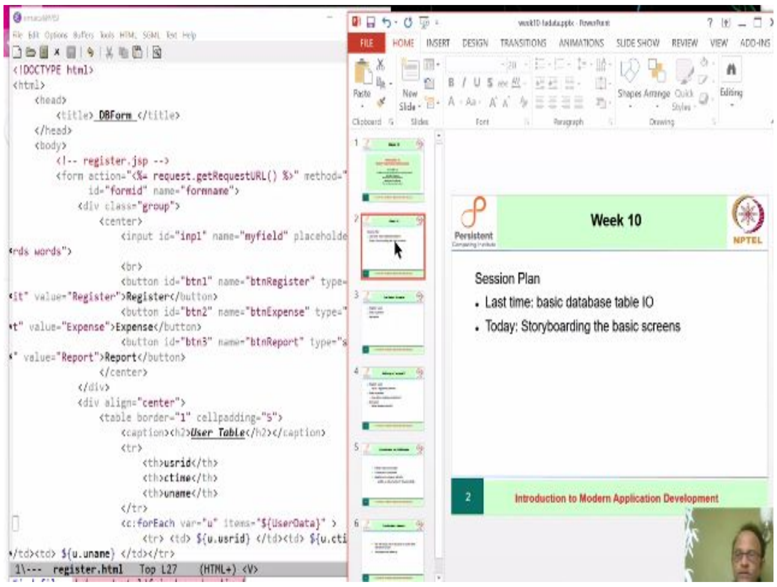
Introduction to Modern Application Development

Prof. Aamod Sane
FLAME University and Persistent Computing Institute
Indian Institute of Technology – Madras

Lecture – 29
Week 10 – Part 1

(Refer Slide Time: 00:30)



```
<!DOCTYPE html>
<html>
    <head>
        <title> DBForm </title>
    </head>
    <body>
        <!-- register.jsp -->
        <form action="<%= request.getRequestURL() %>" method="
              id="formid" name="formname">
            <div class="group">
                <center>
                    <input id="inp1" name="myfield" placeholde
rds words">
                    <br>
                    <button id="btn1" name="btnRegister" type=
it" value="Register">Register</button>
                    <button id="btn2" name="btnExpense" type="
t" value="Expense">Expense</button>
                    <button id="btn3" name="btnReport" type="s
" value="Report">Report</button>
                </center>
            </div>
            <div align="center">
                <table border="1" cellpadding="5">
                    <caption><h2>User Table</h2></caption>
                    <tr>
                        <th>usrid</th>
                        <th>ctime</th>
                        <th>uname</th>
                    </tr>
                    <c:forEach var="u" items="${UserData}" >
                        <tr> <td> ${u.usrid} </td><td> ${u.cti
/td><td> ${u.uname} </td></tr>
1\---  register.html   Top L27    (HTML+) <V>
```

Week 10

Session Plan
- Last time: basic database table IO
- Today: Storyboarding the basic screens

2 Introduction to Modern Application Development

Last session: Basic database table IO and user making calls to the database look at what data you get from it, how the tables inside the database could be structured and so on.

Demo:

<u>Current scenario:</u> We have started as usual this Tomcat manager, launched dbjsp (What dbjsp does is it first shows up under the URL dbjsp as we have loaded it as /dbjsp in the manager) and no matter what button you press is to simply print the user table.

Code Organisation: In dbjsp directory, there are jsp, src, web-inf, meta inf. src contains the servlets that we write such as dbjsp.java and the jsp contains the two jsp's that we care about which includes dbjsp.jsp usertable.jsp.

We have a servlet code which we wrote, we have the two jsp's and we have combined these together into an application. So first of all you have the servlet, here you have jsp and the first visit which is to say the visit that we make when we first just call dbjsp itself which shows us

this form, is going to a jsp page. We know that because we configure that on the web. xml file.
So web.xml contains how the URLs are mapped to the various implementations within the system.

Because this URL pattern maps to this servlet name and this servlet name match to this particular internal component. Now, the minute you visit a jsp, Tomcat will compile it into java and load it and then execute it.

What does that db.jsp will look like?

It is just a page which says what should happen when one of the buttons is pressed. So we have the post method and we have the form action. Now when we press this, we visit the URL called dbj, right. As we can see here as well when you press it, the result is the URL that ends in dbj.

How do we know what that URL does?

This is because of web.xml. So we take a look at this servlet again and it says per URL called /dbj maps to a servlet called / dbj which in turn maps to the class that you wrote for the jsp.

The form calls dbjsp/dbj, the URL maps to dbj, the servlet which maps to nptel.dbjsp the class and in turn it is the dbjsp class which does the actual functionality that we wanted which is use the context to connect to the database, run this statement, get a request dispatcher and then use this to run the jsp that we want which we have specified in here in the request dispatcher. After all this is done, the user table jsp gets called.

It gets the result of user data which we have put here in the set attribute and jsp is now what they do is so user table dot jsp makes these kinds of references and last time we saw how we fixed a bug in this format. So that is the reminder of what the basic structure is. Now we are going to elaborate this basic structure into something where for each button there is a different behavior and that behavior is whatever it is that we decide to do for our overall application.

GOAL:

We know that in our initial storyboarding what we have is we have to have some way to register users, some way to enter expenses and some way to get a report of what just happened. Long time ago when we were doing the GUI, we looked at this kind of a design where we had an input field, three buttons and some output field.

1. Replicate the screen design in html.
2. With each of the buttons in html, there will be a different behavior.
3. Whenever that behavior happens, we want our app to return the appropriate results to us.

It is actually simpler to get a nice UI in the web as compared to getting a nice UI in java because a layout is much easier to understand in html than it is in the java libraries, although some of the newer libraries like java FX and all have attempted to fix them.

Write a html file called register.html which has a table with an input field which is centered and buttons as before.

<u>Constructing register.html</u>

In html we can create tables with rows and columns. So we have a table row and another table row with a bunch of buttons. The novelty is that we are going to allow the topsail to span 3 columns. We add colspan = 3 and align = center. Similarly table height, width, cell padding, etc can be provided.

<u>Cascading Style Sheets : Designing front end of web pages systematically</u>

CSS provides a uniform way of adjusting the sizes using the concept of *style*. Style adjustment can be done explicitly by using CSS.

Create a new element and add an attribute. While adding attribute style, for example when ‘h’ is typed shows all different styles available starting with ‘h’, upon picking height followed by colon, it gives a completion possibility.

Different ways to specify measurements:

1. **<u>AUTO</u>**

   The default answer is **auto** in which case the choice is made suitably, but we consider in general the options available to say make the height of the input field to be twice the size of the font.

2. <u>**Size measurement using em**</u>:We use a size measurement per em, so **1em is the width of a small letter m in the font that is getting used**. One example where em becomes more convenient is to make the input field twice the size of the font, we need to just set the height to 2em.
3. <u>**Percentage:**</u> Suppose we want the width to be a certain percentage of the available space, say 80% (here the available space is the entire cell and within it about 80% of the space is used by the input field and then because the alignment is center it is perfectly centered).

Styling buttons:

Usually a button text can completely occupy the box of the button. We can create some spacing around the borders. There are standard style attributes called **padding, border and margin** to achieve this.

Padding is the area around the content. **Border** is an area around padding away from content. **Margin** is the area outside the border.

CSS is like a collection of style related elements as we saw, and these styles are added to elements and the elements themselves do not contain its style information.

We can visualise all the above in any browser's developer tool's right window.

PRO TIP:

We have seen the standard attributes that are available for every element in the browser and what this style of writing is called CSS and usually instead of writing it as a part of the page, it is normal to get it from an external style file.

Advantage of percentage style for measurements: Upon adjusting the browser window size, the element adjusts itself to maintain the percentage of the total available size that is specified.

Disadvantage of percentage style for measurements: However if the window is too short, it remains in the middle, but if it becomes tall then roughly it maintains the distance from the top.

What is the ideal expectation when browser window size is modified?

Viewport is the size of the browser window. So we want the size of the input field to be proportionate to the viewport so that when the viewport changes, the size of the field is reduced or enlarged proportionately.

We will say that this should be 20% of the viewport, so **VH here means viewport height**.

Also notice that when size is too small, a scrollbar appears. However when we didn't use viewport with percentage, the field stayed in center and did not reduce proportionately.

<u>Responsive design mode: Even More Versatile Option</u>
Nowadays, we see sites not only on browsers with large screens but also on mobile devices and such and for that these tools provide something called a **responsive design mode**. The screen starts looking like the screen of a browser. It requires much more detailed knowledge of CSS to be able to do responsive design.

Professional website designers will typically not use tables, instead they will rely on CSS, but if we are not there yet so tables are the easiest to talk about and use in our examples and this is the basic layout that we will use from now on to build the screens of our app.