html>

## In Lab Task:

1. Ricky Dowlin and his friends are in very starting stage of learning and experimenting how to build a website, they wanted to build their personal blogs with a standard requirement as mentioned below in order to have all their personal blogs to be same.
   1. They want webpage to be segregated into 3 divisions namely header, body, footer.
   2. They want the background color to be **black**.
   3. They named their website title to be **Ricky Dowlin**.
   4. They wanted the main heading in the website to be as their name along with personal blog and wanted it to be displayed in the middle of the header division.
      1. Ex: If the name is Ricky then the heading would be **Ricky’s Personal Blog**.
   5. They wanted header color to be **skyblue** and the font color need to be **black**, they even wanted to have the font to be **helvetica**.
   6. In the body division they need to get an image inserted in the center having the dimensions of **width = 300** and **height = 300**.
   7. Now they need to write their biodata below to the image inserted.
   8. After inserting the biodata Ricky wants the user to review his blog. So he want to write a line specifying “**Please review my blog**” and place a **RATE** button below that.
   9. They wanted footer containing “**copyrights reserved** ©” in **antique white** color and wanted to align in the **center** of the footer.

**Writing space for the Problem:(For Student’s use only)**

<!DOCTYPE html>

<head>

 <title>Ricky Dowlin</title>

 <style>

 header{

 background-color: skyblue;

 background-size: cover;

 border: 4px solid white;

 }

 header >h1

{

 color: black;

}

 body {

 text-align: center;

 background-color:black;

background-position: center;

color: white;

 font-family: helvetica;

 }

p

{

 font-size: 22px;

color: antiquewhite;

}

input {

border:0;

padding: 10px;

font-size:18px;

}

Input[type=”submit”] {

background: red;

color: white;

}

 footer{

 padding: 5px;

 }

 </style>

</head>

<header>

<h1>Ricky's Blog</h1>

</header>

<body>

 <img src="1.jpg" id="p" width="300" height="300">

 <p>Hi! I'm Ricky, a NYC-based marketer. Say hello!</p>

<p>Please review my blog.</p>

 <button onclick="location.href='lab-1(in)-2.html'">Rate</button>

</body>

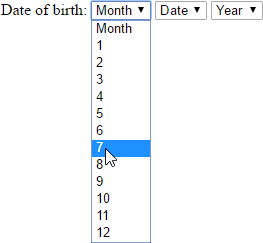
<footer>

 <p>Copyrights reserved <p>&copy;</p>

</footer>

1. After looking at your webpage they wanted you to also do the review form for their personal blog. The form should be created in another webpage. The form webpage needs to be linked by using the button naming **“Rate”** in the first webpage and should have the following details:

NAME (max 30 characters a-z and A-Z)



GENDER Male Female

RATE THE BLOG GOOD NOTBAD CAN IMPROVE

SUBMIT

**Writing space for the Problem:(For Student’s use only)**

<!DOCTYPE html>

<html>

 <head>

 <title>Ricky Downy Form</title>

<style> header{ background-color: skyblue;

 border: 4px solid white; padding: inherit;

}

 header >h1{ color: black;

 }

 Body

{

text-align: center; background-color:black;

background-position: center;

color: white;

 font-family: helvetica;

padding: 0px;

 }

 .footer{ position: fixed; left: 0;

 bottom: 0;

width: 100%;

 background-color: red;

 color: white;

 text-align: center;

}

 </style>

</head>

 <header>

 <h1>Rate the Ricky Downy's Form</h1>

</header>

 <body>

 <form>

Name:

 <input type="text" name="Name" <p>(max 30 characters a-z and A-Z) </p>

Date of birth: <input type="date" name="DOB">

 <br><br>

Gender:

 <form action="#">

 <input list="browsers" name="browser">

<datalist id="browsers">

 <option value="Male">

 <option value="Female">

<option value="Prefer not to say">

</datalist>

 <br><br>

 Rate the Blog:

 <form action="#">

<input list="Rating" name="rate">

<datalist id="Rating">

<option value="Very Bad">

 <option value="Bad">

 <option value="Average">

 <option value="Good">

 <option value="Very Good">

 </datalist>

 </form>

<br><br>

<button type="button" onclick="alert('Your form is submiited successfully..!! Thank

You...!!')">Click Me!</button>

</body>

<div class="footer">

 <p>Copyrights Reserved</p>

<p>&copy;</p>

</div>

 </html>

**Writing space for the Problem:(For Student’s use only)**

## Post Lab Task:

1. What is JavaScript and what are the advantages of JavaScript?

JavaScript is a scripting or programming language that allows you to implement complex things on web  pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes,etc. — you can bet that JavaScript is probably involved. It is the third layer of the layer cake of standard webtechnologies, two of which (HTML and CSS) we have covered in much more detail in other parts of the Learning Area.The biggest advantages to a JavaScript having a ability to produce the same result on all modern browsers. Client-Side execution: No matter where you host JavaScript, Execute always on client environment to save a bandwidth and make execution process fast.

1. What are the features of JavaScript?

Features of JavaScript are:

a)JavaScript is a object-based scripting language.

 b)Giving the user more control over the browser.

c)It Handling dates and time.

d)It Detecting the user's browser and OS,

e)It is light weighted.

f)JavaScript is a scripting language and it is not java.

g)JavaScript is interpreter based scripting language

1. Honey is learning JavaScript and she wanted to write a program in which she wants to read two number from the console and add them. After adding them she wants to check whether it is an Even number or Odd number and print the output. Help Honey by writing the code.

**Writing space for the Problem:(For Student’s use )**

var foo0 = prompt("Give me input");

var foo1 = prompt("Give me input2");

var foo2 = foo0+foo1;

if (foo2%2==0){window.alert("The number inputted is Even");

}

Else

{

 window.alert("The inputted number is odd");

 }

**Writing space for the Problem:(For Student’s use only)**

*(For Evaluator’s use only)*

Signature of the Evaluator Date of Evaluation:

Evaluator’s Observation

Marks Secured: out of

Full Name of the Evaluator:

Comment of the Evaluator (if Any)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# JAVASCRIPT #2

**Date of the Session: / / Time of the Session: to**

**Prerequisite:**

* **Basic idea on JavaScript Arrays and objects.**

## Pre-Lab Task:

**Problem Description**

1. Jack and Jill were participating in a coding rivalry. There are 4 rounds to be a finalist. In the first round the two of them were asked to illuminate the inquiry on strings wherein the inquiry was plainly referenced that the lowercase letters should be changed over as uppercase letters and the uppercase letters should be changed over into lowercase letters. In this way, would you be able to help Jack and Jill to tackle their concern by executing the code in JavaScript.

**For example,**

INPUT: **This is a Coding Competition**

OUTPUT: **tHIS IS A cODING cOMPETITION**

**Writing space for the Problem:(For Student’s use only)**

HTML FILE:

<head>

<meta charset="utf-8">

<title>JS swap case</title>

</head>

<body>

</body>

JSP FILE:

swapcase = function swapcase(str) {

 return str.replace(/([a-z]+)|([A-Z]+)/g, function(match, chr) {

 return chr ? match.toUpperCase() : match.toLowerCase();

 });

 }

console.log(swapcase('This is a Coding Competition'));

 Note: You can do in a single program

**Writing space for the Problem:(For Student’s use only)**

## In Lab Task:

1. Ibrahim is a making site to take care of mathematics issues in all respects effectively. To do this he has to make a calculator as a piece of site. Can you help him by making a calculator?
   1. Create a button for each operation.
   2. Display the obtained result on a new window.

**Writing space for the Problem:(For Student’s use only)**

<html>

 <head>

<body>

 <form name="calculator">

<table>

<tr>

 <td colspan="4">

<input type="text" name="display" id="display" disabled>

</td>

</tr>

 <tr>

 <td><input type="button" name="one" value="1" onclick="calculator.display.value += '1'"></td>

 <td><input type="button" name="two" value="2" onclick="calculator.display.value += '2'"></td>

 <td><input type="button" name="three" value="3" onclick="calculator.display.value += '3'"></td>

 <td><input type="button" class="operator" name="plus" value="+" onclick="calculator.display.value += '+'"></td>

 </tr>

<tr>

<td><input type="button" name="four" value="4" onclick="calculator.display.value += '4'"></td>

 <td><input type="button" name="five" value="5" onclick="calculator.display.value += '5'"></td>

<td><input type="button" name="six" value="6" onclick="calculator.display.value += '6'"></td>

 <td><input type="button" class="operator" name="minus" value="-" onclick="calculator.display.value += '-'"></td>

</tr> <tr> <td><input type="button" name="seven" value="7" onclick="calculator.display.value += '7'"></td>

 <td><input type="button" name="eight" value="8" onclick="calculator.display.value += '8'"></td>

 <td><input type="button" name="nine" value="9" onclick="calculator.display.value += '9'"></td>

<td><input type="button" class="operator" name="times" value="x" onclick="calculator.display.value += '\*'"></td>

 </tr>

<tr>

<td><input type="button" id="clear" name="clear" value="c"

onclick="calculator.display.value = ''"></td>

 <td><input type="button" name="zero" value="0" onclick="calculator.display.value +=

'0'"></td>

<td><input type="button" name="doit" value="=" onclick="calculator.display.value = eval(calculator.display.value)"></td>

 <td><input type="button" class="operator" name="div" value="/"

onclick="calculator.display.value += '/'"></td>

 </tr>

 </table>

 </form>

 </body>

</head>

</html>

## Post Lab Task

1. John, a curator in University of Michigan needs to sort every one of the books dependents on title of the book in Alphabetical request to keep up the record of books present in the library. He was supposed to write a program to sort all the book titles in Alphabetical order and place them accordingly. To compose the code he needs some assistance from a coder who can code. So, implement a code in JavaScript where you’ll be taking array of objects as input and sort them alphabetically based on the title of the book.

**Sample object:**

var library = [

{author: 'Bill Gates', title: 'The Road Ahead', libraryID: 1254},

{author: 'Steve Jobs', title: 'Walter Isaacson', libraryID: 4264},

{author: 'Suzanne Collins', title: 'Mockingjay: The Final Book of The Hunger Games', libraryID: 3245}

];

**Writing space for the Problem:(For Student’s use only)**

HTML FILE:-

<title>JavaScript Array objects sorting based on title </title>

<p>

 </br>

Sample object:

</br>

var library = [ { author: 'Bill Gates', title: 'The Road Ahead', libraryID: 1254},

 { author: 'Steve Jobs', title: 'Walter Isaacson', libraryID: 4264},

{ author: 'Suzanne Collins', title: 'Mockingjay: The Final Book of The Hunger Games', libraryID: 3245} ];

</p>

<div id=’sol’>

</div>

JSP FILE:-

var library = [

{ author: 'Bill Gates', title: 'The Road Ahead', libraryID: 1254},

{ author: 'Steve Jobs', title: 'Walter Isaacson', libraryID: 4264},

{ author: 'Suzanne Collins', title: 'Mockingjay: The Final Book of The Hunger Games', libraryID: 3245}

 ];

Array.prototype.sortByTitle = function (){

 if(this[0].hasOwnProperty('title') == false){ return this; }

for(var i=0; i<this.length; i++){

for(var j=0; j<this.length; j++){

 if(this[i].title < this[j].title){

 var temp = this[j];

 this[j] = this[i];

this[i] = temp;

}

 }

 }

 return this;

};

Array.prototype.printObjTitle = function() {

if(this[0].hasOwnProperty('title') == false){ return this; }

if(this[0].hasOwnProperty('author') == false){ return this; }

 var str = "";

 for(var i=0;i<this.length;i++){

str += "[ " + this[i].title + " by " + this[i].author + " ] ";

}

 return str.substring(0, str.length-1);

};

document.getElementById('sol').innerHTML = library.sortByTitle().printObjTitle();

**Writing space for the Problem:(For Student’s use only)**

*(For Evaluator’s use only)*

Signature of the Evaluator Date of Evaluation:

Evaluator’s Observation

Marks Secured: out of

Full Name of the Evaluator:

Comment of the Evaluator (if Any)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# JS DOM and Validations #3

**Date of the Session: / / Time of the Session: to**

**Prerequisite:**

* **General Idea on JavaScript DOM and Validations.**
* **Idea on Event Listener and onchange Events.**

## Pre-Lab Task:

1. Drishti and Shiv are playing Ludo and they lost their dice. Since Drishti is a coder she thought to write a code in JavaScript in which it generated a random number between 1 to 6 on clicking a button, and she wanted to add a feature that when the number generated is odd, the background changes to red otherwise the background changes to green. Assuming you as Drishti implement a code in JavaScript.

**Writing space for the Problem:(For Student’s use only)**

var library = [

{ author: 'Bill Gates', title: 'The Road Ahead', libraryID: 1254},

 { author: 'Steve Jobs', title: 'Walter Isaacson', libraryID: 4264},

{ author: 'Suzanne Collins', title: 'Mockingjay: The Final Book of The Hunger Games', libraryID:

3245}

 ];

Array.prototype.sortByTitle = function (){

 if(this[0].hasOwnProperty('title') == false){ return this; }

 for(var i=0; i<this.length; i++){

 for(var j=0; j<this.length; j++){

if(this[i].title < this[j].title){

var temp = this[j];

 this[j] = this[i];

this[i] = temp;

}

 }

 }

 return this;

};

Array.prototype.printObjTitle = function() {

 if(this[0].hasOwnProperty('title') == false){ return this; }

 if(this[0].hasOwnProperty('author') == false){ return this; }

var str = "";

 for(var i=0;i<this.length;i++){

 str += "[ " + this[i].title + " by " + this[i].author + " ] ";

}



 return str.substring(0, str.length-1);

};

Document.getElementById(‘sol’).innerHTML=library.sortByTitle().printObjTitle()

**Writing space for the Problem:(For Student’s use only)**

## In Lab Task:

1. Rohan wants to watch a horror movie. He is filling his details in the form given by ABC ticket booking app. Create a form which follows following condition (Use JavaScript validations)
   1. Create first name and last name columns. Once user leaves the column, input must change to uppercase. (Use onchange event)
   2. Create an age column using the type “text”. Validate the input and show an error message if age is less than 18.
   3. Create phone number field and make it mandatory field. Validate the input and show a message if any error.
   4. Create an email id field with the type “email” and make the field mandatory.
   5. Create a dropdown list to select number of tickets.
   6. Create a checkbox saying I accept to all terms and conditions and make it mandatory field.
   7. Finally create a submit button which gives an alert message “Do you want to submit? (Use Event Listener)

**Writing space for the Problem:(For Student’s use only)**

</head>

<body>

<h2>Welcome to ABC ticket booking </h2>

<html>

<head>

<script>

function validateForm() {

 var x = document.forms["myForm"]["fname"].value.toUpperCase();

 var y=document.forms["myForm"]["lname"].value.toUpperCase();

var z= document.getElementById("numb").value;

 var h=document.getElementById("number").value;

var length = Math.log(h) \* Math.LOG10E + 1 | 0;

var text1=document.getElementById("email").value;

var regx=/^([a-z 0-9\.-]+)@([a-z0-9-]+).([a-z]{2,8})(.[a-z]{2,8})?$/;

 var option = document.getElementById("myCheck");

if (x == "" || y== "")

{

alert("Name must be filled out");

 }

if(isNaN(z)||z<18) { text = "You are too younger to watch this movie";

 alert(text);

 }

if (isNaN(h) || length!=10) {

 text = "Phone number is not valid ";

 alert(text);

 if(regx.test(text1))

 {

 document.getElementById("lbltext").innerHTML="Valid";

 }

else {

 alert("enter valid email id ");

}

 }

 if(option.checked==false) {

 alert("Please check the terms and conditions");

 }

 else {

 alert("Your tickets have been successfully booked !!! Enjoy your movie");

}

}

 </script>

<form name="myForm" onsubmit="validateForm()" method="post">

<P>Enter your first name and last name</P>

First Name: <input type="text" name="fname" >

 Last Name : <input type="text" name="lname">

 <p>Note: age must be 18 or above</p>

 AGE : <input id="numb" name = "age">

<p>Enter your 10-digit mobile number</p>

 Number :<input id="number" name="phn nbr">

 <p>Enter your email address</p>

 Email :<input type="text" id="email" name="email">

<p>Select number of tickets u want to purchase</p>

 <select>

 <option value="1">1</option>

 <option value="2">2</option>

 <option value="3">3</option>

 <option value="4">4</option>

 <option value="5">5</option>

</select>

I Agree to all terms and conditions: <input type="checkbox" id="myCheck" name="myCheck">

<p>Do u want to submit</p>

 <input type="submit" value="Submit">

</form>

</body>

</html>

**Post Lab Task:**

1. Natasha is given with different set of values and she wants to validate which value is number and which is not. Help her solve this problem using JavaScript.

**Writing space for the Problem:(For Student’s use only)**

<!DOCTYPE html>

<html>

<head>

 <meta charset="utf-8">

<title>This JavaScript program is to validate whether a given value is number or not.</title>

</head>

<script>

 function is\_number(value)

{

 return !isNaN(value) && toString.call(value) === '[object Number]';

console.log(is\_number(NaN));

console.log(is\_number(42.32));

console.log(is\_number(72));

</script>

<body>

</body>

</html>

**Writing space for the Problem:(For Student’s use only)**

*(For Evaluator’s use only)*

Signature of the Evaluator Date of Evaluation:

Evaluator’s Observation

Marks Secured: out of

Full Name of the Evaluator:

Comment of the Evaluator (if Any)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# JDBC #4

**Date of the Session: / / Time of the Session: to**

**Prerequisite:**

* **How a java program is connected to database**
* **Basic SQL commands to perform ‘crud’ operations**

## Pre-Lab Task:

1.

* 1. List out different kinds of JDBC drivers.
* JDBC-ODBC Bridge Driver

Native Driver

Network Protocol Driver

Thin Driver

* 1. Write the steps involved in connecting any java application with data base using JDBC.

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

Register the Driver class

Create connection

Create statement

Execute queries

Close connection

* 1. Write a small description and syntax for the following methods:
     1. forName()

Description:-

This method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

Syntax:-public Static void for Name(String className) throws ClassNotFoundException

* + 1. registerDriver()

Description:-

This method of DriverManager class registers the given driver in the DriverManager's list. If the driver is null, it returns the NullPointerException

Syntax:-public static void registerDriver(Driver driver) throws SQLException

* + 1. getConnection()

Description:-

This method of DriverManager class is used to establish connection with the database.

Syntax:-public static Connection getConnection(String url) throwsSQLException

Public static Connection getConnection(String url,String name,String password) throws SQLException

* + 1. createStatement()

Description:-

This method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Syntax:-public Statement createStatement() throws  SQLException

* + 1. prepareStatement()

Description:-

This method of Connection interface is used to return the object of PreparedStatement.

Syntax:-public PreparedStatement prepareStatement(String query) throws  SQLException{}

* + 1. executeQuery()

Description:-

This method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Syntax:-public ResultSet executeQuery(String sql) throws SQLException

* + 1. executeUpdate()

Description:-

This method is used to execute SQL statements, for which you expect to get a number of rows affected.

Syntax:-int executeUpdate(String SQL)

* + 1. close()

Description:-

This method of Connection interface is used to close the connection.

Syntax:-public void close() throws SQLException

* + 1. setBinaryStream()

Description:-

This method of PreparedStatement is used to set Binary information into the  parameterIndex.

Syntax:-public void setBinaryStream(int paramIndex,InputStream stream)throws SQLException

Public void setBinaryStream( int paramIndex,InputStream stream,long length)throws SQLException

* + 1. setCharacterStream()

Description:-

This method of PreparedStatement is used to set character information into the  parameterIndex.

Syntax:-

void setCharacterStream (int parameterIndex, Reader reader ) throws SQLException

* + 1. available()

Description:-

This is used to return the estimated number of bytes that can be read from the input stream.

Syntax:-

Public int available()

1. Write a java program which executes the given SQL queries.

(NOTE: create a table ‘student’ with attributes to store the student’s id(number),name(varchar2),age(number) manually in database)

* Add the following records into the ‘student’ table.

|  |  |  |
| --- | --- | --- |
| **Id** | **name** | **age** |
| 1 | Felix | 20 |
| 2 | jack | 19 |
| 3 | mark | 21 |

* Print the details of the students whose age is less than 21.

**Writing space for the Problem:(For Student’s use only)**

Package com.klu;

Import java.sql.\*;

import java.util.\*;

public class Insert {

public static void main(String[] args) throws Exception {

Driver dre=new oracle.jdbc.driver.OracleDriver();DriverManager. registerDriver (dre);

Connection con =

DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

System.out .println("Enter the no of Records:");

Scanner sc = new Scanner(System.in);

Int n=sc.nextInt();

PreparedStatement  ps;

for(int i=0;i<n;i++) {

System.out .println("Enter the Id,Name,Age of the student:");

Int x=sc.nextInt();

String y=sc.next()+sc.nextLine();

int z=sc.nextInt();

 ps = con.prepareStatement("insert into studentvalues(?,?,?)");

 ps.setInt(1,x);

 ps.setString(2,y);

 ps.setInt(3,z);

 ps.executeUpdate();

}

}

}

**Writing space for the Problem:(For Student’s use only)**

1. A student joined in the class; teacher gave the first assignment to the student. The teacher wants the student to take the input from the user about their username and password, and check these against the registered user table in your database. Teacher gave a suggestion to solve the problem by using the PreparedStatement class in the JDBC API.What would be the answer for assignment.

**Writing space for the Problem:(For Student’s use only)**

Package com.klu;

import java.sql.\*;

import java.util.\*;

public class Check {

public static void main(String[] args) throwsException {

Driver dre=new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver (dre);

Connection con =

DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

System.out .println("Enter the Registered Details.");

Scanner sc = newScanner(System.in);

System.out .println("Username:");String uname = sc.next();

System.out .println("Password:");

String  pwd = sc.next();

//System.out.println(ps);

PreparedStatement stmt=con.prepareStatement("select \* from logindetails where

uname=? and pwd=?");

stmt.setString(1,uname);

stmt.setString(2, pwd);

if (stmt.executeUpdate()==1) {

System.out .println("You are in the list.");

}

else

 {

System.out .println("You are not in the list");

}

}

}

## In Lab Task:

1. Dany went to the kings landing supermarket and bought some groceries. As a vendor you have to ask the numbers of items she bought and store the itemid, itemname, cost of the items and store them in the ‘sales’ table of database and also execute an SQL query to find the total cost and list out the item she bought.

**Writing space for the Problem:(For Student’s use only)**

Package com.klu;

Import java.sql.\*;

Import java.util.\*;

Public class Market {

Public static void main(String args[] throws Exception {

Class. forName("oracle.jdbc.driver.OracleDriver");

Connection

con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manager"); System.out .println("Enter the No of Items Bought.");

Scanner sc=new Scanner(System.in);

Int n=sc.nextInt();

for(int i=0;i<n;i++) {

System.out .println("ID:");

Int id=sc.nextInt();

System.out .println("Item Name:");String name =sc.next();

System.out .println("Item cost:”);

float cost=sc.nextFloat();

PreparedStatement  ps = con.prepareStatement("insert into sales values(?,?,?)");

 ps.setInt(1,id);

 ps.setString(2,name);

 ps.setFloat(3,cost);

 ps.executeUpdate();

}

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select \* from sales");

while(rs.next()) {

System.out .println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getFloat(3)); }

System.out .println("TOTAL:");

ResultSet rs1=stmt.executeQuery("select sum(cost) from sales");

while(rs1.next()) {

System.out .println(rs1.getFloat(1));

}

Stmt.executeQuery(“truncate table sales”);

}

}

1. A course tutor James of a new class has the ID numbers of the students enrolled into his class. He needs to make a database to store student's name, date of birth and email address. He wants to embed all the id numbers into the database at first and afterward update it by the details given by the student himself on his first class. Help James by doing following things
   1. Create a new table namely Student in the Oracle Database with columns: Student\_ID, Student\_Name, Email, Date\_of\_Birth.
   2. Write a java program to embed id numbers into the database. The program ought to ask the user whether he/she wants to enter the ID number in loop. It should continue if the option is 'yes' and terminate if the option is 'no'. Every time in the loop it should take an ID number as an input from console and insert it into the database in a new row.( Note that the values of the columns other than the Student\_ID remain null).
   3. Write another java program which asks the ID number of the student. If the number is in the database, then the program should ask the Student's name, email address and the date of birth of that student and update the database with the received information at the respective row. The data collection should be repeated by asking every time, whether the user wants to continue entering the details of the next student or not.
   4. Write the java program to see the whole database in a formatted way( along with column headings).

**Writing space for the Problem:(For Student’s use only)**

a)Create table Student(Student -ID int primarykey, Student\_Name varchar(30), Email varchar(40), Date\_of\_Birth varchar(20));

b.

import java.sql.\*;

import java.util.\*;

class Test3{

 public static void main(String []args) throws Exception{

Driver d=new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver(d);Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","manger");

if(con!=null){

System.out.println("connected");

}

else{

System.out.println("not connected");

}

PreparedStatement stmt=con.prepareStatement("insert into student values(?,?,?)");

int sid;

String sname;

String email;

String date;

Scanner sc=new Scanner(System.in);

boolean flag=true;

while(flag){

System.out.println("entersid");

sid=sc.nextInt();

stmt.setInt(1,sid);

stmt.setString(2,null);

stmt.setString(3,null);

int i=stmt.executeUpdate();

System.out.println("Do U want to continue?? yes/no");

String a=sc.next();

if(a.equals("yes"))flag=true;elseflag=false;

}

con.close();

}

}

c.

import java.sql.\*;

import java.util.\*;

class Test4{ public static void main(String []args) throws Exception{

Driver d=new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver(d);

Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","manger");

if(con!=null){System.out.println("connected");

}

else{

System.out.println("not connected");

}

int sid;

String sname;

String mail;

String date;

Scanner sc=new Scanner(System.in);

 boolean flag=true;

while(flag) {

System.out.println("enter sid");

sid=sc.nextInt();

System.out.println("enter sname");

sname=sc.next();

System.out.println("enter email");

mail=sc.next();

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("update student set student\_name='"+sname+"', email='"+mail+"' where student\_id="+sid);

System.out.println("Do U want to continue?? yes/no");

String a=sc.next();

if(a.equals("yes"))

flag=true;

else

flag=false;

}

}

}

1. A workshop based on web development is being conducted in the university. Now James wants to know how many members of his class are interested in it and their details.
2. He needs to collect ID numbers and contact numbers of the interested students in a new table namely Workshop.
3. He wants to retrieve name and email address from the student table and update the Workshop table by creating 2 new columns and inserting the respective details.( note that the student table is the one created in the previous problem).
4. If a student, suddenly wants to drop from attending the workshop, James is supposed to delete the student's data from workshop table.

Write the Java programs for the above situations using JDBC API.

**Writing space for the Problem:(For Student’s use only)**

import java.sql.\*;

import java.util.\*;

class Workshop{

public static void main ( String args[] )throws Exception{

Driver dr = new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver(dr);

Connection c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe""manger");

Statement st = c.createStatement();

Scanner sc = new Scanner (System.in);

char op = 'y';

System.out.print("Do you want to insert your details?(Yes/No) : ");

String opt = sc.nextLine();

if(opt.equals("No") || opt.equals("no") || opt.equals("NO"))

op = 'n';

while(op == 'y'){

System.out.print("Enter the ID number : ");

int id = sc.nextInt();

System.out.print("Enter the contact number : ");

long pn = sc.nextLong();

try{

st.executeUpdate("Insert into workshop values("+id+","+pn+")");

}

catch(Exception e){

System.out.println(e);

System.out.println("Data not inserted");

}

System.out.print("Do you want to insert your details?(Yes/No) : ");

opt = sc.next();if(opt.equals("No") || opt.equals("no") || opt.equals("NO"))

op = 'n';

}

ResultSet rs = st.executeQuery("Select \* from workshop");

while(rs.next()){

System.out.println(rs.getInt(1)+"\t"+rs.getLong(2));

}

}

}

b)

import java.sql.\*;

import java.util.\*;

class Workshop\_b{

public static void main ( String args[] )throws Exception{

Driver dr = new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver(dr);

Connection c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manger");

Statement st1 = c.createStatement();

Statement st2 = c.createStatement();

try{

st1.executeUpdate("alter table Workshop add (name varchar2(30), email varchar2(30))");

}catch(Exception e){

System.out.println(e);

}

ResultSet rs = st1.executeQuery("select student\_id, student\_name, email from student");

while(rs.next()){

//System.out.println(rs.getLong(1) +"\t"+ rs.getString(2));

st2.executeUpdate("Update workshop set email='"+rs.getString(3)+"', name

='"+rs.getString(2)+"' where id="+rs.getLong(1));

}

}

}

c)

import java.sql.\*;

import java.util.\*;

class Workshop{

 public static void main ( String args[] )throws Exception{

Driver dr = new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver(dr);

Connection c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manger");

Statement st = c.createStatement();

Scanner sc = new Scanner (System.in);

char op = 'y';

System.out.print("Do you want to delete your details?(Yes/No) : ");

String opt = sc.nextLine();

if(opt.equals("No") || opt.equals("no") || opt.equals("NO"))op = 'n';

while(op == 'y'){

System.out.print("Enter the ID number : ");

int id = sc.nextInt();

try{

st.executeUpdate("delete from workshop where id = "+id);

}

catch(Exception e){

System.out.println(e);

}

System.out.print("Do you want to delete your details?(Yes/No) : ");

opt = sc.next();

if(opt.equals("No") || opt.equals("no") || opt.equals("NO"))

op = 'n';

}

ResultSet rs = st.executeQuery("Select \* from workshop");

while(rs.next()){

System.out.println(rs.getInt(1)+"\t"+rs.getLong(2));

}

}

}

## Post Lab Task

1. Tony is a wildlife photographer. He went to Amazon rainforest to explore the beauty of the flora and fauna. He captured pictures of different trees and animals. He also described each animal and plant in a separate notepad. Now he wants to save the name, the category (animal

/ plant), its image and the description file of the species into a database. Help Tony by providing him with an executable java program.

* 1. Create a table with columns Name, Category, Image and Description\_File before executing the file.
  2. The name, category, path of the image(.jpg), path of the description file (.txt) should be read as input from the console.

**Writing space for the Problem:(For Student’s use only)**

import java.sql.\*;

import java.io.\*;

import java.util.\*;

public class Amazon{ public static void main(String[]args)throws Exception {

Scanner sc= new Scanner(System.in);

Driver d= new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver(d);Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manger");

int cont=1;

while(cont==1){

PreparedStatement st =con.prepareStatement("insert into forest values(?,?,?,?)");

System.out.println("Enter the name of the plant/animal ?");

String name =sc.next();

System.out.println("Enter the category ?");

String category=sc.next();

System.out.println("Enter the path of the image");

String ipath=sc.next();

System.out.println("Enter the path of the text file");

String fpath=sc.next();

st.setString(1,name);

st.setString(2,category);

FileInputStream f =new FileInputStream(ipath);

st.setBinaryStream(3,f,f.available());

File fs = new File(fpath);

FileReader fr=new FileReader(fs);

st.setCharacterStream(4,fr,(int)fs.length());

st.executeUpdate();

System.out.println("Do you want to Continue(yes/no) :");

String flag = sc.next();

if(flag.equals("yes")){cont=1;

}

else{

cont=0;

}

}

Con.close();

}

}

1. After exploring the dense forest, Tony returns to home and wants to publish his work in a magazine. He is supposed to give the information to the publisher in the following way.
   1. Two folders namely Animals and Plants should be created on desktop
   2. The image of every animal should be saved as "animal\_name.jpg" and the description file as "animal\_name.txt" in the Animal folder
   3. Similarly, the image of every plant should be saved as "plant\_name.jpg" and the description file as "plant\_name.txt" in the Plant folder

**Writing space for the Problem:(For Student’s use only)**

import java.sql.\*;

import java.io.\*;

import java.util.\*;

 public class Amazon2{ public static void main(String[]args)throws Exception {

Scanner sc= new Scanner(System.in);

Driver d= new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver(d);

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manger");

PreparedStatement st= con.prepareStatement("select \* from forest");

ResultSet rs=st.executeQuery();

while(rs.next()){Blob b= rs.getBlob(3);

byte a[] = b.getBytes(1,(int)b.length());

String check=rs.getString(2);

if(check.equals("animal")){

FileOutputStream fout=new

FileOutputStream("C:\\Users\\klu\\Desktop\\animal\\"+rs.getString(1)+".jpg");

fout.write(a);

fout.close();

}

else{

FileOutputStream fout = new FileOutputStream("C:\\Users\\klu\\Desktop\\plant\\"+rs.getString(1)+".jpg");

fout.write(a);fout.close();}Clob c= rs.getClob(4);

Reader r =c.getCharacterStream();

if(check.equals("animal")){

FileWriter fw = new

FileWriter("C:\\Users\\klu\\Desktop\\animal\\"+rs.getString(1)+".txt");

int i;

while((i=r.read())!=-1){

fw.write((char)i);

}

fw.close();

}

else{

FileWriter fw = new

FileWriter("C:\\Users\\klu\\Desktop\\plant\\"+rs.getString(1)+".txt");

int i;

while((i=r.read())!=-1){

fw.write((char)i);

}

fw.close();

}

}

con.close();

System.out.println("Success");

}

}

*(For Evaluator’s use only)*

Signature of the Evaluator Date of Evaluation:

Evaluator’s Observation

Marks Secured: out of

Full Name of the Evaluator:

Comment of the Evaluator (if Any)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# Servlets - 1 #5

**Date of the Session: / / Time of the Session: to**

**Prerequisite:**

* Servlets and their applications.
* ‘CRUD’ operations in SQL.

## Pre-Lab Task:

1. Write a note on servlet and its life cycle.

The web container maintains the life cycle of a servlet instance

Servelet class is loaded

Servelet instance is created

Init method is invoked

Service method is invoked

Destroy method is invoked

1. Draw the Structure of of directories that are to be created under Tomcat’s ‘webapps’ folder.

A screenshot of a computer

Description automatically generated

1. Create a project named ‘myservice’ and using servlets make a service which prints “Welcome to Servlets!!” on to the server console. The class must be named ‘FirstServlet’ and the URL pattern must be ‘/your\_name’.

Add another service into the same ‘myservice’ project which prints “Bye to Servlets!!” on to the server console. Name the class ‘SecondServlet’ and set URL pattern to ‘/your\_name143’. (Hint: Modify the ‘web.xml’ file by adding another servlet name and URL pattern.)

Check the working condition of both the services.

**Writing space for the Problem:(For Student’s use only)**

FirstServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class FirstServlet extends HttpServlet { private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

System.out.println("Welcome to Servlet!!");

}

protected void doPost(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

}

}

SecondServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpServletResponse;

public class SecondServlet extends HttpServlet { private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

System.out.println("Bye to Servlets!!");

}

protected void doPost(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

}

}

web.xml

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

<web-app xmlns:xsi=<http://www.w3.org/2001/XMLSchema-instance>

xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/webapp\_2\_5.xsd"

xsi:schemaLocation="http://java.sun.com/xml/ns/jvaee http://java.sun.com/xml/ns/javaee/web-

app\_2\_5.xsd"

id="WebApp\_ID" version="2.5">

<servlet>

<servlet-name>First</servlet-name>

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

</servlet>

<servlet-mapping>

<servlet-name>First</servlet-name>

<url-pattern>/akshay</url-pattern>

</servlet-mapping>

<servlet>

<servlet-mapping>

<servlet-name>Second</servlet-name>

<url-pattern>/akshay143</url-pattern>

</servlet-mapping>

</web-app>

**Writing space for the Problem:(For Student’s use only)**

## In Lab Task:

1. A professor from K L University after 21 years of Work he decided to retire from his job and before he leaves, he wants to get feedback about him from all his students. As you are his favorite student, he asked you for help, so, you decided to make a Web app using servlets in which the students must first register and then login into it with the registered credentials, which in turn gives a page and Asks for remarks , the students can type and submit their views in it and this login credentials and remarks must be stored into ‘farewell’ table in database. He wants everyone to register and login because he wants to know how every particular student think about him.

**Writing space for the Problem:(For Student’s use only)**

Index.html:

<!DOCTYPE html>

<html>

<head>

<meta charset= "ISO-8859-1" >

title>Insert title here</title>

</head>

< body>

<form action= "Registration" method="post" >

id<input type= "text" name="id" />

name<input type= "text" name="name" />

email<input type= "text" name="email" />

username<input type= "text" name="uname" />

password<input type= "text" name="pwd" />

<input type= "submit" name="submit" />

</form>

</ body>

</html>

Registration.java:

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class Registration

\*/

public class Registration extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public Registration() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath());

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stub

//doGet(request, response);

PrintWriter out=response.getWriter();

response.setContentType("text/html");

String name=request.getParameters(“name”);

String id=request.getParameter("id");

String email=request.getParameter("email");

String uname=request.getParameter("uname");

String pwd=request.getParameter("pwd");

Connection con;

try {

Class.forName("oracle.jdbc.driver.OracleDriver");

con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System","Manager");

PreparedStatement ps=con.prepareStatement("insert into farewell values(?,?,?,?,?)");

ps.setString(1,id);

ps.setString(2, name);

ps.setString(3, email);

ps.setString(4,uname);

ps.setString(5, pwd);

int rs=ps.executeUpdate();con.commit();

if(rs!=0) {out.println("succesfull");

out.println("<a href='index1.html'>Login</a>");

}

else out.println("try again");

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

}

}

Index1.html:

<!DOCTYPE html>

<html>

<head>

<meta charset= "ISO-8859-1" >

<title>Insert title here</title>

</head>

< body>

<form method= "post"  action="Login" >

username<input type= "text" name="uname" />

password<input type= "password" name="pwd" />

<input type= "submit"  name=“login" />

</form>

</ body>

</html>

Login.java:

import java.io.IOException;

import java.io.PrintWriter;

importjava.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;/\*\* \* Servlet implementation class Login \*/

public class Login extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

Public Login() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

Protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stub//response.getWriter().append("Servedat:").append(request.getContextPath());

PrintWriter out=response.getWriter()

response.setContentType("text/html");

String uname=request.getParameter("uname");

@SuppressWarnings("unused") String  pwd=request.getParameter("pwd");

Connection con;

try

{

Class. forName("oracle.jdbc.driver.OracleDriver");

con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System","Manager"); PreparedStatement  ps=con.prepareStatement("select uname,pwd from farewell

where uname=? and pwd=?");

ps.setString(1, uname);

ps.setString(2,  pwd);

ResultSet rs= ps.executeQuery();

if (rs.next()) {

HttpSession session=request.getSession();

session.setAttribute("uname", uname);

session.setAttribute("pwd", pwd);

out.println("<a href='index2.html'>Enter your remarks</a>");

}

}

catch

(Exception e) {out.println(e);}

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

Protected void doPost(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stubdoGet(request, response);

}

}

Index2.html:

<!DOCTYPE html>

<html>

<head>

<meta charset= "ISO-8859-1" >

<title>Insert title here</title>

</head>

< body>

<form method= "post" action="Login2" >

< input type=”text” name=”remarks”/>

<input type= "submit" name="submit" />

</form>

</ body>

</html>

Login2.java:

Import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\*

\* Servlet implementation class Login2

\*/

Public class Login2 extends HttpServlet {

Private static final long serialVersionUID= 1L;

/\*\*

\* @see  HttpServlet#HttpServlet()

\*/

public Login2() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

Protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stub

//response.getWriter().append("Served at: ").append(request.getContextPath());

PrintWriter out=response.getWriter();

response.setContentType("text/html");

HttpSession session=request.getSession();

String remark =request.getParameter("remark");

String uname=(String)session.getAttribute("uname");

String  pwd=(String)session.getAttribute("pwd");

Connection con;

try

{

Class. forName("oracle.jdbc.driver.OracleDriver");

con=DriverManager. getConnection("jdbc:oracle:thin:@localhost:1521:xe","System","Manager");con.setAutoCommit(false);

PreparedStatement  ps=con.prepareStatement("insert into farewell1 values(?,?,?)");

ps.setString(1, uname);

ps.setString(2,  pwd);

ps.setString(3, remark );

int p= ps.executeUpdate();

con.commit();

if

( p!=0) {

out.println("thank you");

}

else{out.println("try again");}

}

catch (Exception e) {out.println(e);}

}/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

Protected void doPost(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}

Web.xml:

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

<web-app xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee"  xsi:schemaLocation=

"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/webapp\_2\_5.xsd"  id="WebApp\_ID"  version="2.5"

> <display-name>LogReg</display-nme>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

<servlet>

<description></description>

<display-name>Registration</display-name>

<servlet-name>Registration</servlet-name>

<servlet-class>Registration</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Registration</servlet-name>

<url-pattern>/Registration</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>Login</display-name>

<servlet-name>Login</servlet-name>

<servlet-class>Login</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Login</servlet-name>

<url-pattern>/Login</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>Login2</display-name>

<servlet-name>Login2</servlet-name>

<servlet-class>Login2</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Login2</servlet-name>

<url-pattern>/Login2</url-pattern>

</servlet-mapping>

</web-app>

**Writing space for the Problem:(For Student’s use only)**

## Post Lab Task

* 1. Create a login Servlet which on login retrieves and Displays the Previous Remark Given by you. You must also provide an Update Remark Text Field in which the student can type a new remark and replace the previous remark with it. For login, use the credentials that are in the ‘farewell’ table. **Use Eclipse IDE for ease of the task.**

**Writing space for the Problem:(For Student’s use only)**

Index.html:

<!DOCTYPE html>

<html>

<head>

<meta charset= "ISO-8859-1" >

<title>Insert title here</title>

</head>

< body>

<form method= "post" action="LoginCheck" >

username<input type= "text" name="uname" />

password<input type= "password" name="pwd" />

<input type= "submit" name="login" />

</form>

</ body>

</html>

LoginCheck.java:

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\*

\* Servlet implementation class LoginCheck

\*/ public class LoginCheck extends HttpServlet { private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public LoginCheck() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stub//doGet(request, response);

PrintWriter out=response.getWriter();

response.setContentType("text/html");

HttpSession session=request.get Session();

String uname=request.getParameter("uname");

String pwd=request.getParameter("pwd");

session.setAttribute("uname", uname);

session.setAttribute("pwd",pwd);

Connection con;try {

Class.forName("oracle.jdbc.driver.OracleDriver");

con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System","Manager";

PreparedStatement ps=con.prepareStatement("select uname,pwd,remarks from

farewell1 where uname=? and pwd=?");

ps.setString(1, uname);

ps.setString(2, pwd);

ResultSet rs=ps.executeQuery();

if(rs.next()) {

//PreparedStatement ps=con.prepareStatement("select remark from

farewell1 where uname=? and pwd=?\");

out.println("Your remarks are:");

out.println(rs.getString(3));

out.println("<br/><a href='index1.html'>click here to change your

remarks</a>”);

}catch(Exception 2) {out.println€;}

}

}

Index1.html:

<!DOCTYPE html>

<html>

<head>

<meta charset= "ISO-8859-1" >

<title>Insert title here</title>

</head>

< body>

<form method= "post" action="ChangeRemark" >

<input type= "text" name="remark" />

<input type= "submit"  name="submit" />

</form>

</ body>

</html>

ChangeRemark.java:

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;import java.sql.Statement;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\*

\* Servlet implementation class ChangeRemark

\*/ public class ChangeRemark extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/ public ChangeRemark() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doPost(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stub

//doGet(request, response);

@SuppressWarnings("unused")String remark=request.getParameter("remark");

PrintWriter out=response.getWriter();

HttpSession session=request.getSession();

String uname=(String)session.getAttribute("uname");

String pwd=(String)session.getAttribute("pwd");response.setContentType("text/html");

Connection con;

try {

Class.forName("oracle.jdbc.driver.OracleDriver");

con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System","Manager";

con.setAutoCommit(false);

//PreparedStatement ps=con.prepareStatement("delete uname,pwd,remarks from

farewell1 where uname=?,pwd=?");

//ps.setString(1, uname);

//ps.setString(2, pwd);

//ps.executeUpdate();

//con.commit();

Statement ps1=con.createStatement();

String x="update farewell1 set remarks='remark' where uname=uname and  pwd=pwd";

int p1=ps1.executeUpdate(x);

//ps1.setString(1, remark);

//ps1.setString(1, uname);

//ps1.setString(2, pwd);

//int p1=ps1.executeUpdate();

con.commit();

if(p1!=0) {out.println("Remark changed");

}

else {

out.println("try again");

}

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

}

}

}

*(For Evaluator’s use only)*

Signature of the Evaluator Date of Evaluation:

Evaluator’s Observation

Marks Secured: out of

Full Name of the Evaluator:

Comment of the Evaluator (if Any)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# Servlets - 2 #6

**Date of the Session: / / Time of the Session: to**

**Prerequisite:**

* **Concepts of Init and context Parameters.**
* **Request Dispatcher functionality.**
* **Send Redirect Method usage.**

## Pre-Lab Task:

* + 1. Sam is getting registered in a degree college. He has to take up online registration process for which he needs url of college website, username, password. Like sam there are many students who need to register online. For every person there is unique username and password. For letting them know their credentials, create servlets for say 3 persons. Set url as context parameter and unique username and password as respective init parameters. Print url, username and password of each servlet. Try accessing username and password of one student from another student’s servlet.

**Writing space for the Problem:(For Student’s use only)**

Sone.java

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletConfig;

import javax.servlet.ServletContext;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class SOne

\*/

public class SOne extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/ public SOne() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stub//response.getWriter().append("Served at: ").append(request.getContextPath());

PrintWriter out=response.getWriter();response.setContentType("text/html");

out.println("SOne called...</br>");

ServletConfig init=getServletConfig();

String x=init.getInitParameter("var1");

out.println("init parameter of SOne called "+x);

ServletContext con=getServletContext();

String y=con.getInitParameter("var");

out.println("init parameter of SOne called "+y);

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doPost(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stubdoGet(request, response);

}

}

STwo.java

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletConfig;

import javax.servlet.ServletContext;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class STwo

\*/ public class STwo extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/ public STwo() { super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stub//response.getWriter().append("Served at: ").append(request.getContextPath());

PrintWriter out=response.getWriter();

response.setContentType("text/html");

ServletConfig init=getServletConfig();

String x=init.getInitParameter("var1");

out.println("init parameter of SOne called in STwo whose value is "+x);

ServletContext con=getServletContext();

String y=con.getInitParameter("var");

out.println("Contect parameter value in STwo "+y);

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doPost(HttpServletRequest request, HttpServletResponse response) throwsServletException, IOException {

// TODO Auto-generated method stubdoGet(request, response);

}

}

Web.xml

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

<web-app xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"  id="WebApp\_ID"  version="2.5" >

<display-name>initContext</display-name>

<context-param>

< param-name>var </ param-name>

< param-value>333</ param-value>

</context-param>

<welcome-file-list>

<welcome-file>index.html</welcome-fiele>

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

<servlet>

<description></description>

<display-name>SOne</display-name>

<servlet-name>SOne</servlet-name>

<servlet-class>SOne</servlet-class>

<init-param>

< param-name>var1</ param-name>

< param-value>33</ param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>SOne</servlet-name>

<url-pattern>/SOne</url-pattern>

</servlet-mapping>

<servlet>

<description></description>

<display-name>STwo</display-name>

<servlet-name>STwo</servlet-name>

<servlet-class>STwo</servlet-class>

<init-param>

< param-name>var2</ param-name>

< param-value>34</ param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>STwo</servlet-name>

<url-pattern>/STwo</url-pattern>

</servlet-mapping>

</web-app>

* + 1. Use forward() of request dispatcher interface to forward date from 1st servlet to 2nd servlet which contains system time. Print them on the webpage.

**Writing space for the Problem:(For Student’s use only)**

Destination.java:-

package arjun;

import java.io.IOException;

import java.util.Calendar;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class Destination

\*/

@WebServlet("/Destination")

public class Destination extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/ public Destination() { super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stubresponse.getWriter().println("Destination Servlet called....</br>");

response.getWriter().println(request.getAttribute("day"));

response.getWriter().println("</br>Time is "+Calendar.getInstance().getTime());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doPost(HttpServletRequest reguest,HttpServletResponse response) throws

ServletException, IOException {

// TODO Auto-generated method stubresponse.getWriter().println("Destination Servletcalled....</br>");

response.getWriter().println(request.getAttribute("day"));

response.getWriter().println("</br>Time is "+Calendar.getInstance().getTime());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doPost(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {// TODO Auto-generated method stubdoGet(request, response);}}

Source.java:-

package arjun;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class Source

\*/

@WebServlet("/Source")

public class Source extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/ public Source() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\*

@see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

PrintWriter printWriter=response.getWriter();

printWriter.println("Source servlet called..</br>");

request.setAttribute("day", "Saturday");

RequestDispatcher rd=request.getRequestDispatcher("/Destination");

//rd.forward(request, response);

rd.include(request, response);

}

/\*\*

\*

@see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throwsServletException, IOException {

// TOO Auto-generated method stubdoGet(request, response);

}

{ public void service(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException{String n=(String)req.getAttribute("name");int s=(int)req.getAttribute("x");PrintWriter out=res.getWriter();out.println("Student"+n+" secured "+s+" marks");}

Writing space for the Problem:(For Student’s use only)

In Lab Task:

Online tests have become common these days. To evaluate these online tests using servlets, build an evaluation servlet. Create servlets for each student whose init parameters are answers to MCQ questions. Correct answers are set as context parameters. In each servlet compare answers with correct marks and calculate score. This score must be forwarded to another servlet which prints student marks.

Writing space for the Problem:(For Student’s use only)

Web.xml:

<web-app>

<context-param>

<param-name>Q1</param-name>

<param-value>A</param-value>

</context-param><context-param>

<param-name>Q2</param-name>

<param-value>D</param-value>

</context-param>

<servlet>

<servlet-name>first</servlet-name>

<servlet-class>First</servlet-class>

<init-param>

<param-name>Q1</param-name>

<param-value>B</param-value>

</init-param>

<init-param>

<param-name>Q2</param-name>

<param-value>D</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>first</servlet-name>

<url-pattern>/eval1</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>second</servlet-name>

<servlet-class>Second</servlet-class>

<init-param>

<param-name>Q1</param-name>

<param-value>A</param-value>

</init-param>

<init-param>

<param-name>Q2</param-name>

<param-value>D</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>second</servlet-name>

<url-pattern>/eval2</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>third</servlet-name>

<servlet-class>Third</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>third</servlet-name>

<url-pattern>/Third</url-pattern>

</servlet-mapping>

</web-app>

First.java:-

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class First extends HttpServlet{

public void service(HttpServletRequest req,HttpServletResponse res)throwsServletException,IOException{

ServletContext app=getServletContext();

String ans1=app.getInitParameter("Q1");

String ans2=app.getInitParameter("Q2");

ServletConfig conf=getServletConfig();

String res1=conf.getInitParameter("Q1");

String res2=conf.getInitParameter("Q2");int score=0;if(res1.equals(ans1))score+=1;

if(res2.equals(ans2))score+=1;req.setAttribute("name","1");

//RequestDispatcher rd=req.getRequestDispatcher("/Third");

//rd.forward(req,res);

req.setAttribute("x",score);

RequestDispatcher rd=req.getRequestDispatcher("/Third");

rd.forward(req,res);

}

}

Second.java:

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

 public class Second extends HttpServlet{

public void service(HttpServletRequest req,HttpServletResponse res)throws

ServletException,IOException{ServletContext app=getServletContext();

String ans1=app.getInitParameter("Q1");String ans2=app.getInitParameter("Q2");

ServletConfig conf=getServletConfig();String res1=conf.getInitParameter("Q1");

String res2=conf.getInitParameter("Q2");

Intscore=0;

if(res1.equals(ans1))score+=1;

if(res2.equals(ans2))score+=1;

req.setAttribute("name","2");

//rd.forward(req,res);

req.setAttribute("x",score);

RequestDispatcher rd=req.getRequestDispatcher("/Third");

rd.forward(req,res);

}

}

Third.java:

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class Third extends HttpServlet

{

public void service(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException{

String n=(String)req.getAttribute("name");

int s=(int)req.getAttribute("x");

PrintWriter out=res.getWriter();

out.println("Student"+n+" secured "+s+" marks");

}

**Writing space for the Problem:(For Student’s use only)**

## In Lab Task:

* + - 1. Online tests have become common these days. To evaluate these online tests using servlets, build an evaluation servlet. Create servlets for each student whose init parameters are answers to MCQ questions. Correct answers are set as context parameters. In each servlet compare answers with correct marks and calculate score. This score must be forwarded to another servlet which prints student marks.

**Writing space for the Problem:(For Student’s use only)**

Web.xml:

<web-app>

<context-param>

<param-name>Q1</param-name>

<param-value>A</param-value>

</context-param><context-param>

<param-name>Q2</param-name>

<param-value>D</param-value>

</context-param>

<servlet>

<servlet-name>first</servlet-name>

<servlet-class>First</servlet-class>

<init-param>

<param-name>Q1</param-name>

<param-value>B</param-value>

</init-param>

<init-param>

<param-name>Q2</param-name>

<param-value>D</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>first</servlet-name>

<url-pattern>/eval1</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>second</servlet-name>

<servlet-class>Second</servlet-class>

<init-param>

<param-name>Q1</param-name>

<param-value>A</param-value>

</init-param>

<init-param>

<param-name>Q2</param-name>

<param-value>D</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>second</servlet-name>

<url-pattern>/eval2</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>third</servlet-name>

<servlet-class>Third</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>third</servlet-name>

<url-pattern>/Third</url-pattern>

</servlet-mapping>

</web-app>

First.java:-

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class First extends HttpServlet{

public void service(HttpServletRequest req,HttpServletResponse res)throwsServletException,IOException{

ServletContext app=getServletContext();

String ans1=app.getInitParameter("Q1");

String ans2=app.getInitParameter("Q2");

ServletConfig conf=getServletConfig();

String res1=conf.getInitParameter("Q1");

String res2=conf.getInitParameter("Q2");int score=0;if(res1.equals(ans1))score+=1;

if(res2.equals(ans2))score+=1;req.setAttribute("name","1");

//RequestDispatcher rd=req.getRequestDispatcher("/Third");

//rd.forward(req,res);

req.setAttribute("x",score);

RequestDispatcher rd=req.getRequestDispatcher("/Third");

rd.forward(req,res);

}

}

Second.java:

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

 public class Second extends HttpServlet{

public void service(HttpServletRequest req,HttpServletResponse res)throws

ServletException,IOException{ServletContext app=getServletContext();

String ans1=app.getInitParameter("Q1");

String ans2=app.getInitParameter("Q2");

ServletConfig conf=getServletConfig();

String res1=conf.getInitParameter("Q1");

String res2=conf.getInitParameter("Q2");

Intscore=0;

if(res1.equals(ans1))score+=1;

if(res2.equals(ans2))score+=1;

req.setAttribute("name","2");

//rd.forward(req,res);

req.setAttribute("x",score);

RequestDispatcher rd=req.getRequestDispatcher("/Third");

rd.forward(req,res);

}

}

Third.java:

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class Third extends HttpServlet

{

public void service(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException{

String n=(String)req.getAttribute("name");

int s=(int)req.getAttribute("x");

PrintWriter out=res.getWriter();

out.println("Student"+n+" secured "+s+" marks");

}

* + - 1. Ben made a web app which reads a password and checks the strength of it, if all the characters are of lowercase and no numbers in it then it must forward to a servlet which displays a message "The password must contain at least one uppercase and number in it.". If all the characters are uppercase and no number then it must forward to a servlet which displays "The password must contain at least a lowercase and a number in it.". If all the characters are numbers it must include to a servlet which displays "The password is very weak!!". The main servelt must pirnt the requirements of the password as"The password must contain an upper case,lower case and a numbet in it!!!"

**Writing space for the Problem:(For Student’s use only)**

index.html:-

<!DOCTYPE html>

<html>

<head>

<title>Login Form</title>

</head>

<body>

<div style="position:absolute;top:250px;left:500px;">

<form action="PasswordCheck" method="post">

username:<input type="text" name="uname"/>

<br/>

 password:<input type="password" name="pwd"/>

<br/>

<input type="submit" value="login"/>

</form>

</div>

</body>

</html>

Check.java:

-import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

 \* Servlet implementation class Check

  \*/

@WebServlet("/Check") public class Check extends HttpServlet {

private static final long serialVersionUID = 1L;

 /\*\*

 \* @see HttpServlet#HttpServlet()

 \*/

public Check() {

 super();

 // TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

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

// TODO Auto-generated method stub

//response.getWriter().append("Served at: ").append(request.getContextPath());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub//doGet(request, response);

String x=(String)request.getAttribute("pwd");if(x.equals(x.toLowerCase()))

{

System.out.println("All are lower case. Atleast one uppercase must be present");

/\*

System.out.println("<!DOCTYPE html>\r\n" +

"<html>\r\n" +

" <head>\r\n" +

" <title>Login Form</title>\r\n" +

" </head>\r\n" +

" <body>\r\n" +

" <div style=\"position:absolute;top:250px;left:500px;\">\r\n" +

" <form action=\"PasswordCheck\" method=\"post\">\r\n" +

" username:<input type=\"text\" name=\"uname\"/><br/>\r\n" +

" password:<input type=\"password\" name=\"pwd\"/><br/>\r\n" +

" <input type=\"submit\" value=\"login\"/>\r\n" +

" </form>\r\n" +

" </div>\r\n" +

" </body>\r\n" +

"</html>");\*/}}}

Chech1.java:-

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

 \* Servlet implementation class Chech1

 \*/@WebServlet("/Chech1")

public class Chech1 extends HttpServlet {

 private static final long serialVersionUID = 1L;

/\*\*

 \* @see HttpServlet#HttpServlet()

\*/

 public Chech1() {

 super();

// TODO Auto-generated constructor stub

 }

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub//doGet(request, response);

String x=(String)request.getAttribute("pwd");

if(x.equals(x.toUpperCase()))

{

System.out.println("All are upper case. Atleast three lowercase must be present");

/\*

System.out.println("<!DOCTYPE html>\r\n" +

"<html>\r\n" +

" <head>\r\n" +

" <title>Login Form</title>\r\n" +

" </head>\r\n" +

" <body>\r\n" +

" <div style=\"position:absolute;

top:250px;left:500px;\">\r\n" +

" <form action=\"PasswordCheck\" method=\"post\">\r\n" +

" username:<input type=\"text\" name=\"uname\"/><br/>\r\n" +

" password:<input type=\"password\" name=\"pwd\"/><br/>\r\n" +

" <input type=\"submit\" value=\"login\"/>\r\n" +

" </form>\r\n" +

" </div>\r\n" +

" </body>\r\n" +

"</html>");

\*/

}

}

}

PasswordCheck.java:-

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.ServletResponse;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

 \* Servlet implementation class PasswordCheck

  \*/

@WebServlet("/PasswordCheck")

 public class PasswordCheck extends HttpServlet {

 private static final long serialVersionUID = 1L;

 /\*\*

 \* @see HttpServlet#HttpServlet()

 \*/ public PasswordCheck() {

super();

 // TODO Auto-generated constructor stub

 }

/\*\*

\*

@see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

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

// TODO Auto-generated method stub

//response.getWriter().append("Served at: ").append(request.getContextPath());

}

/\*\*

\*

@see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/ protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

PrintWriter out=response.getWriter();

((ServletResponse) out).setContentType("text/html");

String x=request.getParameter("pwd");

out.print("password check "+x);

request.setAttribute("pwd","x");

RequestDispatcher rd=request.getRequestDispatcher("/Check");

rd.forward(request,response);

}

}

**Writing space for the Problem:(For Student’s use only)**

## Post Lab Task

1. Use sendRedirect() to open google and search keywords stored in database. Establish jdbc connection by setting url of database as context parameter and username, password as init parameters.

**Writing space for the Problem:(For Student’s use only)**

Index.html:

<!DOCTYPE html>

<html>

<head>

<title>sendRedirect example</title>

</head>

<body>

<form action="hello">

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

<input type="submit" value="Google Search">

</form>

</body>

</html>

Web.xml:

<web-app>

<servlet>

<servlet-name>fs</servlet-name>

<servlet-class>MySearcher</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>fs</servlet-name>

<url-pattern>/hello</url-pattern>

</servlet-mapping>

</web-app>

Mysearcher.java:

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.sql.\*;

 public class MySearcher extends HttpServlet {

PrintWriter out; public void service(HttpServletRequest request, HttpServletResponse response) throws ServletException,IOException{

out=response.getWriter();

try{

Class.forName("oracle.jdbc.driver.OracleDriver");



Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

int number=Integer.parseInt(request.getParameter("name"));

String sql="select word from search where sno=?";

PreparedStatement stmt=con.prepareStatement(sql);

stmt.setInt(1,number);

ResultSet rs=stmt.executeQuery();

if(rs.next())

{

Stringname=rs.getString(1);

response.sendRedirect("https://www.google.co.in/#q="+name);

}

else{

out.println("Wrong choise");

}

}catch(Exception e)

{

out.println("Error...."+e);

}

}

}

*(For Evaluator’s use only)*

Signature of the Evaluator Date of Evaluation:

Evaluator’s Observation

Marks Secured: out of

Full Name of the Evaluator:

Comment of the Evaluator (if Any)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# JSP(Java Server Pages)-1 #7

**Date of the Session: / / Time of the Session: to**

**Prerequisite:**

* **Knowledge of JSP.**
* **Basic SQL commands to perform ‘CRUDs’ operations**

## Pre-Lab Task:

1. Clarify in detail each phase of JSP lifecycle.

**Writing space for the Problem:(For Student’s use only)**

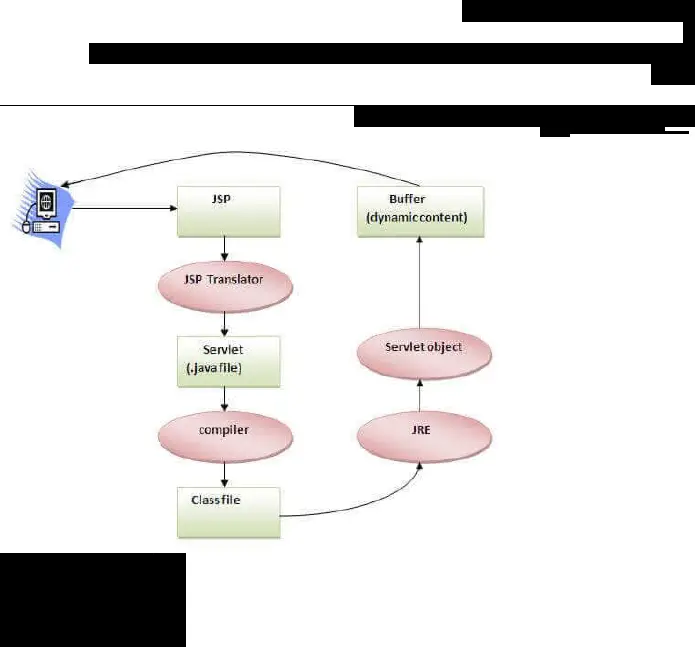
The LifeCycle of a JSP:-

JSP page is translated into Servlet by the help of JSP translator.

The JSP translator is a part of the web server which is responsible for translating the JSP page into Servlet.

After that, Servlet page is compiled by the compiler and gets converted into the class file.

Moreover, all the processes that happen in Servlet are performed on JSP later like initialization, committing response to the browser and destroy.



1. Print Current date and time on the screen utilizing JSP.

Date.jsp

<%@ page import="java.util.Date"  %>

<%@ page language="java" contentType="text/html;" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" >

<title>JSP DATE</title>

</head>

<body>

<b>Current Date and Time</b>: <%=newDate() %>

</body>

</html>

Web.xml

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

<web-app >

<display-name>Date</display-name>

 <welcome-file-list>

 <welcome-file>date.jsp</welcome-file>

</welcome-file-list>

</web-app>

## In Lab Task:

1. John needs to perform CRUD tasks on database utilizing JSP. Help him perform creation, retrieval, update and cancellation tasks on database utilizing JSP.

Creation :

<%@ page import="java.sql.\*" %>

<%

Connection conn = null;

PreparedStatement stmt = null;

try {

// Get a connection to the database

conn = DriverManager.getConnection("jdbc:mysql://localhost/mydatabase", "username", "password");

// Create a statement

stmt = conn.prepareStatement("INSERT INTO mytable (name, email) VALUES (?, ?)");

// Set the parameters

stmt.setString(1, "John Doe");

stmt.setString(2, "johndoe@example.com");

// Execute the statement

stmt.executeUpdate();

} catch (SQLException e) {

// Handle the exception

} finally {

// Close the connection

if (conn != null) {

try {

conn.close();

} catch (SQLException e) {

// Handle the exception

}

}

}

%>

Retrieval :

<%@ page import="java.sql.\*" %>

<%

Connection conn = null;

PreparedStatement stmt = null;

ResultSet rs = null;

try {

// Get a connection to the database

conn = DriverManager.getConnection("jdbc:mysql://localhost/mydatabase", "username", "password");

// Create a statement

stmt = conn.prepareStatement("SELECT \* FROM mytable WHERE name = ?");

// Set the parameters

stmt.setString(1, "John Doe");

// Execute the statement

rs = stmt.executeQuery();

// Process the results

while (rs.next()) {

// Print the name and email

System.out.println(rs.getString("name") + " - " + rs.getString("email"));

}

} catch (SQLException e) {

// Handle the exception

} finally {

// Close the connection

if (conn != null) {

try {

conn.close();

} catch (SQLException e) {

// Handle the exception

}

}

}

%>

Update :

<%@ page import="java.sql.\*" %>

<%

Connection conn = null;

PreparedStatement stmt = null;

try {

// Get a connection to the database

conn = DriverManager.getConnection("jdbc:mysql://localhost/mydatabase", "username", "password");

// Create a statement

stmt = conn.prepareStatement("UPDATE mytable SET email = ? WHERE name = ?");

// Set the parameters

stmt.setString(1, "johndoe@gmail.com");

stmt.setString(2, "John Doe");

// Execute the statement

stmt.executeUpdate();

} catch (SQLException e) {

// Handle the exception

} finally {

// Close the connection

if (conn != null) {

try {

conn.close();

} catch (SQLException e) {

// Handle the exception

}

}

}

%>

Cancellation :

<%@ page import="java.sql.\*" %>

<%

Connection conn = null;

PreparedStatement stmt = null;

try {

// Get a connection to the database

conn = DriverManager.getConnection("jdbc:mysql://localhost/mydatabase", "username", "password");

// Create a statement

stmt = conn.prepareStatement("DELETE FROM mytable WHERE name = ?");

// Set the parameters

stmt.setString(1, "John Doe");

// Execute the statement

stmt.executeUpdate();

} catch (SQLException e) {

// Handle the exception

} finally {

// Close the connection

if (conn != null) {

try {

conn.close();

} catch (SQLException e) {

// Handle the exception

}

}

}

%>

**Writing space for the Problem:(For Student’s use only)**

**Post Lab Task**

1. Julie needs to set up a startup. Make Registration and login frames for web based shopping entrance made by her. Make an enrollment page. After client registers, demonstrate a login page where client can login just on the off chance that he submits right certifications.

**Writing space for the Problem:(For Student’s use only)**

login.jsp:-

<%@ page import ="java.sql.\*"  %>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-88591" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1" >

</head>

<body>

<form action="LoginRegister">

<table background-color: Lightgreen; margin-left: 20px; margin-left: 20px;>

<tr>

<td>

<h3 style="color: red;">

Login page !!!

</h3>

</td>

<td></td>

</tr>

<tr>

<td>UserName :</td>

<td><input type="text" name="username"></td>

</tr><tr>

<td>

Password :</td><td>

<input type="password" name="password"></td>

</tr><tr>

<td><input type="submit" name="submit" value="Login"></td>

<td><a href="register.jsp">

Registration</a>

</tr>

</table>

</form>

</body>

</html>

register.jsp:-

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

 pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" >

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>registration</title>

</head>

<body>

<form action="LoginRegister">

<table background-color:skyblue; margin-left: 20px; margin-left: 20px;>

<tr>

<td><h3 style="color: black;">Registration page !!!</h3></td>

<td></td>

</tr><tr>

<td>UserName :</td>

<td><input type="text" name="username"></td>

</tr><tr>

<td>Name :</td>

<td><input type="text" name="name"></td>

</tr><tr>

<td>Password :</td>

<td><input type="password" name="password1"></td>

</tr><tr>

<td>Re-Type Password :</td>

<td><input type="password" name="password2"></td>

</tr><tr>

<td><input type="submit" name="submit" value="Register"></td>

<td></td>

</tr>

</table>

</form>

</body>

</html>

**Writing space for the Problem:(For Student’s use only)**

*(For Evaluator’s use only)*

Comment of the Evaluator (if Any)

Evaluator’s Observation Marks Secured: out of

Full Name of the Evaluator:

Signature of the Evaluator Date of Evaluation:

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# JSP(JAVA SERVER PAGE)-2 #8

**Date of the Session: / / Time of the Session: to**

**Prerequisite:**

* Basic idea on Java Server pages and Action tags

## Pre-Lab Task:

* 1. Write the Syntax for following Action tags:
     1. <jsp : include>

includes another resource.

<jsp:include page="relativeURL | <%= expression %>">

<jsp:param name="parametername" value="parametervalue | <%=expression%>" />

</jsp:include>

* + 1. <jsp : forward>

forwards the request and response to another resource.

<jsp:forward page="relativeURL | <%= expression %>">

<jsp:param name="parametername" value="parametervalue | <%=expression%>" />

</jsp:forward>

* + 1. <jsp : param>

sets the parameter value. It is used in forward and include mostly.

<jsp:param name="parametername" value="parametervalue | <%=expression%>" />

* + 1. <jsp : usebean>

creates or locates bean object.

<jsp:useBean id= "instanceName" scope= "page | request | session | application"

class = "packageName.className" type="packageName.className"

beanName="packageName.className | <%= expression >" >

</jsp:useBean>

* + 1. <jsp : getproperty>

prints the value of property of the bean.

<jsp:getProperty name="instanceOfBean" property="propertyName" />

* + 1. <jsp : setproperty>

sets the value of property in bean object.

<jsp:setProperty name="instanceOfBean" property= "\*" |

property="propertyName" param="parameterName" |

property="propertyName" value="{ string | <%= expression %>}"

/>

1. Explain about MVC Design pattern in JSP with a neat Diagram.

MVC stands for Model View and Controller. It is a design pattern that separates the business logic, presentation logic and data.

Controller acts as an interface between View and Model. Controller intercepts all the incoming requests.

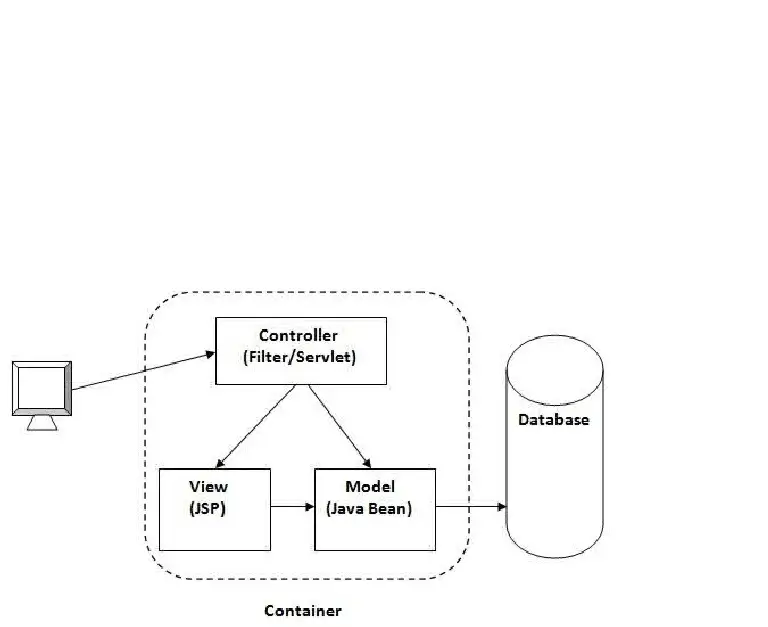
Model represents the state of the application i.e. data. It can also have business logic.

View represents the presentation that is UI(User Interface).

Advantage of MVC (Model 2) Architecture

1.Navigation Control is centralized

2.Easy to maintain the large application



## In Lab Task:

* 1. Develop an application which displays login form and login options such as login through facebook (facebook.jsp) and login through gmail (gmail.jsp) using JSP action tags like <jsp:include>, <jsp: forward>, Which takes input from first jsp and load the data AND Insert it into First JSP table, so that the user can login through the login.jsp next time.

**Writing space for the Problem:(For Student’s use only)**

check.jsp:-

<%

String x=request.getParameter("x");

String u=request.getParameter("uname");

String p=request.getParameter("pwd");

if(x.equals("LOGIN")){%>

<jsp:forward page="login.jsp">

<jsp:param name="u" value ="<%= u%>"/>

<jsp:param name="p" value ="<%= p%>"/>

</jsp:forward>

<%}else if(x.equals("LOGIN through FACEBOOK")){%>

<jsp:forward page="facebook.jsp">

<jsp:param name="u" value ="<%= u%>"/>

<jsp:param name="p" value ="<%= p%>"/>

</jsp:forward>

<%

}

else if(x.equals("LOGIN through GMAIL")){

%>

<jsp:forward page="gmail.jsp">

<jsp:param name="u" value ="<%= u%>"/>

<jsp:param name="p" value ="<%= p%>"/>

</jsp:forward>

<%

}

%>

facebook.jsp:-

<%@ page import="java.sql.\*" %>

<%@ page import="java.io.\*" %>

<%

try{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

String u= request.getParameter("u");

String p= request.getParameter("p");

PreparedStatement ps =con.prepareStatement("select \* from facebook where uname=? and  pwd=?");

ps.setString(1,u);

ps.setString(2,p);

ResultSet rs=ps.executeQuery();

if(rs.next())out.println("Facebook Login Success");

else{out.println("Wrong Credentials");

%>

<jsp:include page="index.html"/>

<%

}

} catch(Exception e){

out.println("Exception is "+e); }%>

gmail.jsp:-

<%@ page import="java.sql.\*" %>

<%@ page import="java.io.\*" %>

<%

 try{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

String u= request.getParameter("u");

String p= request.getParameter("p");

PreparedStatement ps =con.prepareStatement("select \* from gmail where uname=? and pwd=?");

ps.setString(1,u);

 ps.setString(2,p);

ResultSet rs=ps.executeQuery();

if(rs.next())

out.println("Gmail Login Success");

else{

out.println("Wrong Credentials");

%>

<jsp:include page="index.html"/>

<%

}

} catch(Exception e){

out.println("Exception is "+e);

}%>

login.jsp:-

<%@ page import="java.sql.\*" %>

<%@ page import="java.io.\*" %>

<% try{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

String u= request.getParameter("u");

String p= request.getParameter("p");

PreparedStatement ps =con.prepareStatement("select \* from login where uname=? and pwd=?");

ps.setString(1,u);

ps.setString(2,p);

ResultSet rs=ps.executeQuery();

if(rs.next())

out.println("Login Success");

else{

out.println("Wrong Credentials");

%>

<jsp:include page="index.html"/><%

}

} catch(Exception e){

out.println("Exception is "+e);

}

%>

index.html:-

<html>

<head>

<title>LOGIN FORM</title>

</head>

<h2> LOGIN FORM</h2>

<body>

<form action = "check.jsp" method = "post">

username:<input type="text" name="uname">

<br>

 password:<input type="password" name="pwd">

<br>

<input type="submit" name="x" value="LOGIN">

<br>

<input type="submit" name="x" value="LOGIN through FACEBOOK">

<br>

<input type="submit" name="x" value="LOGIN through GMAIL">

<br>

</form>

</body>

</html>

**Writing space for the Problem:(For Student’s use only)**

* 1. Develop an MVC application such that it takes 2 values and the response with the help of buttons which determines the operation to be performed on the values. A controller must be used to transfer values and button response to model which performs actual Business logic and sends the result along with button response to the view to display the result accordingly.

**Writing space for the Problem:(For Student’s use only**

control.jsp:-

<%String x=request.getParameter("op1");

String y=request.getParameter("op2");

String z=request.getParameter("ty");

String ch="ADD";

if(z.equals(ch))

{

%>

<jsp:forward page="add.jsp">

<jsp:param name="a" value="<%=x %>" />

<jsp:param name="b" value="<%=y %>" />

</jsp:forward>

<%

}

else{

%>

<jsp:forward page="sub.jsp">

<jsp:param name="a" value="<%=x %>" />

<jsp:param name="b" value="<%=y %>" />

</jsp:forward><%}%>

add.jsp:-

<%int x=Integer.parseInt(request.getParameter("a"));

int y=Integer.parseInt(request.getParameter("b"));

int res=x+y;

%>

<jsp:forward page="view.jsp">

<jsp:param name="result" value="<%=res %>"

/>

</jsp:forward>

sub.jsp:-

<%int x=Integer.parseInt(request.getParameter("a"));

int y=Integer.parseInt(request.getParameter("b"));

int res=x-y;

%>

<jsp:forward page="view.jsp">

<jsp:param name="result" value="<%=res %>"

/>

</jsp:forward>

test.jsp:-

<%String x=request.getParameter("op1");

String y=request.getParameter("op2");

String z=request.getParameter("ty");

out.println(z);

%>

17CS3116 ENTERPRISE PROGRAMMING

view.jsp:-

<%String r=request.getParameter("result");

String b=request.getParameter("ty");

String ch="ADD";

if(b.equals(ch)){

out.println("Addition of two numbers is "+r);

}

else{

out.println("Subtraction of two numbers is "+r);

}

%>

<jsp:include page =”index.html”/>

**Writing space for the Problem:(For Student’s use only)**

## Post Lab Task:

ST. MARYS school Administrator is keen about the details of a student who is joining, so the students are requested to Fill the Name, Password, Email using index.html and develop a User.java file which is a bean class having setters and getters to set the values and using process.jsp which sets the incoming values to bean object and prints the values.

**Writing space for the Problem:(For Student’s use only)**

index.html:-

<html>

<body>

<form acon="process.jsp" method="post">

Name:<input type="text" name="name">

<br>

Password:<input type="password" name="password">

<br>

Email:<input type="text" name="email">

<br>

<input type="submit" value="register">

</form>

</body>

</html>

User.java:-

public class User {

private String name,password,email;

//seers and geerspublic void setName(String name) {

 this.name = name;

}

public void setPassword(String password) {

 this.password = password;

 }

public void setEmail(String email) {

 this.email = email;

}

public String getName() {

return name;

 }

public String getPassword() {

 return password;

}

public String getEmail() {

 return email;

 }

}

process.jsp:-

<jsp:useBean id="u" class="User">

</jsp:useBean>

<jsp:setProperty property="\*" name="u"/>

Record:<br>

<jsp:getProperty property="name" name="u"/>

<br>

<jsp:getProperty property="password"name=”u”/><br>

<jsp:getproperty property=”email”name=”u”/><br>

**Writing space for the Problem:(For Student’s use only)**

*(For Evaluator’s use only)*

Signature of the Evaluator Date of Evaluation:

Evaluator’s Observation

Marks Secured: out of

Full Name of the Evaluator:

Comment of the Evaluator (if Any)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# SESSION HANDLING#9

**Date of the Session: / / Time of the Session: \_to**

**Prerequisite:**

* Basic idea on Session handling techniques.

## Pre-Lab Task:

* + 1. What is session tracking? Why is session tracking used?

Http is a stateless protocol. It doesn’t store state of any user sending request to server. It treats every user as new user for every user. Session tracking is way of maintain data of the user. It contains details like session id, lifetime etc. It is used to identify state of any user.

* + 1. What are different types of session tracking techniques? Write few real-life examples for each session tracking technique.

Types of session tracking techniques

* 1. cookies-Gmail
  2. Hidden form field- viewing items in Shopping cart. (if user wants to add more items to the cart, present items are sent in hidden fields.)c)URL rewriting- in any regular websites without any sensitive informationd) Http session – Used for online fund transaction
     1. Understand how a cookie works. Write syntax to create a cookie and store values, access and delete them.

Cookie is a small piece of information which is used to identify every user sending request to server. When user sends a request, we add some details like unique id, lifetime etc to the response from servlet.  Next time a request is sent, this cookie is added by default recognizing user as an old one.Creating a cookie:

Cookie ck=new Cookie(“userid”,”123”);

Response.addCookie(ck);

Use Cookie:

Cookie ck=request. getCookies();

For (int i=0; i<ck.length(); i++)

System.out.println(ck[i].getName()+” ”+ck[i].getValue());

Delete cookie:

Cookie ck=new Cookie(“userid”,””);

Ck.setMaxAge(0);

Response.addCookie(ck);

## In Lab Task:

1. Kamal needs to login to his Gmail account but he forgot his secret password. Create a login form where he enters his username and password. As password is wrong, show a message which says “Wrong Password” and forgot password link is shown. When user clicks it, use URL redirect method to display username entered in login form and change his password by displaying password field.

**Writing space for the Problem:(For Student’s use only)**

Index.html

<!DOCTYPE html>

<html>

<head>

<meta charset= "ISO-8859-1" >

<title>Insert title here</title>

</head>

< body>

<form method= "post" action="PwdCheck" >

name:<input type= "text" name="uname" />

 password<input type= "text"  name="pwd" />

<input type= "submit" value="login" />

</form>

</ body>

</html>

PwdCheck.java:-

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;/\*\* \* Servlet implementation class PwdCheck  \*/

public class PwdCheck extends HttpServlet {

private static final long serialVersionUID = 1L;

 /\*\*

 \* @see HttpServlet#HttpServlet() \*/

Public PwdCheck() {

super();

 // TODO Auto-generated constructor stub

 }

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response) \*/

Protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

 ServletException, IOException {

// TODOAuto-generated method stub

//response.getWriter().append("Served at:").append(request.getContextPath());

}

/\*\*

 \* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

 \*/

Protected void doPost(HttpServletRequest request, HttpServletResponse response) throws

 ServletException, IOException {

// TODOAuto-generated method stub

//doGet(request, response);

PrintWriter out=response.getWriter();

try

 {

response.setContentType("text/html");

String uname=request.getParameter("uname");

String  pwd=request.getParameter("pwd");

if

( pwd.equals("Admin")) {

out.println("Welcome "+uname);

}

else{

out.println("Wrong Password");

out.println("please try again");

out.println("<form method=\"post\" action=\"PwdCheck\">\r\n" +

"name:<input type=\"text\" name=\"uname\"/>\r\n" +

"password<input type=\"text\" name=\"pwd\"/>\r\n" +

"<input type=\"submit\" value=\"submit\"/>\r\n" +

"</form>");

out.println("<a href='PwdChange?uname="+uname+"'>Forgot Password</a>");

out.close();

}

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

}

}

}

PwdChange.java:-

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;/\*\* \* Servlet implementation class PwdChange \*/

public class PwdChange extends HttpServlet {

private static final long serialVersionUID= 1L;

 /\*\*

 \* @see HttpServlet#HttpServlet() \*/

Public PwdChange() {

super();

 // TODO Auto-generated constructor stub }

/\*\*

 \* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

Protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

 ServletException, IOException {

// TODO Auto-generated method stub

//response.getWriter().append("Served at: ").append(request.getContextPath());

PrintWriter out=response.getWriter();

try {

response.setContentType("text/html");

String uname=request.getParameter("uname");

out.println("Hello "+uname);

out.println("Please enter new password");

out.println("<form method=\"post\" action=\"PwdCheck\">\r\n" +

"new password<input type=\"text\" name=\"pwd\"/>\r\n" +

"<a href=\'index.html\'>submit</a>\r\n" +

"</form>");

}catch(Exception e) {

out.println(e);

}

}/\*\*

 \* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

Protected void doPost(HttpServletRequest request, HttpServletResponse response) throws

 ServletException, IOException {

// TODO Auto-generated method stub//doGet(request, response);}}

Web.xml:-

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

<web-app xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/webapp\_2\_5.xsd" id="WebApp\_ID" version="2.5" >

 <display-name>UrlRewriting</display-name>

<welcome-file-list>

 <welcome-file>index.html</welcome-file>

 <welcome-file>index.htm</welcome-file>

 <welcome-file>index.jsp</welcome-file>

 <welcome-file>default.html</welcome-file>

 <welcome-file>default.htm</welcome-file>

 <welcome-file>default.jsp</welcome-file>

 </welcome-file-list>

 <servlet>

 <description></description>

 <display-name>PwdCheck </display-name>

 <servlet-name>PwdCheck </servlet-name>

 <servlet-class>PwdCheck </servlet-class>

 </servlet>

 <servlet-mapping>

 <servlet-name>PwdCheck </servlet-name>

<url-pattern>/PwdCheck </url-pattern>

</servlet-mapping>

 <servlet>

 <description></description>

<display-name>PwdChange</display-name>

 <servlet-name>PwdChange</servlet-name>

 <servlet-class>PwdChange</servlet-class>

 </servlet>

 <servlet-mapping>

 <servlet-name>PwdChange</servlet-name>

<url-pattern>/PwdChange</url-pattern>

</servlet-mapping>

</web-app>

**Writing space for the Problem:(For Student’s use only)**

1. Udemy is a site which gives online courses of various domains. This site chooses the cost of the course dependent on number of visits to the site from your account. If number of visits are more, cost is more. Discover number of times client tapped on any course from his account.

**Writing space for the Problem:(For Student’s use only)**

Index.html:-

<!DOCTYPE html>

<html>

<head>

<meta charset= "ISO-8859-1" >

<title>Insert title here</title>

</head>

< body>

<a href = 'HitCounter' >

Click me

</a>

</ body>

</html>

NumberOfClicks.java

Import  java.io.IOException;

Import java.io.PrintWriter;

import javax.servlet.ServletConfig;

import javax.servlet.ServletException;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;/\*\* \* Servlet implementation class HitCounter  \*/

public class HitCounter

extends HttpServlet {

private static final long serialVersionUID = 1L;

private int y;

 /\*\*

 \* @see HttpServlet#HttpServlet() \*/

public HitCounter() {

super();

 // TODO Auto-generated constructor stub

 }

/\*\*

 \*

@see Servlet#init(ServletConfig) \*/@SuppressWarnings("unused")

Public void init(ServletConfig config) throws

 ServletException {

// TODOAuto-generated method stub

Int y;y=0;

//x=Integer.toString(y);

//Cookie ck=new Cookie("num",x);

}

/\*\*

 \*

@see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response) \*/

Protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

 ServletException, IOException {

// TODOAuto-generated method stub

//response.getWriter().append("Served at: ").append(request.getContextPath());y++;

PrintWriter out=response.getWriter();

response.setContentType("text/html");

out.println("No of clicks = "+y);

}

/\*\*

 \*

@see  HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response) \*/

Protected void doPost(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

// TODOAuto-generated method stubdoGet(request, response);

}

}

**Writing space for the Problem:(For Student’s use only)**

**Writing space for the Problem:(For Student’s use only)**

## Post Lab Task:

* 1. Sharan needs to pay his college fee on the web. So, he opens online banking site and logins with right accreditations. In the wake of signing in, show another structure which contains fields like username, college name, amount to be paid. After that, put username, school name and amount to be paid in session scope. Show a message "You are paying to 'college name' Rs. 'amount' " and a continue button. If client clicks continue, show a bill and end the session.

**Writing space for the Problem:(For Student’s use only)**

Index.html:-

<!DOCTYPE html>

<html >

<head>

<meta charset= "ISO-8859-1" >

<title>Insert title here</title>

</head>

< body>

<form action= "FeeStruc"  method="post" >

name:<input type= "text" name="uname" />

 password<input type= "password" name="pwd" />

<input type= "submit" value="login" />

</form>

</ body>

</html>

FeeStruc.java:-

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import

 javax.servlet.http.HttpServlet;

import

 javax.servlet.http.HttpServletRequest;

import

 javax.servlet.http.HttpServletResponse;/\*\* \* Servlet implementation class FeeStruc \*/

public class FeeStruc extends HttpServlet {

private static final long serialVersionUID= 1L;

 /\*\*

 \* @see HttpServlet#HttpServlet() \*/

Public FeeStruc() {

super();

 // TODO Auto-generated constructor stub

 }

/\*\*

 \* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response) \*/

Protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

 ServletException, IOException {

// TODO Auto-generated method stub//

response.getWriter().append("Served at: ").append(request.getContextPath());

}/\*\*

 \* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response) \*/

Protect void doPost(HttpServletRequest request, HttpServletResponse response) throws

 ServletException, IOException {

// TODO Auto-generated method stub

//doGet(request, response);

PrintWriter out=response.getWriter();

response.setContentType("text/html");

try{

String name=request.getParameter("uname");

String  pwd=request.getParameter("pwd");

if (name.equals("Me"))

{

if ( pwd.equals("MyPassword")) {

out.println("Welcome "+name+"\nPlease fill the details");

out.println("<form action=\"FeeSession\" method=\"post\">\r\n" +

"name<input type=\"text\" name=\"uname\"/>\r\n" +

"college<input type=\"text\" name=\"clg\"/>\r\n" +

"amount<input type=\"text\" name=\"amt\"/>\r\n" +

"<input type=\"submit\" value=\"pay\"/>\r\n" +

"</form>");

}

}

else{

out.println("Wrong Password..Try again\n");

out.println("<form action=\"FeeStruc\" method=\"post\">\r\n" +

"name:<input type=\"text\" name=\"uname\"/>\r\n" +

"password<input type=\"password\" name=\"pwd\"/>\r\n" +

"<input type=\"submit\" value=\"login\"/>");

}

}catch

(Exception e) {out.println(e);

}

}

}

FeeSession.java:-

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

  /\*\*

 \* Servlet implementation class FeeSession \*/

Public class FeeSession extends HttpServlet {

Private static final long serialVersionUID= 1L;

 /\*\*

 \* @see HttpServlet#HttpServlet() \*/

public FeeSession() {

super();

 // TODOAuto-generated constructor stub

 }

/\*\*

 \* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response) \*/

Protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

 ServletException, IOException {

// TODOAuto-generated method stub//response.getWriter().append("Servedat:").append(request.getContextPath());

}

/\*\*

 \* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response) \*/

Protected void doPost(HttpServletRequest request, HttpServletResponse response) throws

 ServletException, IOException {

// TODO Auto-generated method stub//doGet(request, response);

PrintWriter out=response.getWriter();

response.setContentType("text/html");

try{

String name=request.getParameter("uname");

String clg=request.getParameter("clg");

String amt=request.getParameter("amt");

HttpSession session=request.getSession();

session.setAttribute("name",name);

session.setAttribute("clg",clg);

session.setAttribute("amt",amt);

out.println("You are going to pay to "+clg+" an amount of "+amt);

out.println("\nDo you want to continue?");

out.println("<form action=\"Bill\" method=\"post\">\r\n" +

"<input type=\"submit\" value=\"yes\"/>\r\n" +

"</form>\r\n" +

"<form action=\"FeeStruc\" method=\"post\">\r\n" +

"<input type=\"submit\" value=\"no\"/>\r\n" +

"</form>");

}catch

(Exception e) {out.println(e);

}

}

}

Bill.java:-

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\* \* Servlet implementation class Bill \*/

public class Bill extends HttpServlet {

private static final long serialVersionUID = 1L;

 /\*\*

 \* @see HttpServlet#HttpServlet() \*/

public Bill() {

super();

 // TODOAuto-generated constructor stub

 }

/\*\*

 \* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response) \*/

Protected void doGet(HttpServletRequest request, HttpServletResponse response) throws

 ServletException, IOException {

// TODOAuto-generated method stub//response.getWriter().append("Servedat").append(request.getContextPath());

}

/\*\*

 \* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response) \*/

Protected void doPost(HttpServletRequest request, HttpServletResponse response) throws

 ServletException, IOException {

// TODO Auto-generated method stub//doGet(request, response);

PrintWriter out=response.getWriter();

response.setContentType("text/html");

try{

HttpSession session=request.getSession();

out.println("Bill");

String name=(String)session.getAttribute("name");

String clg=(String)session.getAttribute("clg");

String amt=(String)session.getAttribute("amt");

out.println("</br>PAYMENT ID:"+session.getId());

out.println("</br>NAME:"+name);

out.println("</br>COLLEGE:"+clg);

out.println("</br>AMOUNT:"+amt);

session.invalidate();

}catch (Exception e) {

out.println(e);

}

}

**Writing space for the Problem:(For Student’s use only)**

**Writing space for the Problem:(For Student’s use only)**

*(For Evaluator’s use only)*

Evaluator’s Observation Marks Secured: out of

Full Name of the Evaluator:

Signature of the Evaluator Date of Evaluation:

Comment of the Evaluator (if Any)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# HIBERNATE-1#10

**Date of the Session: / / Time of the Session: \_to**

**Prerequisite:**

* Basic idea on Hibernate

## Pre-Lab Task:

1. What is ORM tool?

RM stands for Object-Relational Mapping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages such as Java, C#, etc. An ORM tool simplifies the data creation, data manipulation and data access

1. Disadvantages of JDBC/ Advantages of Hibernate/ Difference between the JDBC and Hibernate

Disadvantages of JDBC:

1. Boilerplate code: JDBC requires writing a significant amount of boilerplate code for establishing database connections, executing queries, handling result sets, and managing transactions. This can make the codebase complex and error-prone.

2. Object-relational impedance mismatch: JDBC deals with SQL databases, which have a relational data model, while most modern object-oriented programming languages use an object model. This impedance mismatch can result in complex mapping between objects and database tables, making it tedious to work with data.

3. Tedious transaction management: JDBC requires manual management of database transactions, including handling commit and rollback operations. This can be error-prone and lead to inconsistencies in the data if not implemented correctly.

Advantages of Hibernate:

1. Object-relational mapping (ORM): Hibernate is an ORM framework that simplifies the mapping between Java objects and database tables. It eliminates the need for writing low-level SQL queries and handles the object-relational impedance mismatch transparently. This makes it easier to work with data in an object-oriented manner.

1. Productivity and maintainability: Hibernate reduces the amount of boilerplate code required compared to JDBC. It provides a higher level of abstraction, allowing developers to focus more on business logic rather than database interactions. This can improve productivity and make the codebase more maintainable.

2. Automatic transaction management: Hibernate provides built-in transaction management, automatically handling commit and rollback operations. This simplifies the handling of database transactions, reducing the chances of errors and ensuring data integrity.

Difference between JDBC and Hibernate:

2. Mapping: With JDBC, developers need to write explicit SQL queries and manually map the result sets to Java objects. Hibernate, as an ORM framework, handles the mapping between Java objects and database tables automatically, using annotations or XML configurations.

3. Query language: JDBC uses SQL for querying databases, which requires writing and managing SQL statements. Hibernate provides HQL (Hibernate Query Language), an object-oriented query language that is similar to SQL but operates on persistent objects and their properties.

4. Caching and performance optimization: Hibernate offers caching mechanisms that can improve performance by reducing the number of database trips. JDBC does not provide built-in caching capabilities, and performance optimization is left to the developer's implementation.

1. Level of abstraction: JDBC is a low-level API that requires writing SQL queries and handling database connections, transactions, and result sets manually. Hibernate, on the other hand, is a higher-level ORM framework that abstracts away many of these low-level details, providing a more intuitive and object-oriented approach to working with databases.

1. Name the 4 layers of hibernate architecture

The four layers of Hibernate architecture:

Java application layer

Hibernate framework layer

Backhand api layer

Database layer

1. Write the syntax of mapping of xml and configuration of xml?

Syntax of Mapping xml

 <hibernate-mapping>

<class name="POJO class name" table="table name in database">

<id name="variable name" column="column name in database" type="java/hibernate type" />

<property name="variable1 name" column="column name in database" type="java/hibernate type"/>

<property name="variable2 name" column="column name in database" type="java/hibernate

type" />

</class>

</hibernate-mapping>

Syntax of Configuration xml

<hibernate-configuration>

<session-factory>

<!-- Related to the connection START -->

<property name="connection.driver\_class">Driver Class Name </property>

<property name="connection.url">URL </property>

<property name="connection.user">user </property>

<property name="connection.password">password</property>

<!-- Related to the connection END -->

<!-- Related to hibernate properties START -->

<property name="show\_sql">true/false</property>

<property name="dialet">Database dialet class</property>

<property name="hbm2ddl.auto">create/update or what ever</property>

<!-- Related to hibernate properties END-->

<!-- Related to mapping START-->

<mapping resource="hbm file 1 name .xml" / >

<mapping resource="hbm file 2 name .xml" / >

<!-- Related to the mapping END -->

</session-factory>

</hibernate-configuration>

## In Lab Task:

1. Sreenivas, the proprietor of a rice mill, needs to keep up the information about the rice bags produced at his mill. He wants to save the cost, type of the rice (polished/ non-polished) and amount in kilograms for each bag. Write a hibernate application to insert the details of bags manufactured. The application should ask the user whether he wants to insert a details of a bag each time until he says 'no'. For every 'yes' it should gather the details of the bag i.e, the id number, amount, cost and type.

**Writing space for the Problem:(For Student’s use only)**

POJO file(Ricebag.java):-

 package lab10in1;

 public class Ricebag {

private int id;

 private int quantity;

private double cost;

 private String type;

 public int getId() {

return id;

}

 public void setId(int id) {

this.id = id;

}

 public int getQuantity() {

return quantity;

}

 public void setQuantity(int quantity) {

this.quantity = quantity;

}

 public double getCost() {

return cost;

}

 public void setCost(double cost) {

this.cost = cost;

}

 public String getType() {

return type;

}

 public void setType(String type) {

this.type = type;

}

}

MAPPING file(ricebag.hbm.xml):-

 package lab10in1;

 public class Ricebag {

 private int id;

 private int quantity;

 private double cost;

private String type;

public int getId() {

return id;

}

 public void setId(int id) {

this.id = id;

}

 public int getQuantity() {

return quantity;

}

 public void setQuantity(int quantity) {

this.quantity = quantity;

}

 public double getCost() {

return cost;

}

 public void setCost(double cost) {

this.cost = cost;

}

 public String getType() {

return type;

}

 public void setType(String type) {

this.type = type;

}

}

CONFIGURATION file(hibernate.cfg.xml):-

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

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- connection details -->

<property name = "connection.driver\_class" >oracle.jdbc.driver.OracleDriver</property>

<property name = "connection.url" >jdbc:oracle:thin:@localhost:1522:xe</property>

<property name="connection.user" >system</property>

<property name= "connection.password" >system</property>

<!-- Hibernate details --><property name="show\_sql" >true</property>

<property name="hbm2ddl.auto" >update</property>

<property name="dialect" >org.hibernate.dialect.OracleDialect</property>

<!-- mapping resources -->

<mapping resource = "ricebag.hbm.xml" />

</session-factory>

</hibernate-configuration>

LOGIC file:-

 package lab10in1;

import java.util.Scanner;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

 public class InsertDetails {

 public static void main(String[]args){

Boolean continues = true;

Scanner sc = new Scanner(System.in);

Configuration cfg= new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf= cfg.buildSessionFactory();

Session s = sf.openSession();

while(continues){

Ricebag r= new Ricebag();

System.out.println("Enter the id of the rice bag");

int bid =sc.nextInt();

System.out.println("Enter the quantity of rice bag in kgs");

int amt=sc.nextInt();

System.out.println("Enter the cost of rice bag");

double cst=sc.nextDouble();

System.out.println("Enter the type of rice bag(polished/nonpolished)");

String t=sc.next();

r.setId(bid);

r.setQuantity(amt);

r.setCost(cst);

r.setType(t);

s.save(r);

Transaction tx= s.beginTransaction();

tx.commit();

System.out.println("Do you want to insert(yes/no) :" );

String proceed=sc.next();

if(proceed.equals("yes")){

continues=true;

}

else{

continues=false;

}

}

System.out.println("Data has been entered");

s.close();

}

}

**Writing space for the Problem:(For Student’s use only)**

1. Now Sreenivas, wants to sell a bag to a customer. he knows the id number of the bag and needs to retrieve the cost. Sreenivas doesn't encourage bargaining. So if the customer wants to buy the bag at the same value then he sells the bag and erases information about that bag. Write the hibernate application which asks the id number of the bag, displays the price of that bag, and then asks whether the customer wants to purchase the bag or not. If 'yes' then the details of the bag should be erased. (Note: Write only the java file with logic. While executing, implement the POJO, mapping, configuration files of previous question, since we are operating on same data)

**Writing space for the Problem:(For Student’s use only)**

package lab10in1;

import java.util.Scanner;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

 public class SellBag {

public static void main(String[]args){

Scanner sc = new Scanner(System.in);

Configuration cfg = new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf=cfg.buildSessionFactory();

Session s= sf.openSession();

System.out.println("Enter the id of the rice bag");

int x = sc.nextInt();

Object o = s.load(Ricebag.class,new Integer(x));

Ricebag r = (Ricebag)o;

System.out.println("the cost of the rice bag is :"+r.getCost());

System.out.println("Want to purchase the bag(yes/no)");

String y = sc.next();

if(y.equals("yes")) {

Transaction tx = s.beginTransaction();

s.delete(r);

tx.commit();

System.out.println("the bag is sold");

}

else {

System.out.println("Not sold");

}

s.close();

}

}

## Post Lab Task:

1. Gokul Fabrics has wide range of various fabrics in different colours. The owner of the shop needs to keep up the information of the fabrics he have. He want to save the type, colour, available length and cost per metre for each fabric. He also wants to update the length of the fabric when he sells it to the customer.
   1. Write a Hibernate application to insert the details of the fabric.
   2. Write a Hibernate application which requests the id number of the fabric and its length sold, displays the total selling cost and then updates the data with the remaining length.

**Writing space for the Problem:(For Student’s use only)**

POJO file(GokulFabrics.java):-

 package lab10post;

 public class GokulFabrics {

 private int fid;

 private String type;

 private double availability;

private double costpermetre;

public int getFid() {

return fid;

}

 public void setFid(int fid) {

this.fid = fid;

}

 public String getType() {

return type;

}

 public void setType(String type) {

this.type = type;

}

 public double getAvailability() {

return availability;

}

 public void setAvailability(double availability) {

this.availability = availability;

}

 public double getCostpermetre() {

return costpermetre;

}

 public void setCostpermetre(double costpermetre) {

this.costpermetre = costpermetre;

}

}

Mapping file(gokulfabrics.hbm.xml):-

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

<!DOCTYPE hibernate-mapping PUBLIC"-

//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="lab10post.GokulFabrics" table="fabrics" >

<id name = "fid"  column="idno" >

<generator class = "assigned" />

</id><property name = "type"  column="type" />

<property name = "availability" column="length" />

<property name = "costpermetre" column="cost" />

</class>

</hibernate-mapping>

Configuration file(hibernate.cfg.xml)

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

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- connection details -->

<property name = "connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

<property name = "connection.url" >jdbc:oracle:thin:@localhost:1522:xe</property>

<property name="connection.user" >system</property>

<property name= "connection.password" >system</property>

<!-- Hibernate details --><property name="show\_sql" >true</property>

<property name="hbm2ddl.auto" >update</property>

<property name="dialect" >org.hibernate.dialect.OracleDialect</property>

<!-- mapping resources -->

<mapping resource = "gokulfabrics.hbm.xml" />

</session-factory>

</hibernate-configuration>

LOGIC file:-

  package lab10post;

import java.util.Scanner;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

import lab10in1.Ricebag;

 public class InsertFabric {

 public static void main(String[]args){

Scanner sc= new Scanner(System.in);

Boolean continues=true;

Configuration cfg= new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf= cfg.buildSessionFactory();

Session s = sf.openSession();

while(continues){

GokulFabrics g= new GokulFabrics();

System.out.println("Enter the id of the fabric");

int id =sc.nextInt();

System.out.println("Enter the type of fabric");

String type=sc.next();

System.out.println("Enter the availability of fabric in metres");

double avail=sc.nextDouble();

System.out.println("Enter the cost per metre ");

Doublec=sc.nextDouble();

g.setFid(id);

g.setType(type);

g.setAvailability(avail);

g.setCostpermetre(c);

s.save(g);

Transaction tx= s.beginTransaction();

tx.commit();

System.out.println("Do you want to insert(yes/no) :" );

String proceed=sc.next();

if(proceed.equals("yes")){

continues=true;

}

else{

continues=false;

}

}

System.out.println("Data has been entered");

s.close();

}

}

LOGIC file:-

 package lab10post;

import java.util.Scanner;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

 public class UpdateFabric {

 public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out .println("Enter the id of fabric");int x= sc.nextInt();

System.out .println("Enter the length of fabric sold out");

double y = sc.nextDouble();

Configuration cfg = new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf = cfg.buildSessionFactory();

Session s = sf.openSession();

Object o=s.load(GokulFabrics.class,new Integer(x));

GokulFabrics g=(GokulFabrics)o;

Transaction tx= s.beginTransaction();

double sellingcost = g.getCostpermetre()\*y;

System.out .println("The selling cost of fabric is :"+sellingcost);

double rem=g.getAvailability()-y;

g.setAvailability(rem);

tx.commit();

System.out .println("Data is updated");

s.close();

}

}

**Writing space for the Problem:(For Student’s use only)**

**Writing space for the Problem:(For Student’s use only)**

*(For Evaluator’s use only)*

Evaluator’s Observation Marks Secured: out of

Full Name of the Evaluator:

Signature of the Evaluator Date of Evaluation:

Comment of the Evaluator (if Any)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# HQL, HCQL #11

**Date of the Session: / / Time of the Session: \_to**

**Prerequisite:**

* Basic idea on Hibernate Query Language, Query Interface
* Basic idea on hibernate Criteria Query Language

## Pre-Lab Task:

1. Write the HQL query to find the count of red colored bottles and of quantity 1 litre.

“SELECT count(\*) FROM Bottle AS B WHERE B.colour=’red’ and B.quantity=1”

1. Write the HQL query to get data of the first 10 bottles which are microwave safe ( i.e, the value of microwave safe is “yes”) in ascending order with respect to the quantity.

String hql=”Select” from Bottlr As B WHERE B.microwaveSafe=’yes’ ORDER BY B.quantity ASC”;

query query=session.createQuery(hql);

query.setFirstResult(0);

query.setResults(10);

1. Write the HCQL query to get the 15th to 30th record.

Crietria c=session.createCriteria(Emp.

class

); c.setFirstResult(15); c.setMaxResult(30); List list=c.list();

1. Write the HCQL query to get the records who salary is greater than 150000.

Crietria c=session.createCriteria(Emp.class);

c.add(Restrictions.gt("salary",150000));

List list=c.list();

1. Write the HCQL query to get the records in ascending order on the basis of salary.

Crietria c=session.createCriteria(Emp.class);

c.addOrder(Order.asc("salary"));

List list=c.list();

## In Lab Task:

1. Stoins is the manager of Minimal Cube company. He maintains the records of employees working in his company. Create a java class where he gets and sets the values of EmpID, EmpName, EmpSalary, EmpAddress. Use Hibernate Frame-work to reduce manual work. When employees are terminated he is likely to delete the record of employee in his database and update the data when it is required. After performing all the operations he is likely to know the employees working in his company at the end of every month, so retrieve the data those who are working in his company. Create separate java class for retrieving, updating and deleting so that the Mr. Stoins can easily work when an employee data needs to update, delete or retrieve. Use Concept of HCQL (Hibernate Criteria Query Language).

**Writing space for the Problem:(For Student’s use only)**

Employee.java:-

 package com.klu.hibernate;

public class Employee {

private int empId;

 private String empName;

private String empSalary;

 private String empAddress;

 public int getEmpId() {

return empId;

}

 public void setEmpId(int empId) {

this.empId = empId;

}

 public String getEmpName() {

return empName;

}

 public void setEmpName(String empName) {

this.empName = empName;

}

 public String getEmpSalary() {

return empSalary;

}

 public void setEmpSalary(String empSalary) {

this.empSalary = empSalary;

}

 public String getEmpAddress() {

return empAddress;

}

 public void setEmpAddress(String empAddress) {

this.empAddress = empAddress;

}

}

employee.hbm.xml:-

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

<!DOCTYPE hibernate-mapping PUBLIC"

-//Hibernate/Hibernate Mapping DTD//EN"

"[http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd](http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd#sent/_blank)">

<hibernate-mapping>

<class name="com.klu.hibernate.Employee">

<id name="empId">

<generator class="increment"></generator>

</id>

<property name="empName"></property>

<property name="empSalary"></property>

<property name="empAddress"></property>

</class>

</hibernate-mapping>

employee.cfg.xml:-

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

<!DOCTYPE hibernate-configuration SYSTEM"

<http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd>">

<hibernate-configuration>

<session-factory>

<property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

<property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

<property name="connection.user">system</property>

<property name="connection.password">manger</property>

<property name="show\_sql">true</property>

<property name="hbm2ddl.auto">update</property>

<property name="dialect">org.hibernate.dialect.OracleDialect</property>

<mapping resource="employee.hbm.xml"></mapping>

</session-factory>

</hibernate-configuration>

Test.java:-

 package com.klu.hibernate;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

 public class Test {

 public static void main(String[] args){

Configuration cfg=new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf=cfg.buildSessionFactory();

Session session=sf.openSession();

Transaction transaction=session.beginTransaction();

Employee employee=newEmployee();

employee.setEmpName("preethika");

employee.setEmpSalary("34000");

employee.setEmpAddress("kavali");

Employee employee2=new Employee();

employee2.setEmpName("Abhi");

employee2.setEmpSalary("98000");

employee2.setEmpAddress("Chennai");

session.save(employee);

session.save(employee2);

transaction.commit();

session.close();

System.out.println("Inserted SuccessFully");

}

}

UpdateQuery.java:-

 package com.klu.hibernate;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

 public class UpdateQuery {

 public static void main(String[] args){

Configuration cfg=new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf=cfg.buildSessionFactory();

Session session=sf.openSession();

Transaction transaction=session.beginTransaction();

Query query=session.createQuery("update Employee set empName=:n where empId=:i");

query.setParameter("n", "Ram");

query.setParameter("i", 2);

intstatus=query.executeUpdate();

System.out.println(status);

transaction.commit();

session.close();

System.out.println("Updated Successfully !!");

}

}

DeleteQuery.java:-

 package com.klu.hibernate;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

 public class DeleteQuery {

public static void main(String[] args){

Configuration cfg=new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf=cfg.buildSessionFactory();

Session session=sf.openSession();

Transaction transaction=session.beginTransaction();

Query query=session.createQuery("delete from Employee where

id=4");query.executeUpdate();

transaction.commit();

session.close();

System.out.println("Deleted a Record Successfully");

}

}

RetriveQuery.java:-

 package com.klu.hibernate;

import java.util.Iterator;

import java.util.List;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

 public class RetrieveQuery {

 public static void main(String[] args){

Configuration cfg=new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf=cfg.buildSessionFactory();

Session session=sf.openSession();

Transaction transaction=session.beginTransaction();

Query query=session.createQuery("from Employee");

List list=query.list();

Iterator itr=list.iterator();

while(itr.hasNext()){

Employee employee=(Employee)itr.next();

System.out.println(employee.getEmpId()+" "+employee.getEmpName()+" "+employee.getEmpSalary()+" "+employee.getEmpAddress());

}

}

}

CriteriaQuery.java:-

 package com.klu.hibernate;

import java.util.Iterator;

import java.util.List;

import org.hibernate.Criteria;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

 public class CriteriaQuery {

 public static void main(String[] args){

Configuration cfg=new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf=cfg.buildSessionFactory();

Session session=sf.openSession();

Transaction transaction=session.beginTransaction();

Criteria criteria=session.createCriteria(Employee.class);

List<Employee> list=criteria.list();

Iterator itr=list.iterator();

while(itr.hasNext()){

Employee employee=(Employee)itr.next();

System.out.println(employee.getEmpId()+" "+employee.getEmpName()+" "+employee.getEmpSalary()+" "+employee.getEmpAddress());

}

}

}

## Post Lab Task:

1. Mr. Deekshit is running an Online Shopping website where shopping take place in his website regularly. So he decided to maintain two different tables namely ORDER containing attributes like ID, ORDERDATE, ORDRENUMBER, CUSTOMERID, TOTALAMOUNT where ID is primary key and another table describing CUSTOMER (ID, FIRSTNAME, LASTNAME, CITY, COUNTRY, PHONE) where ID will not have a null value. By using JOINS through Hibernate Frame-work help Mr.Deekshit to know the details of customer and items customer purchased. Display Order-number, Total-amount, First-name, Last-name, City, Country using ORDER.ID and CUSTOMER.ID.

**Writing space for the Problem:(For Student’s use only)**

import java.io.\*;

 import javax.servlet.\*;

 import javax.servlet.http.\*;

import java.sql.\*;

  public class Display extends HttpServlet {

  public void doGet(HttpServletRequest req, HttpServletResponse res) throws

IOException, ServletException {

 PrintWriter out = res.getWriter();

res.setContentType("text/html");

 try {

 Class.forName("oracle.jdbc.driver.OracleDriver");

 Connectioncon=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","manager");

 sql="SELECT OrderNumber, TotalAmount, FirstName, LastName, City, Country

FROM [OrderDetails] JOIN CustomerDetails ON [OrderDetails].CustomerId = CustomerDetails.Id"; //preparedStatement stmt = con.PrepareStatement(sql);

Statement stmt = com.createStatement();

ResultSet rs = stmt.executeQuery(sql);

 out.println("<table border=1 width=50% height=50%>");

 out.println("<tr><th>EmpId</th><th>EmpName</th><th>Salary</th><tr>");

 while (rs.next()) {

 Integer n = rs.getInt("OrderNumber");

Integer nm = rs.getInt("TotalAmount");

String n = rs.getString("FirstName");

String n = rs.getString("LastName");

String n = rs.getString("City");

String n = rs.getString("Country");

}

 out.println("</table>");

 out.println("</html></body>");

 con.close();

 }

 catch (Exception e) {

 out.println(e);

 }

}

 }

**Writing space for the Problem:(For Student’s use only)**

*(For Evaluator’s use only)*

Evaluator’s Observation

Marks Secured: out of Full Name of the Evaluator:

Signature of the Evaluator Date of Evaluation:

Comment of the Evaluator (if Any)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# INHERITANCE MAPPING #12

**Date of the Session: / / Time of the Session:** \_**to**

**Prerequisite:**

* Basic idea on Hibernate Framework

## Pre-Lab Task:

1. What is Inheritance Mapping and List out Inheritance Mapping Hierarchy.

The mapped superclass strategy is the simplest approach to mapping an inheritance structure to database tables. It maps each concrete class to its own table. That allows you to share the attribute definition between multiple entities.

1.Table Per Hierarchy

2.Table Per Concrete class Hierarchy

3.Table Per Subclass Hierarchy

1. Explain each Type of Hierarchy.

Table per Hierarchy:In table per hierarchy mapping, single table is required to map the whole hierarchy, an extra column (known as discriminator column) is added to identify the class. But nullable values are stored in the table .Table per ConcreteClass Hierarchy:In case of table per concrete class, tables are created as per class. But duplicate columnis added in subclass tables.

Table per subclass Hierarchy:

In this strategy, tables are created as per class but related by foreign key. So there are no duplicate columns

### In Lab Task:

1. A Bank Customer wants to try Inheritance through Table per class Hierarchy so he created a base class named Payment, containing attributes like paymentId, amount and two derived classes namely CreditCard consisting type of CreditCard they used and Cheque containing type of cheque. So, when the object of derived class is saved, object of base class will also get stored in the database in a single table. To know the Object of which class is stored we must use the Discriminator column. Apply Concept of Table per Class Hierarchy through Hibernate Frame Work.

**Writing space for the Problem:(For Student’s use only)**

Payment.java:

package com.minik;

public class Payment{

private int paymentId;

private double amount;

public int getPaymentId() {

return  paymentId;

}

Public void setPaymentId(int  paymentId) {

this. paymentId =  paymentId;

}

Public double getAmount() {

Return amount;

}

Public void setAmount(double amount)

{

this.amount = amount;

}

}

CreditCard.java:-

package com.minik;

public class CreditCard extends Payment{

private String CreditCardType;

public String getCreditCardType()

{

Return CreditCardType;

}

Public void setCreditCardType(String creditCardType)

{

CreditCardType = creditCardType;

}

  }

Cheque.java:-

package minik;

public class Cheque extends Payment{

private String ChequeType;

public String getChequeType() {

return ChequeType;

}

Public void setChequeType(String chequeType)

{

ChequeType = chequeType;

}

}

Customer.java:-

package com.minik;

import org.hibernate.\*;

import org.hibernate.cfg.\*;

public class Customer {

public static void main(String[] args) {

 Configuration cfg = new Configuration();

 cfg.configure("hibernate.cfg.xml");

 SessionFactory factory = cfg.buildSessionFactory();

 Session session = factory.openSession();

 CreditCard c=new CreditCard();

c.setPaymentId(10);

c.setAmount(2500);

 c.setCreditCardType("VisaCard");

 Cheque c1=new Cheque();

 c1.setPaymentId(11);

c1.setAmount(2600);

 c1.setChequeType("ICICI Bank");

 Transaction tx = session.beginTransaction();

 session.save(c);

 session.save(c1);

tx.commit();

 session.close();

 factory.close();

 }

}

Payment.hbm.xml:-

<?xml version= "1.0" ?>

<!DOCTYPE hibernate-mapping PUBLIC "

-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name= "str.Payment" table="PAYMENT" >

<id name= "paymentId"  column="pid"  />

<discriminator  column= "dcolumn"  type="string"  length="5" />

< property name= "amount"  column="amt"  />

<subclass name= "str.CreditCard" discriminator-value="CC" >

< property name= "CreditCardType"  column="cctype" length="10" />

</subclass>

<subclass name= "str.Cheque" discriminator-value="cq" >

< property name= "ChequeType" column="cqtype" length="10" />

</subclass>

</class>

</hibernate-mapping>

Hibernate.cfg.xml:-

<?xml version= '1.0' encoding='UTF-8' ?>

<!DOCTYPE hibernate-configuration PUBLIC "

-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

< property name= "connection.driver\_class" >oracle.jdbc.driver.OracleDriver </ property>

< property name= "connection.url" > jdbc:oracle:thin:@localhost:1521:XE</ property> < property name= "connection.username" >system</ property>

< property name= "connection.password" >manager </ property>



< property name= "dialect" >org.hibernate.dialect.OracleDialect</ property>

< property name= "show\_sql" >true</ property>

< property name= "hbm2ddl" >auto</ property>

<mapping resource= "Payment.hbm.xml" />

</session-factory>

</hibernate-configuration>

1. Mr. Robert Hiezman is a Manager at Company. Create a class where it gets and set the values like name and Id of employee. They hire different types of employees like regular employee who has salary, bonus they get and contract employee who has amount they earn per hour and time duration they work. Use Concept of Table per Concrete Class Hierarchy through Hibernate FrameWork and observe the Structure of data stored in the database.

**Writing space for the Problem:(For Student’s use only)**

Employee.java:-

package com.minik;

public class Employee {

private int id;

private String name;

public int getId() {

return id;}

public void setId(

int id) {

this.id = id;

}

public String getName() {

return name;

}

Public void setName(String name) {

this.name = name;

}

}

Contract\_employee.java:-

package com.minik;

public class Contract\_Employee extends Employee{

private float  pay\_per\_hour ;

private String contract\_duration;

public float getPay\_per\_hour() {

return  pay\_per\_hour ;}

public void setPay\_per\_hour(float  pay\_per\_hour ) {

this. pay\_per\_hour  =  pay\_per\_hour ;

}

Public String getContract\_duration() {

return contract\_duration;

}

Public void setContract\_duration(String contract\_duration) {

this.contract\_duration = contract\_duration;

}

}

Regular\_employee.java:-

Package com.minik;

Public class Regular\_Employee extends Employee{

Private float salary;

Private int  bonus;

Public float getSalary() {

return salary;

}

Public void setSalary(float salary) {

this.salary = salary;

}

Public int getBonus() {

return  bonus;

}

Public void setBonus(int bonus) {

this. bonus =  bonus;

}

}

StoreData.java:-

 package com.minik;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

 public class StoreData {

 public static void main(String[] args) {

StandardServiceRegistry ssr=newStandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta=new MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory=meta.getSessionFactoryBuilder().build();

Session session=factory.openSession();

Transaction t=session.beginTransaction();

Employee e1=new Employee();

e1.setName("Gaurav Chawla");

Regular\_Employee e2=new Regular\_Employee();

e2.setName("Vivek Kumar");

e2.setSalary(50000);

e2.setBonus(5);

Contract\_Employee e3=new Contract\_Employee();

e3.setName("Arjun Kumar");

e3.setPay\_per\_hour(1000);

e3.setContract\_duration("15 hours");

session.persist(e1);

session.persist(e2);

session.persist(e3);

t.commit();

session.close();

System.out.println("success");

}

}

Employee.hbm.xml:-

<?xml version= '1.0'  encoding='UTF-8' ?>

 <!DOCTYPE hibernate-mapping PUBLIC  "

-//Hibernate/Hibernate Mapping DTD 5.3//EN"

 "http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">

 <hibernate-mapping>

  <class name= "com.minik.Employee" table="emp122" >

 <id name= "id" >

 <generator  class= "increment" >

</generator >

 </id>

  < property name= "name" ></ property>

  <union-subclass name= "com.minik.Regular\_Employee" table="regemp122" >

 < property name= "salary" ></ property>

  < property name= "bonus" ></ property>

  </union-subclass>

  <union-subclass name= "com.minik.Contract\_Employee" table="contemp122" >

 < property name= "pay\_per\_hour" ></ property>

< property name= "contract\_duration" ></ property>

  </union-subclass>

  </class>

  </hibernate-mapping>

Hibernate.cfg.xml:-

<?xml version= '1.0'  encoding='UTF-8' ?>

 <!DOCTYPE hibernate-configuration PUBLIC  "

-//Hibernate/Hibernate Configuration DTD 5.3//EN"

 "http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

 <hibernate-configuration>

<session-factory>

  < property name= "hbm2ddl"

>auto </ property>

< property name= "dialect" >org.hibernate.dialect.Oracle9Dialect</ property>

  < property name= "connection.url" > jdbc:oracle:thin:@localhost:1521:xe</ property>

  < property name= "connection.username" >system</ property>

< property name= "connection.password" >manager </ property>

  < property name= "connection.driver\_class" >oracle.jdbc.driver.OracleDriver </ property>

  <mapping resource= "employee.hbm.xml" />

 </session-factory>

</hibernate-configuration>

### Post Lab Task:

1. From the Bank-Payment Scenario use the concept of table per subclass Hierarchy using Hibernate FrameWork and observe how it works and storing data in database.

**Writing space for the Problem:(For Student’s use only)**

Payment.java:-

package com.minik;

public class Payment{

private int  paymentId;

private double amount;

public int getPaymentId() {

return paymentId;

}

Public void setPaymentId(int paymentId)

{

this. paymentId =  paymentId;

}

Public double getAmount() {

return amount;

}

Public void setAmount(

double amount) {

this.amount = amount;

}

}

CreditCard.java:-

package com.minik;

public class CreditCard extends Payment{

private String CreditCardType;

public String getCreditCardType() {

return CreditCardType;

}

Public void setCreditCardType(String creditCardType) {

CreditCardType = creditCardType;

}

}

Cheque.java:-

package com.minik;

public class Cheque extends Payment{

private String ChequeType;

public String getChequeType() {

return ChequeType;

}

Public void setChequeType(String chequeType) {

ChequeType = chequeType;

}

}

Customer.java:-

package com.minik;

import org.hibernate.\*;

import org.hibernate.cfg.\*;

public class Customer {

public static void main(String[] args) {

 Configuration cfg = new Configuration();

 cfg.configure("hibernate.cfg.xml");

SessionFactory factory = cfg.buildSessionFactory();

 Session session = factory.openSession();

 CreditCard c=new CreditCard();

 c.setPaymentId(10);

 c.setAmount(2500);

c.setCreditCardType("VisaCard");

 Cheque c1=new Cheque();

 c1.setPaymentId(11);

 c1.setAmount(2600);

 c1.setChequeType("ICICI Bank");

 Transactiontx =session.beginTransaction();

 session.save(c);

 session.save(c1);

 tx.commit();

 session.close();

factory.close();

}

}

Payment.hbm.xml:-

<?xml version= "1.0" ?>

<!DOCTYPE hibernate-mapping PUBLIC "

-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name= "str.Payment" table="PAYMENT" >

<id name= "paymentId"  column="pid" />

< property name= "amount" column="amt" />

< joined-subclass name= "str.CreditCard" table="CreditCard" ><key column= "dummy1" />

< property name= "CreditCardType" column="cctype" length="10" />

</ joined-subclass>

< joined-subclass name= "str.Cheque" table="Cheque" >

<key column= "dummy2" />

< property name= "ChequeType" column="cqtype"  length="10" />

</ joined-subclass>

</class>

</hibernate-mapping>

Hibernate.cfg.xml:-

<?xml version= '1.0' encoding='UTF-8' ?>

<!DOCTYPE hibernate-configuration PUBLIC "

-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

< property name= "connection.driver\_class" >oracle.jdbc.driver.OracleDriver </ property>

< property name= "connection.url" > jdbc:oracle:thin:@localhost:1521:XE</ property> < property name= "connection.username" >system</ property>

< property name= "connection.password" >manager </ property>

< property name= "dialect" >org.hibernate.dialect.OracleDialect</ property>

< property name= "show\_sql" >true</ property>

< property name= "hbm2ddl" >auto</ property>

<mapping resource= "Payment.hbm.xml" />

</session-factory>

</hibernate-configuration>

**Writing space for the Problem:(For Student’s use only)**

*(For Evaluator’s use only)*

Evaluator’s Observation

Marks Secured: out of Full Name of the Evaluator:

Signature of the Evaluator Date of Evaluation:

Comment of the Evaluator (if Any)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# SPRING FRAMEWORK #13

**Date of the Session: / / Time of the Session: to**

**Prerequisite:**

* General idea on Spring Framework
* Modules of Spring Framework

## Pre-Lab Task:

* 1. What is a spring?

The Spring Framework is an application framework and inversion of control container for the Java  platform. The framework's core features can be used by any Java application, but there are extensions for  building web applications on top of the Java EE platform

* 1. Name some of the important spring Modules?

Spring Context – for dependency injection.

Spring AOP – for aspect oriented programming.

Spring DAO – for database operations using DAO pattern.

Spring JDBC – for JDBC and DataSource support.

Spring ORM – for ORM tools support such as Hibernate.

* 1. What are the tasks performed by IOC container?

The

IoC container is responsible to instantiate, configure and assemble the objects. The

IoC container gets informations from the XML file and works accordingly. The main

tasks performed by IoC container are: to instantiate the application class.

* 1. How many types of IOC containers are there? Explain them?

a)spring bean factory container

 b)spring application context containe

## In Lab Task:

* + 1. Geetha is creating a website which displays the marks of student from two different java classes. So, define two different java beans and input attributes in Student.java like studentId, studentName and Marks.java which containing clear segregation of marks scored in each subject. Use Spring FrameWork to display all the details of each student to the browser.

Student.java:-

 package com.klu;

 public class Student {

 private int stId; private String stName;

private Mark mark;

 public int getStId() {

return stId;

}

public void setStId(int stId) {

this.stId = stId;

}

 public String getStName() {

return stName;

}

 public void setStName(String stName) {

this.stName = stName;

}

 public Mark getMark() {

return mark;

}

 public void setMark(Mark mark) {

this.mark = mark;

}

}

Mark.java:-

 package com.klu;

public class Mark {

private String maths;

 private String physics;

 private String chemistry;

public String getMaths() {

return maths;

}

 public void setMaths(String maths) {

this.maths = maths;

}

 public String getPhysics() {

return physics;

}

 public void setPhysics(String physics) {

this.physics = physics;

}

 public String getChemistry() {

return chemistry;

}

 public void setChemistry(String chemistry) {

this.chemistry = chemistry;

}

 }

Test.java:-

 package com.klu;

import org.springframework.beans.factory.BeanFactory;

import org.springframework.beans.factory.xml. XmlBeanFactory ;

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.Resource;

 public class Test { public static void main(String[] args) {

Resource resource=new ClassPathResource("applicationcontext.xml");

BeanFactory factory=new XmlBeanFactory (resource) ;

student s=(student)factory.getBean("st");

system.out.println("stId:"+s..getstId());

 system.out.println("stName:"+s.getstName());

system.out.println("stMark:");

 system.out.println("maths:"+s.getmark().getmaths());

 system.out.println("physics:"+e.getmark().getphysics());

 system.out.println("chemistry:"+e.getmark().getchemistry());

}

}

Applicationcontext.xml:-

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

 xmlns:p="http://www.springframework.org/schema/p"

 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd" >

<bean id="mark" class="com.klu.Mark" >

 <property name="maths"  value="50" ></property>

 <property name="physics" value="48" ></property>

 <property name="chemistry" value="45" ></property>

 </bean>

<bean id="st"  class="com.klu.Student" >

 <property name="stId" value="1234567" ></property>

<property name="stName" value="vamsi" ></property>

 <property name="mark"  ref="mark" ></property>

 </bean>

</beans>

* + 1. Mr.Deepak is very keen and interested to overcome Tightcoupling in java so he used Spring Framework with Dependency injection mechanism, he created a class traveler containing an interface vehicle and a method startjourney calling a move method when a journey starts. He creates another two classes bike and car which implements the vehicle class, creates a object for vehicle when a journey starts. To know which vehicle is being used he is calling the method move with vehicle object so that the message will display which is present in implemented classes.

Journey.java:-

Public interface Journey{

Void startJourney()

}

Travel.java :-

 public class Travel implements Journey{

Private Vehicle v;Public void setV(Vehicle v){

this.v=v;

}

Public void startJourney(){

System.out.println(“Journey started…….”);v.move();

}

}

Vehicle.java:-

 public interface Vehicle{

Void move();

}

Car.java:-

Package com.klu;

Public class car

implements Vehicle{

Private String fuelType;

Private int maxSpeed;

Public String getFuelType(){

return fuelType;

}

Public void setFuelType(String fuelType){

this.fuelType=fuelType;

}

Public int getMaxSpeed(){

Return maxSpeed;

}

Public void setMaxSpeed(int maxSpeed){

this.maxSpeed=maxSpeed;

}

Public void move(){

System.out .Println("Fuel type:"+fuelType);

System.out .Println("max speed:"+maxSpeed);

System.out .Println("car started");

}

}

Bus.java:-

Package com.klu;

Public class Bus implements Vehicle{

Private int maxSpeed;

Public int getMaxSpeed(){

Return maxSpeed;

}

Public void setMaxSpeed(int maxSpeed){

this.maxSpeed=maxSpeed;

}

Public void move(){

System.out.Println("max speed:"+maxSpeed);

System.out.Println("Bus started");

}

}

Config1.xml:-

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

<beans xmlns="http://www.springframework.org/schema/beans"

 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

 xmlns:p="http://www.springframework.org/schema/p"

 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd" >

<bean id="car "  class="com.klu.Car" >

 <property name="furlType" value="Diesel" ></property>

 <property name="maxSpeed" value="100" ></property>

 </bean>

<bean id="bus"  class="com.klu.Bus" >

 <property name="maxSpeed" value="80" ></property>

 </bean>

</beans>

Config2.xml:-

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

<beans xmlns="http://www.springframework.org/schema/beans"

 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

 xmlns:p="http://www.springframework.org/schema/p"

 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd" >

<bean id="travel "  class="com.klu.Travel ">

<property name=" Vehicle">

<ref parent="bus"/></property>

</bean>

</beans>

Logic.java:-

Package com.klu;

import org.springframework.beans.factory.BeanFactory;

import org.springframework.beans.factory.xml.XmlBeanFactory;

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.Resource;

public class Logic{

Public static void main(String[] args){

Resource resource=new ClassPathResource("config1.xml");

BeanFactory factory=new XmlBeanFactory(resource);

Resource res=new ClassPathResource("config2.xml");

BeanFactory factory=new XmlBeanFactory(res);

Object o=res.getBean("travel");

Journey j=(Journey)o;

}

}

## Post Lab Task:

1. Ms.Varsha wants to develop an application using Spring FrameWork to display all the details of each employee to the browser. It displays the address of employee from two different java class, so define different java beans and input attributes in Employee.java like empId, empName and Address.java which contains clear address of each person.

Employee.java:-

 package com.klu;

public class Employee {

//yars private int empId;

private String empName;

private Address addr;

 public int getEmpId() {

return empId;

}

 public void setEmpId(int empId) {

this.empId = empId;

}

 public String getEmpName() {

return empName;

}

 public void setEmpName(String empName) {

this.empName = empName;

}

 public Address getAddr() {

return addr;

}

 public void setAddr(Address addr) {

this.addr = addr;

}

}

Address.java:-

 package com.klu;

 public class Address {

 private String street;

 private String city;

 private String state;

 public String getStreet() {

return street;

}

 public void setStreet(String street) {

this.street = street;

}

 public String getCity() {

return city;

}

 public void setCity(String city) {

this.city = city;

}

 public String getState() {

return state;

}

 public void setState(String state) {

this.state = state;

}

 }

Test.java:-

 package com.klu;

import org.springframework.beans.factory.BeanFactory;

import org.springframework.beans.factory.xml.XmlBeanFactory;

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.Resource;

 public class Test {

 public static void main(String[] args) {

Resource resource=new ClassPathResource("applicationcontext.xml");

BeanFactory factory=new XmlBeanFactory(resource);

employee e=(employee)factory.getBean("emp"); system.out.println("empId:"+e.getempId());

 system.out.println("empName:"+e.getempName());

 system.out.println("empAddress:");

 system.out.println("street:"+e.getaddr().getstreet());

 system.out.println("street:"+e.getaddr().getstreet());

 system.out.println("city:"+e.getaddr().getcity());

system.out.println("state:"+e.getaddr().getstate());

}

}

Applicationcontext.java:-

 package com.klu;

import org.springframework.beans.factory.BeanFactory;

import org.springframework.beans.factory.xml.XmlBeanFactory;

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.Resource;

 public class Test {

public static void main(String[] args) {

Resource resource=new ClassPathResource("applicationcontext.xml");

BeanFactory factory=new XmlBeanFactory(resource);

employee e=(employee)factory.getBean("emp");

system.out.println("empId:"+e.getempId());

 system.out.println("empName:"+e.getempName());

 system.out.println("empAddress:");

 system.out.println("street:"+e.getaddr().getstreet());

 system.out.println("street:"+e.getaddr().getstreet());

 system.out.println("city:"+e.getaddr().getcity());

 system.out.println("state:"+e.getaddr().getstate());

}

}

**Writing space for the Problem:(For Student’s use only)**

**Writing space for the Problem:(For Student’s use only)**

**Writing space for the Problem:(For Student’s use only)**

*(For Evaluator’s use only)*

Comment of the Evaluator (if Any)

Signature of the Evaluator Date of Evaluation:

Evaluator’s Observation

Marks Secured: out of

Full Name of the Evaluator:

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# HIBERNATE-SPRING INTEGRATION #14

**Date of the Session: / / Time of the Session: to**

**Prerequisite:**

* **Basic knowledge on hibernate and spring integration**

## Pre-Lab Task:

1. Explain the concept of Integration of Spring-Hibernate Framework.

In hibernate framework , we provide all the database information hibernate.cfg.xml file. But if we are going to integrate the hibernate application with spring, we don’t need to create the hibernate.cfg.xmlfile. We can provide all the information in the applicationContext.xml file

1. List out few advantages of Integration.

Services layer

 The most important benefit is the Spring framework itself. Because of the Spring integration, applications can leverage all the features of spring framework. For example POJO style serve interface, IoC, Aspects, Remoting etc….

HibernateTemplate

Spring provides template for managing sessions, transactions across the application. Without the hibernate template, applications need to manage these on their own

HibernateDaoSupport

Spring provides a class called HibernateDaoSupport. DAO Implementation class implements this class to get all the convenience methods that HibernateTemplate provides. If applications are not using HibernateTemplate, then it’s good to take advantage of HibernateDaoSupport.

DAOException Translation

Hibernate 3 throws Runtime exceptions unlike checked exceptions thrown in the previoys releases. Spring can translate these exceptions into SprinDAO excpetions and map them into Spring DAO exception hierarchy. But you have to use one of the following to get this 1.Use HibernateTemplate2.@Repository

MVC Integration

If you are using Hibernate in ypu web application, which is using Spring MVC, then you can use the OpenSessionInViewFilter or OpenSessionInViewInterceptor and not create sessions per thread, instead

Create a session per request.

**Writing space for the Problem:(For Student’s use only)**

1. List out Methods of Hibernate-template class and write description about it.

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
| 1) | void persist(Object entity) | persists the given object. |
| 2) | Serializable save(Object entity) | persists the given object and returns id. |
| 3) | void saveOrUpdate(Object entity) | persists or updates the given object. If id is found, it updates the record otherwise saves the record. |
| 4) | void update(Object entity) | updates the given object. |
| 5) | void delete(Object entity) | deletes the given object on the basis of id. |
| 6) | Object get(Class entityClass, Serializable id) | returns the persistent object on the basis of given id. |
| 7) | Object load(Class entityClass, Serializable id) | returns the persistent object on the basis of given id. |
| 8) | List loadAll(Class entityClass) | returns the all the persistent objects. |

## In Lab Task:

1. The Navodaya Johar school is wanting to direct a get-together of their 2014-2015 batch students. To design this occasion and oversee, principal appointed a student from a similar batch. To do this the student needed to make hibernate application with spring application. He initially made a table in database to store the subtleties of his companions who are going to the gathering. He made the table with the name Reunion with properties ID No, Name, Contact Number, Amount paid, and Status. ID.NO being the primary key has the most extreme size of 4 digits and Status speaks to how much sum has been paid i.e; completely paid, or partially paid, or not paid. Note that the sum should be paid to go to the gathering is 2000 rupees. Now he needs to make a java project with springs and hibernate integration. Help him in Creating three unique classes for inserting subtleties in database. First make a class for inserting the details of the students who are going to the get-together gathering. Presently make another class for retrieving the amount paid by the students.

POJO class(Reunion.java):-

package lab14in1;

public class Reunion {

private int sid;

private String sname;

private double contact;

private int amount;

private String status;

public int getSid() {

return sid;

}

Public void setSid(int sid) {

this.sid = sid;

}

Public String getSname() {

return sname;

}

Public void setSname(String sname) {

this.sname = sname;

}

Public double getContact() {

return contact;

}

Public void setContact(double contact) {

this.contact = contact;

}

Public int getAmount() {

return amount;

}

Public void setAmount(int amount) {

this.amount = amount;

}

public String getStatus() {

return status;

}

Public void setStatus(String status) {

this.status = status;

}

}

Mapping file(reunion.hbm.xml):-

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

<!DOCTYPE hibernate-mapping PUBLIC "

-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="lab14in1.Reunion" table="reunion" >

<id name="sid" >

<generator class="assigned" ></generator>

</id>

<property name="sname" ></property>

<property name="contact" ></property>

<property name="amount" ></property>

<property name="status" ></property>

</class>

</hibernate-mapping>

DAO class(ReunionDao.java):

package lab14in1;

import org.springframework.orm.hibernate3.HibernateTemplate;

import java.util.\*;

public class ReunionDao {HibernateTemplate template;

public void

 setTemplate(HibernateTemplate template) {

this.template = template;

}

//method to save employee

Public void saveReunion(Reunion e){

template.save(e);

}

//method to update employee

Public void updateReunion(Reunion e){

template.update(e);

}

//method to delete employee

Public void deleteReunion(Reunion e){

template.delete(e);

}

//method to return one employee of given id

Public Reunion getById(int id){

Reunion e=(Reunion)template.get(Reunion.class,id);

Return e;

}

//method to return all employees

public List<Reunion> getReunions(){

List<Reunion> list=new ArrayList<Reunion>();

list=template.loadAll(Reunion.class);

return list;

}

}

applicationContext.xml :-

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

<beans xmlns="http://www.springframework.org/schema/beans"

 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

 xmlns:p="http://www.springframework.org/schema/p"

 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd" >

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" >

<property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" ></property>

 <property name="url" value="jdbc:oracle:thin:@localhost:1522:xe" ></property>

<property name="username"  value="system" ></property>

<property name="password" value="system" ></property>

</bean>

<bean id="mysessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean" > <property name="dataSource" ref="dataSource" ></property>

<property name="mappingResources" >

<list>

<value>reunion.hbm.xml</value>

</list>

</property>

<property name="hibernateProperties" >

<props> <prop key="hibernate.dialect" >org.hibernate.dialect.Oracle9Dialect</prop>

<prop key="hibernate.hbm2ddl.auto" >update</prop>

<prop key="hibernate.show\_sql" >true</prop>

</props>

</property>

</bean>

<bean id="template" class="org.springframework.orm.hibernate3.HibernateTemplate" >

<property name="sessionFactory" ref="mysessionFactory" ></property>

</bean>

<bean id="d" class="lab14in1.ReunionDao" >

<property name="template"  ref="template" ></property>

</bean>

</beans>

File for inserting the students details(Insert.java) :-

package lab14in1;

import java.util.\*;

import org.springframework.beans.factory.BeanFactory;

import org.springframework.beans.factory.xml. XmlBeanFactory ;

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.Resource;

public class Insert {

public static void main(String[] args) {

Resource r =new ClassPathResource("applicationContext.xml");

BeanFactory factory=newXmlBeanFactory (r) ;ReunionDao dao = (ReunionDao)factory.getBean("d");

Scanner sc = new Scanner(System.in);

Int id;

String name;

double phone;

int amt;

String status;

Boolean continues=true;

while(continues) {

Reunion e =new Reunion();

System.out .println("Enter the id number :");

id=sc.nextInt();

System.out .println("Enter the name :");

name=sc.next();

System.out .println("Enter the contact number :");

 phone=sc.nextDouble();

System.out .println("Enter the amount to be paid :");amt=sc.nextInt();

System.out .println("Enter the status(fullypaid/partiallypaid/notpaid):");

status=sc.next();

e.setSid(id);

e.setSname(name);

e.setContact(phone);

e.setAmount(amt);

e.setStatus(status);

dao.saveReunion(e);

System.out .println("Do you want to insert(yes/no) :");

String proceed=sc.next();

if (proceed.equals("yes")) {

continues=true;

}

  else{

continues=false;

}

}

}

}

File for retrieving the amount paid by the student(Retrieve.java) :-

package lab14in1;

import java.util.Scanner;

import org.springframework.beans.factory.BeanFactory;

import org.springframework.beans.factory.xml. XmlBeanFactory ;

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.Resource;

public class Retrieve {

public static void main(String[]args) {

Resource r =new

 ClassPathResource("applicationContext.xml");

BeanFactory factory=new XmlBeanFactory (r) ;ReunionDao dao = (ReunionDao)factory.getBean("d");

Scanner sc = new Scanner(System.in);

Reunion e = new Reunion();

System.out .println("Enter the id of the student whose details are required");

int rid= sc.nextInt();

Reunion h=dao.getById(rid);

System.out .println("the amountpaid by the student is");

System.out .println("Amount paid "+h.getAmount());

}

}

**Writing space for the Problem:(For Student’s use only)**

**Writing space for the Problem:(For Student’s use only)**

**Writing space for the Problem:(For Student’s use only)**

## Post Lab Task:

1. Now to the extension to the last question create a class file in the same java project for updating the database If the student want to pay the amount then update the amount in data base and print the total amount he paid till then and if he had paid the total amount the change the status to fully paid. If not then show the amount that need to be paid by him. Also create a class file for deleting the details of students who cancelled their plan of going go get-together as a result of some issues.

**Writing space for the Problem:(For Student’s use only)**

For updating the amount of students(Update.java):-

package lab14in1;

import java.util.Scanner;

import org.springframework.beans.factory.BeanFactory;

import org.springframework.beans.factory.xml. XmlBeanFactory ;

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.Resource;

public class Update {

public static void main(String[] args) {

Resource r =new ClassPathResource("applicationContext.xml");

BeanFactory factory=new XmlBeanFactory (r) ;ReunionDao dao = (ReunionDao)factory.getBean("d");

Scanner sc = new Scanner(System.in);

Boolean continues=true;

while(continues){

Reunion e =new Reunion();

System.out.println("Any updates required(yes/no)");

String change=sc.next();

if (change.equals("yes")) {

System.out.println("Enter the id number of student");

int uid=sc.nextInt();

Reunion h=dao.getById(uid);

int cost=h.getAmount();

System.out.println("the amount paid by the student is :"+cost);

 System.out.println("Do you want to pay(yes/no)"); String pay=sc.next();

if (pay.equals("yes"))

{

 System.out.println("Enter the amount paid now");

Int now=sc.nextInt();

 cost=cost+now;

h.setAmount(cost);

 dao.updateReunion(h);

if (cost==2000) {

 h.setStatus("Fullypaid");

 dao.updateReunion(h);

 }

 }

else{

 System.out.println("the amount student has to pay is "+(2000-cost));

 }

}

else {

 System.out.println("Updates are done");

 continues=false;

 }

 }

 }

}

For deleting the dtails of students who wants to drop from attending the gathering(Delete.java) : -

package lab14in1;

import java.util.Scanner;

import org.springframework.beans.factory.BeanFactory;

import org.springframework.beans.factory.xml. XmlBeanFactory ;

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.Resource;

public class Delete {

public static void main(String[]args) {

Resource r =new ClassPathResource("applicationContext.xml");

BeanFactory factory=new XmlBeanFactory (r) ;ReunionDao dao = (ReunionDao)factory.getBean("d");

Scanner sc = new Scanner(System.in);

Boolean continues=true;

while(continues) {

System.out .println("enter the id of student who wants to drop from attending the gathering");

Int did=sc.nextInt();

Reunion w= dao.getById(did);

dao.deleteReunion(w);

System.out .println("the details of the student are deleted");

System.out .println("Do you want to delete some more(yes/no)");

String del=sc.next();

if (del.equals("yes")) {

continues=true;

}

else{

continues=false;

}

}

}

}

*(For Evaluator’s use only)*

Signature of the Evaluator Date of Evaluation:

Evaluator’s Observation

Marks Secured: out of

Full Name of the Evaluator:

Comment of the Evaluator (if Any)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SUBJECT CODE: 17CS3116**

**ENTERPRISE PROGRAMMING WORKBOOK**

# SPRING-DAO #15

**Date of the Session: / / Time of the Session: \_to**

**Prerequisite:**

* Basic idea on Spring -DAO

## Pre-Lab Task:

1. What is DAO?

It is a object/interface, which is used to access data from database of data storage.

1. Why do we use DAO?

It abstracts the retrieval of data from a data resource such as a database. The concept is to

"separate a data resource's client interface from its data access mechanism."

1. What are the problems faced if data is accessed directly without objects?

The problem with accessing data directly is that the source of the data can change. Consider, for example, that your application is deployed in an environment that accesses anOracle database. Then it is subsequently deployed to an environment that uses Microsoft SQL Server. If your application uses stored procedures and database-specific code you can handle it by

Rewrite your application to use SQL Server instead of Oracle (or add conditional code to handle the differences), or

Create a layer in between your application logic and the data access.

1. write a short note on the participants in DAO pattern.

Data Access Object Pattern or DAO pattern is used to separate low level data accessing API or operations from high level business services. Following are the participants in DataAccess Object Pattern.

Data Access Object Interface

This interface defines the standard operations to be performed on a model object(s).

Data Access Object concrete class

This class implements above interface. This class is responsible to get data from a data source which can be database / xml or any other storage mechanism.

Model Object or Value Object

This object is simple POJO containing get/set methods to store data retrieved using DAO class.

## In Lab Task:

* 1. Revanth is a student who has a great startup plan but he wants students to work with him. So, students who are interested in startup can contact him. Revanth decided to conduct an interview for the interns and store the details who appeared (Name, ID, Mail, skills of every student) in the “student” table (Name varchar2(30), ID number, Mail varchar2(30), skills varchar2(100)). He Inserts all the details of people during interview and deletes the details people who are below par at the end of the day. So, by using “Data Access object” pattern create a student object which act as a model and create a concrete class and implement Data Access Object Interface and use a Democlass to access the insert, delete methods.

**Writing space for the Problem:(For Student’s use only)**

public class Student {

private String name;

private int rollNo;

 Student(String name, int rollNo){

this.name = name;

 this.rollNo = rollNo;

 }

 public String getName() {

return name;

 }

 public void setName(String name)

{

 this.name = name;

 }

 public int getRollNo() {

return rollNo;

 }

 public void setRollNo(int rollNo) {

this.rollNo = rollNo;

 }

}

StudentDao.java: -

import java.util.List;

 public interface StudentDao {

public List<Student> getAllStudents();

public Student getStudent(int rollNo);

 public void deleteStudent(Student student);

}

StudentDaoImpl.java:

import java.util.ArrayList;

import java.util.List;

 public class StudentDaoImpl implements StudentDao {

 //list is working as a database List<Student> students;

 public StudentDaoImpl(){

students = new ArrayList<Student>();

 Student student1 = new Student("Robert",0);

 Student student2 = newStudent("John",1);

students.add(student1);

 students.add(student2);

 }

 @Override public void deleteStudent(Student student) {

students.remove(student.getRollNo());

System.out.println("Student: Roll No " + student.getRollNo() + ", deleted from database");

 }

 //retrive list of students from the database

@Override public List<Student> getAllStudents() {

 return students;

 }

 @Override public Student getStudent(int rollNo) {

 return students.get(rollNo);

}

}

DaoPatternDemo.java:

 public class DaoPatternDemo {

 public static void main(String[] args) {

 StudentDao studentDao = new StudentDaoImpl();

 //print all students for (Student student : studentDao.getAllStudents()) {

 System.out.println("Student: [RollNo : " + student.getRollNo() + ", Name : " + student.getName() + ]");

}

//get the student studentDao.getStudent(0);

 System.out.println("Student: [RollNo : " + student.getRollNo() + ", Name : " + student.getName() + "]");

}

}

## Post Lab Task:

1. To continue with the previous problem, later some students approached Revanth to add some changes to their details and he also wants to display the people who are in list at the end of every day. So Revanth decided to add update and display methods to the DAO pattern.

**Writing space for the Problem:(For Student’s use only)**

Student.java :-

 public class Student {

 private String name;

private int rollNo;

 Student(String name, int rollNo){

 this.name = name;

this.rollNo = rollNo;

 }

 public String getName() {

return name;

}

 public void setName(String name) {

 this.name = name;

}

 public int getRollNo() {

 return rollNo;

 }

 public void setRollNo(int rollNo) {

 this.rollNo = rollNo;

 }

}

StudentDao.java: -

import java.util.List;

 public interface StudentDao {

 public List<Student> getAllStudents();

 public Student getStudent(int rollNo);

 public void updateStudent(Student student);

 public void deleteStudent(Student student);}

StudentDaoImpl.java: -

import java.util.ArrayList;

import java.util.List;

 public class StudentDaoImpl implements StudentDao {

//list is working as a database List<Student> students;

 public StudentDaoImpl(){

students = new ArrayList<Student>();

Student student1 = new Student("Robert",0);

 Student student2 = new Student("John",1);

 students.add(student1);

students.add(student2);

 }

 @Override public void deleteStudent(Student student) {

students.remove(student.getRollNo());

 System.out.println("Student: Roll No " + student.getRollNo() + ", deleted from database");

 }

 //retrive list of students from the database

 @Override public List<Student> getAllStudents() {

return students;

 }

 @Override public Student getStudent(int rollNo) {

 return students.get(rollNo);

 }

 @Override public void updateStudent(Student student) {

 students.get(student.getRollNo()).setName(student.getName());

 System.out.println("Student: Roll No " + student.getRollNo() + ", updated in the database");

}

}

DaoPatternDemo.java: -

 public class DaoPatternDemo {

 public static void main(String[] args) {

 StudentDao studentDao = new StudentDaoImpl();

//print all students for (Student student : studentDao.getAllStudents()) {

 System.out.println("Student: [RollNo : " + student.getRollNo() + ", Name : " + student.getName()+” ]");

 }

 //update student Student student=studentDao.getAllStudents().get(0); student.setName("Michael");

 studentDao.updateStudent(student);

//get the student studentDao.getStudent(0);

 System.out.println("Student: [RollNo : " + student.getRollNo() + ", Name : " + student.getName()+”]");

 }

}

**Writing space for the Problem:(For Student’s use only)**

*(For Evaluator’s use only)*

Evaluator’s Observation

Marks Secured: out of Full Name of the Evaluator:

Signature of the Evaluator Date of Evaluation:

Comment of the Evaluator (if Any)