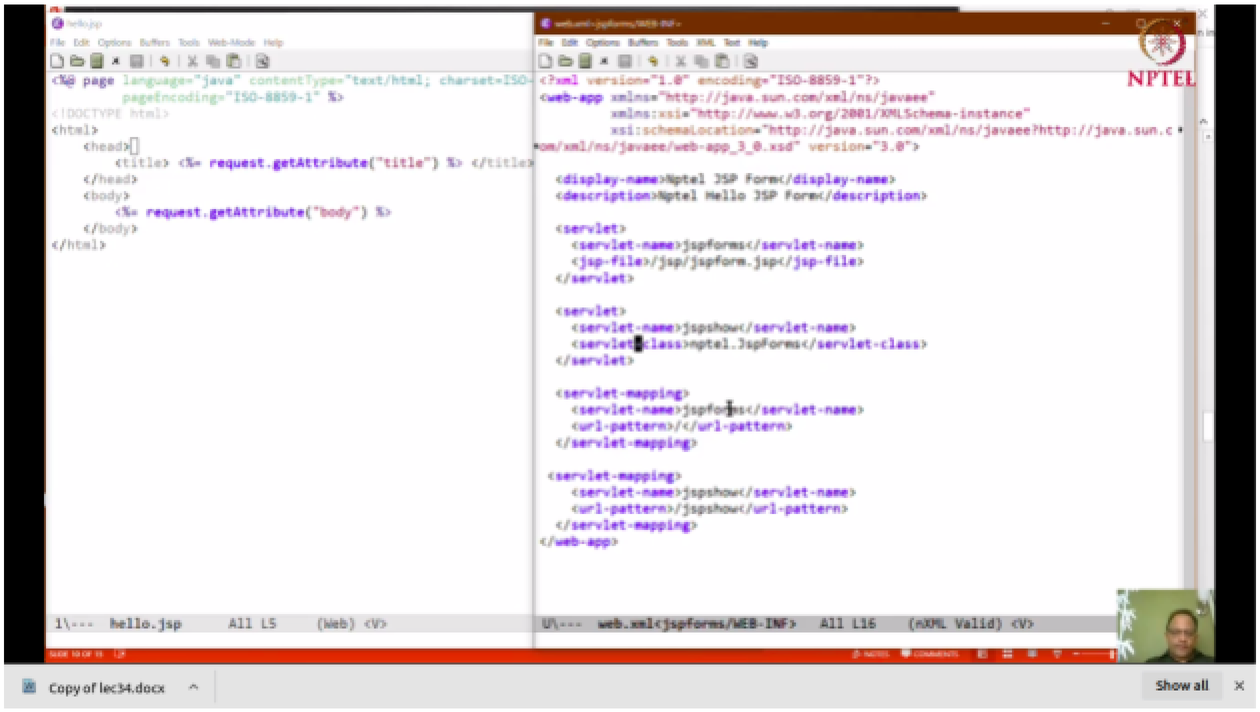
Modern Application Development
Mr. Aamod Sane
FLAME University and Persistent Computing Institute
Indian Institute of Technology - Madras

Lecture – 26
Introduction to Modern Application Development

(Refer Slide Time: 00:11)



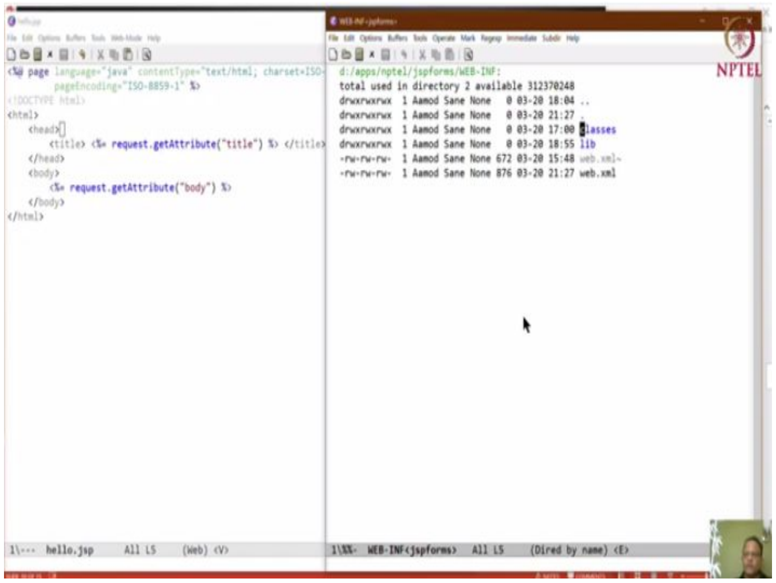
```
<%@ page language="java" contentType="text/html; charset=ISO-
         pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
    <head>
        <title> <%= request.getAttribute("title") %> </title>
    </head>
    <body>
        <%= request.getAttribute("body") %>
    </body>
</html>
```

```
1\---  hello.jsp      All L5     (Web) <V>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee?http://java.sun.c
om/xml/ns/javaee/web-app_3_0.xsd" version="3.0">

  <display-name>Nptel JSP Form</display-name>
  <description>Nptel Hello JSP Form</description>

  <servlet>
    <servlet-name>jspforms</servlet-name>
    <jsp-file>/jsp/jspform.jsp</jsp-file>
  </servlet>

  <servlet>
    <servlet-name>jspshow</servlet-name>
    <servlet-class>nptel.JspForms</servlet-class>
  </servlet>

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

 <servlet-mapping>
    <servlet-name>jspshow</servlet-name>
    <url-pattern>/jspshow</url-pattern>
  </servlet-mapping>
</web-app>
```

```
U\---  web.xml<jspforms/WEB-INF>    All L16     (nXML Valid) <V>
```

(Refer Slide Time: 00:17)



```
<%@ page language="java" contentType="text/html; charset=ISO-
        pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
    <head>
        <title> <%= request.getAttribute("title") %> </title>
    </head>
    <body>
        <%= request.getAttribute("body") %>
    </body>
</html>
```

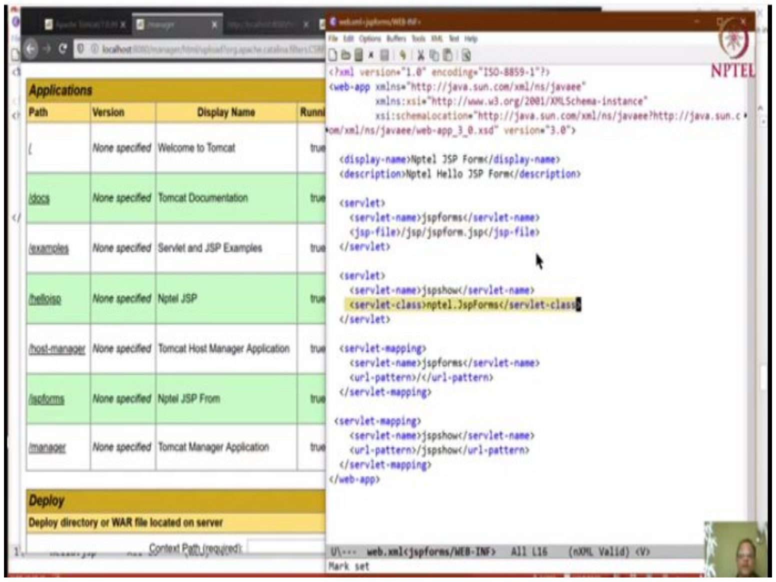
```
d:/apps/nptel/jspforms/WEB-INF:
total used in directory 2 available 312370248
drwxrwxrwx  1 Aamod Sane None   0 03-20 18:04 ..
drwxrwxrwx  1 Aamod Sane None   0 03-20 21:27 .
drwxrwxrwx  1 Aamod Sane None   0 03-20 17:00 classes
drwxrwxrwx  1 Aamod Sane None   0 03-20 18:55 lib
-rw-rw-rw-  1 Aamod Sane None 672 03-20 15:48 web.xml~
-rw-rw-rw-  1 Aamod Sane None 876 03-20 21:27 web.xml
```

```
1\---  hello.jsp     All L5     (Web) <V>
1\%%-  WEB-INF<jspforms>   All L5    (Dired by name) <E>
```

JSP which has a form:

So, here what we are going to do is something slightly different, we will load jspforms.war

(Refer Slide Time: 00:56)



Applications

Path	Version	Display Name	Runni
/	None specified	Welcome to Tomcat	true
/docs	None specified	Tomcat Documentation	true
/examples	None specified	Servlet and JSP Examples	true
/hellojsp	None specified	Nptel JSP	true
/host-manager	None specified	Tomcat Host Manager Application	true
/jspforms	None specified	Nptel JSP From	true
/manager	None specified	Tomcat Manager Application	true

Deploy

Deploy directory or WAR file located on server

Context Path (required):

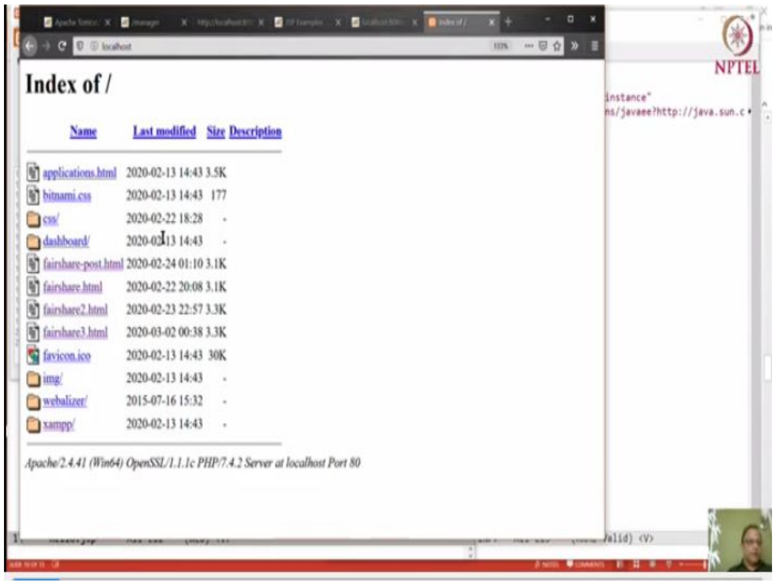
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee?http://java.sun.c
om/xml/ns/javaee/web-app_3_0.xsd" version="3.0">

  <display-name>Nptel JSP Form</display-name>
  <description>Nptel Hello JSP Form</description>

  <servlet>
    <servlet-name>jspforms</servlet-name>
    <jsp-file>/jsp/jspform.jsp</jsp-file>
  </servlet>

  <servlet>
    <servlet-name>jspshow</servlet-name>
    <servlet-class>nptel.JspForms</servlet-class>
  </servlet>

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

 <servlet-mapping>
    <servlet-name>jspshow</servlet-name>
    <url-pattern>/jspshow</url-pattern>
  </servlet-mapping>
</web-app>
```

```
U\---  web.xml<jspforms/WEB-INF>   All L16    (nXML Valid) <V>
Mark set
```

So, when we load jspforms as a war, context path that is taken is actually jspforms itself. So, what should happen when we click on jspforms, so far for example, in web.xml of hello jsp, this URL pattern, the first one you visit corresponds to the servlet, which corresponds to this servlet class.

So, jspforms/ is the opening URL and that will go to a servlet but as we have seen it is actually, a servlet underneath even though, it looks like a standard jsp to us, this is identified by using the jsp file tag rather than the servlet class tag.

(Refer Slide Time: 02:32)

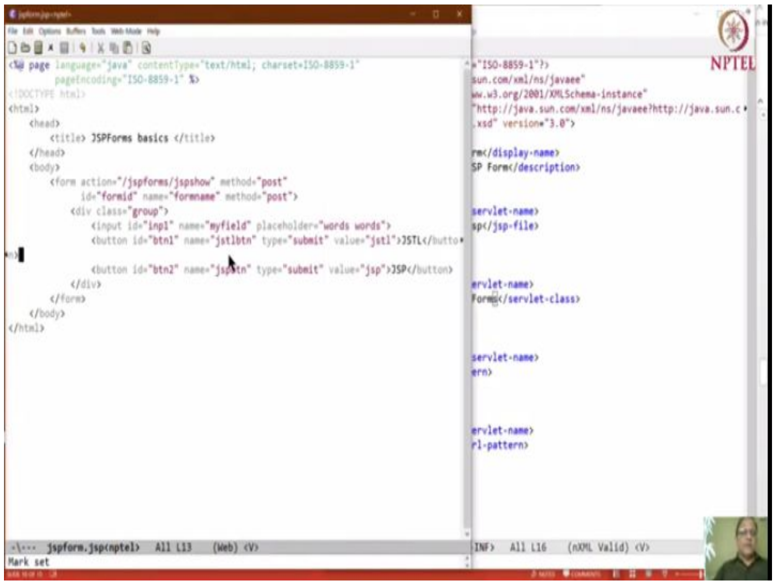


Index of /

Name	Last modified	Size	Description
applications.html	2020-02-13 14:43	3.5K	
bitnami.css	2020-02-13 14:43	177	
css/	2020-02-22 18:28	-	
dashboard/	2020-02-13 14:43	-	
fairshare-post.html	2020-02-24 01:10	3.1K	
fairshare.html	2020-02-22 20:08	3.1K	
fairshare2.html	2020-02-23 22:57	3.3K	
fairshare3.html	2020-03-02 00:38	3.3K	
favicon.ico	2020-02-13 14:43	30K	
img/	2020-02-13 14:43	-	
webalizer/	2015-07-16 15:32	-	
xampp/	2020-02-13 14:43	-	

Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.4.2 Server at localhost Port 80

(Refer Slide Time: 03:38)



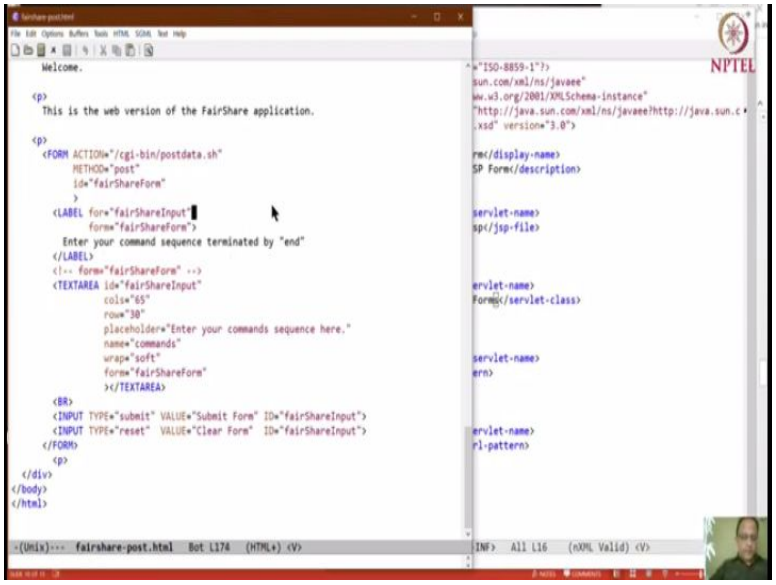
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
  <head>
    <title> JSPForms basics </title>
  </head>
  <body>
    <form action="/jspforms/jspshow" method="post"
        id="formid" name="formname" method="post">
      <div class="group">
        <input id="inp1" name="myfield" placeholder="words words">
        <button id="btn1" name="jstlbtn" type="submit" value="jstl">JSTL</butto
n>

        <button id="btn2" name="jspbtn" type="submit" value="jsp">JSP</button>
      </div>
    </form>
  </body>
</html>
```

```
-\---  jspform.jsp<nptel>   All L13    (Web) <V>
Mark set
```

```
="ISO-8859-1"?>
sun.com/xml/ns/javaee"
ww.w3.org/2001/XMLSchema-instance"
"http://java.sun.com/xml/ns/javaee?http://java.sun.c
.xsd" version="3.0">

rm</display-name>
SP Form</description>

servlet-name>
sp</jsp-file>

ervlet-name>
Forms</servlet-class>

servlet-name>
ern>

ervlet-name>
rl-pattern>

INF>   All L16    (nXML Valid) <V>
```

In principle **jsp which is the general purpose template language**, we can also use other languages. So we start with a form and it is the form that then goes to the second page, jsp show and it is jsp which takes us to the jspforms class. We first visit jsp page which is kind of like the html page.

(Refer Slide Time: 05:45)

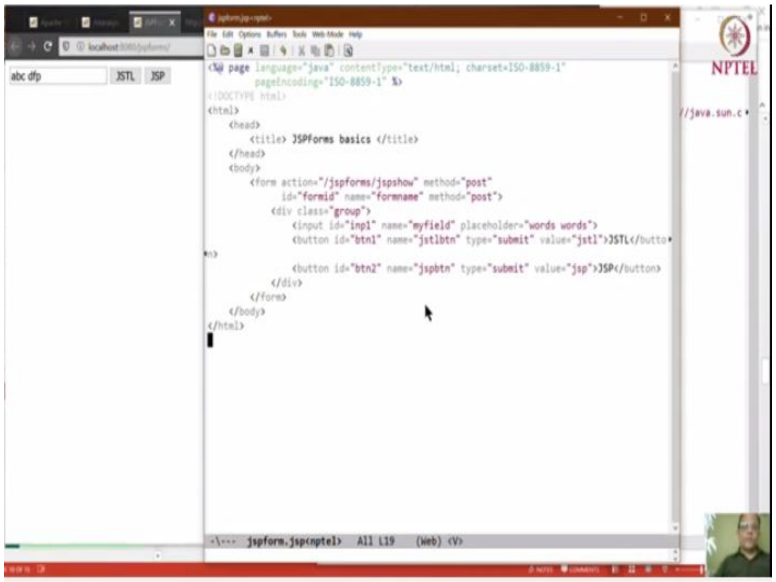


```
    Welcome.

  <p>
    This is the web version of the FairShare application.

  <p>
    <FORM ACTION="/cgi-bin/postdata.sh"
          METHOD="post"
          id="fairShareForm"
          >
      <LABEL for="fairShareInput"
             form="fairShareForm">
        Enter your command sequence terminated by "end"
      </LABEL>
      <!-- form="fairShareForm" -->
      <TEXTAREA id="fairShareInput"
                cols="65"
                row="30"
                placeholder="Enter your commands sequence here."
                name="commands"
                wrap="soft"
                form="fairShareForm"
                ></TEXTAREA>
      <BR>
      <INPUT TYPE="submit" VALUE="Submit Form" ID="fairShareInput">
      <INPUT TYPE="reset"  VALUE="Clear Form"  ID="fairShareInput">
    </FORM>
      <p>
  </div>
</body>
</html>
```

```
-(Unix)--- fairshare-post.html   Bot L174   (HTML+) <V>
```

And just as the html page sends to the cgi bin script, here the jsp sends to a jsp show, which turns out to be a servlet class.

(Refer Slide Time: 06:03)



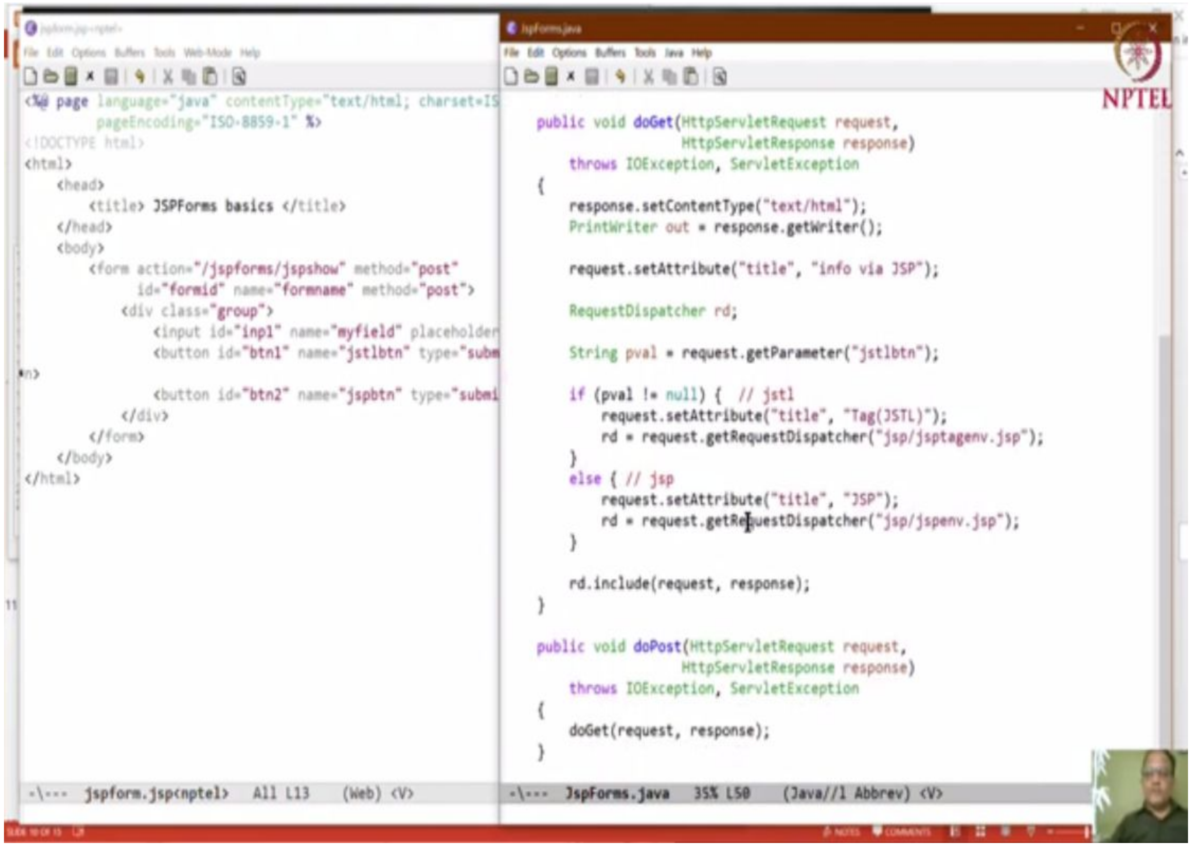
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
    <head>
        <title> JSPForms basics </title>
    </head>
    <body>
        <form action="/jspforms/jspshow" method="post"
              id="formid" name="formname" method="post">
            <div class="group">
                <input id="inp1" name="myfield" placeholder="words words">
                <button id="btn1" name="jstlbtn" type="submit" value="jstl">JSTL</button>

                <button id="btn2" name="jspbtn" type="submit" value="jsp">JSP</button>
            </div>
        </form>
    </body>
</html>
```

```
-\--- jspform.jsp<nptel>   All L19   (Web) <V>
```

Consider the above form, it has 2 buttons, JSTL and JSP, is very similar to our desired app’s GUI (with a text filled and with 3 buttons)

Words are defaults and in practice, upon updating either of the 2 buttons will take us to the back end. There is a form action as usual and an input ID, whose name is my field and here the name is jstl button and jsp button.

(Refer Slide Time: 07:43)



```
<%@ page language="java" contentType="text/html; charset=IS
    pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
    <head>
        <title> JSPForms basics </title>
    </head>
    <body>
        <form action="/jspforms/jspshow" method="post"
            id="formid" name="formname" method="post">
            <div class="group">
                <input id="inp1" name="myfield" placeholder
                <button id="btn1" name="jstlbtn" type="subm
n>
                <button id="btn2" name="jspbtn" type="submi
            </div>
        </form>
    </body>
</html>

-\---  jspform.jsp<nptel>    All L13    (Web) <V>
```

```
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        request.setAttribute("title", "info via JSP");

        RequestDispatcher rd;

        String pval = request.getParameter("jstlbtn");

        if (pval != null) {  // jstl
            request.setAttribute("title", "Tag(JSTL)");
            rd = request.getRequestDispatcher("jsp/jsptagenv.jsp");
        }
        else { // jsp
            request.setAttribute("title", "JSP");
            rd = request.getRequestDispatcher("jsp/jspenv.jsp");
        }

        rd.include(request, response);
    }

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

-\---  JspForms.java    35% L50    (Java//l Abbrev) <V>
```

Jsp Forms class:

There is a request attribute as usual because we are going to use it elsewhere and when that form is clicked, it invokes this class, the post method and a post method in this case actually is no different from get, so it internally sends to the get method.

Need for request dispatcher

Previously in hello JSP, we generated a request dispatcher, which took us from the servlet class to a jsp, here we have arrived from a jsp via a post to this get and for this request, we are going to generate a request dispatcher later and here are the 2 things we do:

1. Check if there is a parameter called jstl button, if so then do jstl processing.
2. Else do jsp processing.

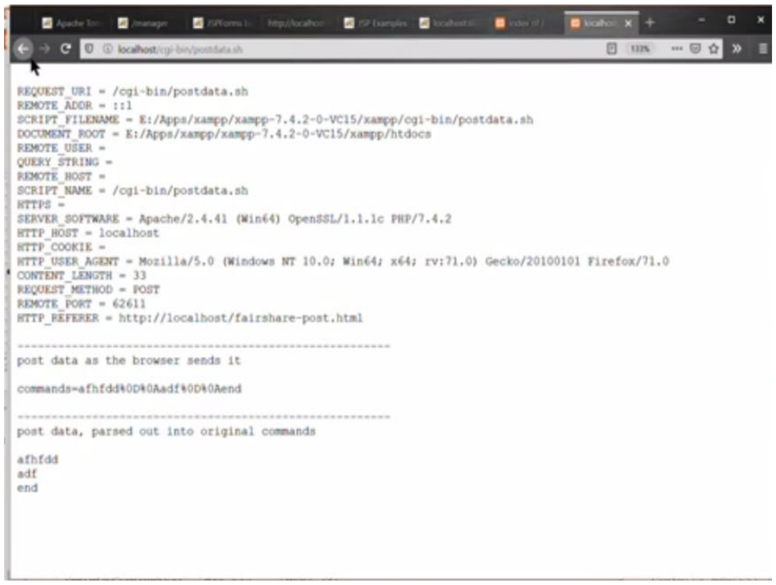
The string arrives via the request parameter (which is the name that we gave), so key value pairs arrive at the servlet and value is the one depending on which button is pressed.

ID of a tag is used on the client side, whereas the name is something which is sent to the server side.

There is a button that submits a form and it sends this value to the back end, when that happens, having determined whether it was jstl or whether it was jsp by using the parameter, two cases arise. In one path, we set the title to jsp and our request dispatcher this time invokes either jspenv .jsp or jsptagenv.jsp, so that is the distinction.

So, we came from the jsp to a servlet class and from a servlet class, we are going again to a jsp, this time there are 2 jsp's.

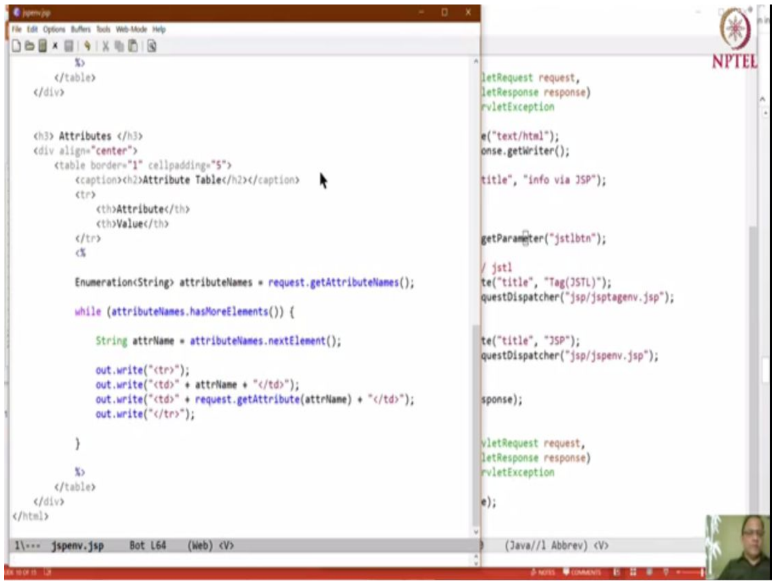
(Refer Slide Time: 11:33)



```
REQUEST_URI = /cgi-bin/postdata.sh
REMOTE_ADDR = ::1
SCRIPT_FILENAME = E:/Apps/xampp/xampp-7.4.2-0-VC15/xampp/cgi-bin/postdata.sh
DOCUMENT_ROOT = E:/Apps/xampp/xampp-7.4.2-0-VC15/xampp/htdocs
REMOTE_USER =
QUERY_STRING =
REMOTE_HOST =
SCRIPT_NAME = /cgi-bin/postdata.sh
HTTPS =
SERVER_SOFTWARE = Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.4.2
HTTP_HOST = localhost
HTTP_COOKIE =
HTTP_USER_AGENT = Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101 Firefox/71.0
CONTENT_LENGTH = 33
REQUEST_METHOD = POST
REMOTE_PORT = 62611
HTTP_REFERER = http://localhost/fairshare-post.html

-------------------------------------------------------
post data as the browser sends it

commands=afhfdd%0D%0Aadf%0D%0Aend

-------------------------------------------------------
post data, parsed out into original commands

afhfdd
adf
end
```

This behavior is similar to fair share when one clicks submit, something gets generated by the back end, okay, in this case, it is just generating whatever you send it and it comes back, if this were html, then we would see it as html but the point is you go from a form to a cgi bin which generates another potential html and that is what is going on here. So, from 1 jsp, you come to this class, from this class, you go to one of these 2 jsp's and what does jspenv look like, let us take a look, so jspenv.

(Refer Slide Time: 12:28)

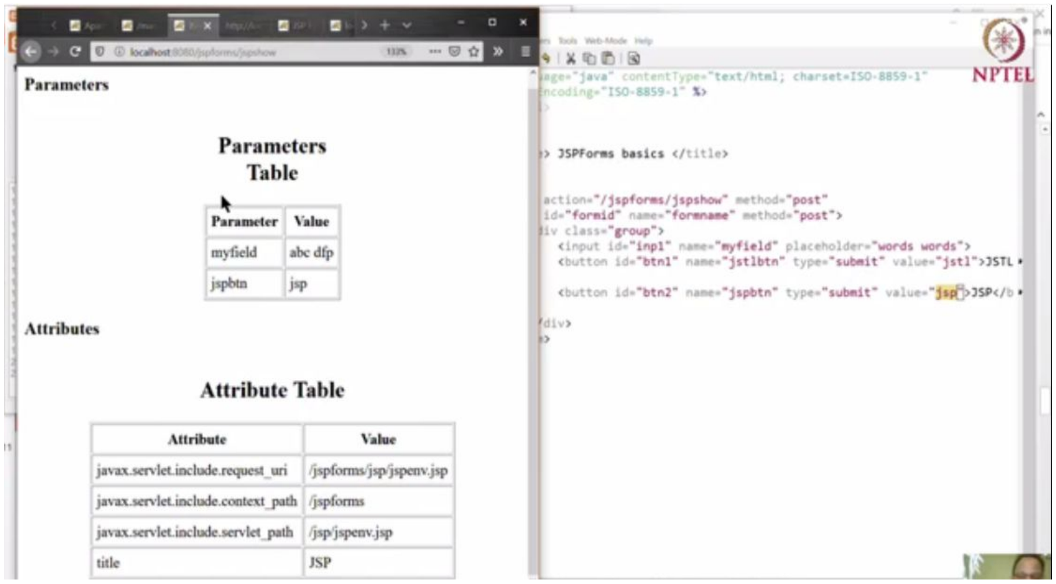


```
        %>
      </table>
    </div>


    <h3> Attributes </h3>
    <div align="center">
      <table border="1" cellpadding="5">
        <caption><h2>Attribute Table</h2></caption>
        <tr>
          <th>Attribute</th>
          <th>Value</th>
        </tr>
        <%

        Enumeration<String> attributeNames = request.getAttributeNames();

        while (attributeNames.hasMoreElements()) {

          String attrName = attributeNames.nextElement();

          out.write("<tr>");
          out.write("<td>" + attrName + "</td>");
          out.write("<td>" + request.getAttribute(attrName) + "</td>");
          out.write("</tr>");

        }

        %>
      </table>
    </div>
</html>
```

```
letRequest request,
letResponse response)
rvletException

e("text/html");
onse.getWriter();

title", "info via JSP");


getParameter("jstlbtn");

/ jstl
te("title", "Tag(JSTL)");
questDispatcher("jsp/jsptagenv.jsp");


te("title", "JSP");
questDispatcher("jsp/jspenv.jsp");


sponse);


vletRequest request,
letResponse response)
rvletException

e);
```

Job of jspenv is to tell all the parameters of the request and all the attributes of the request.

(Refer Slide Time: 12:52)



localhost:8080/jspforms/jspshow

Parameters

Parameters Table

Parameter	Value
myfield	abc dfp
jspbtn	jsp

Attributes

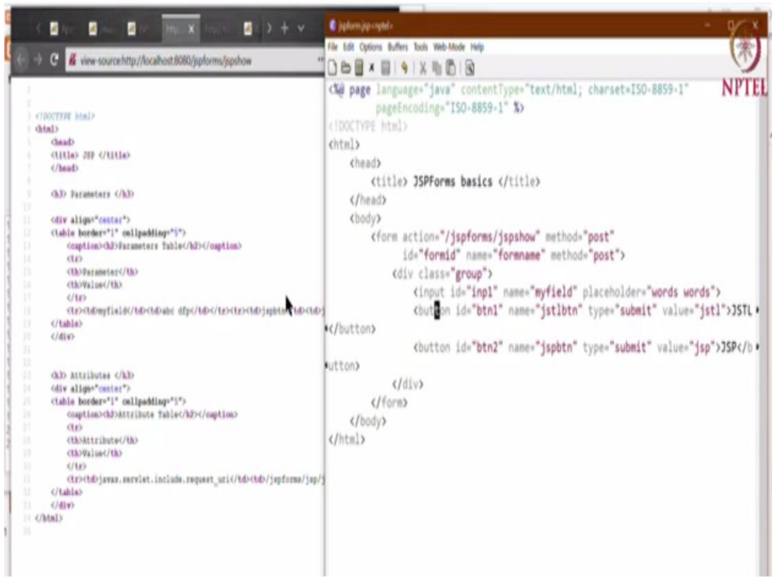
Attribute Table

Attribute	Value
javax.servlet.include.request_uri	/jspforms/jsp/jspenv.jsp
javax.servlet.include.context_path	/jspforms
javax.servlet.include.servlet_path	/jsp/jspenv.jsp
title	JSP

```
age="java" contentType="text/html; charset=ISO-8859-1"
ncoding="ISO-8859-1" %>


> JSPForms basics </title>


 action="/jspforms/jspshow" method="post"
 id="formid" name="formname" method="post">
iv class="group">
  <input id="inp1" name="myfield" placeholder="words words">
  <button id="btn1" name="jstlbtn" type="submit" value="jstl">JSTL

  <button id="btn2" name="jspbtn" type="submit" value="jsp">JSP</b

/div>
>
```

When jsp is clicked, we get the parameters table as shown above, which says that there was a my field and there was a jsp button. So parameter is my field and value in this case is whatever we entered and jsp button is the one whose value happens to be jsp. Now, as far as the attributes are concerned, some are internal and others are entered which are titled jsp.

(Refer Slide Time: 14:06)



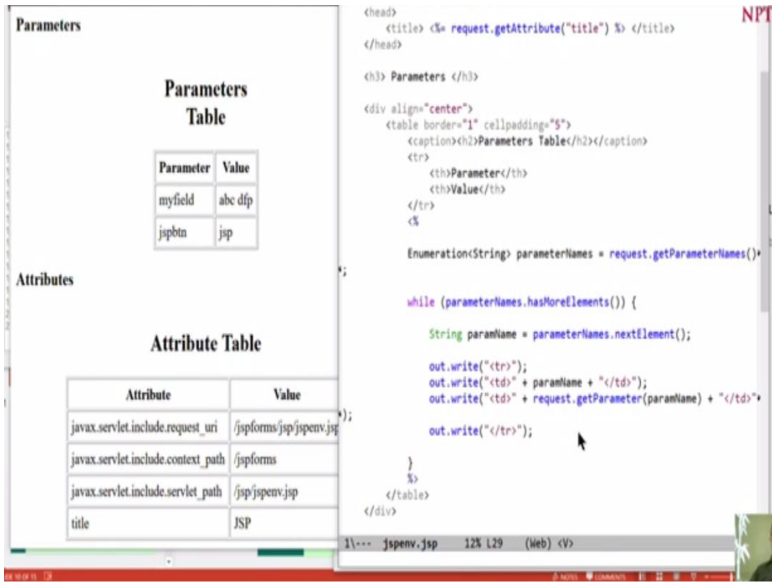
view-source:http://localhost:8080/jspforms/jspshow

```
<!DOCTYPE html>
<html>
    <head>
    <title> JSP </title>
    </head>

    <h3> Parameters </h3>

    <div align="center">
    <table border="1" cellpadding="5">
        <caption><h2>Parameters Table</h2></caption>
        <tr>
        <th>Parameter</th>
        <th>Value</th>
        </tr>
        <tr><td>myfield</td><td>abc dfp</td></tr><tr><td>jspbtn</td><td>
    </table>
    </div>


    <h3> Attributes </h3>
    <div align="center">
    <table border="1" cellpadding="5">
        <caption><h2>Attribute Table</h2></caption>
        <tr>
        <th>Attribute</th>
        <th>Value</th>
        </tr>
        <tr><td>javax.servlet.include.request_uri</td><td>/jspforms/jsp/
    </table>
    </div>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
         pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
    <head>
        <title> JSPForms basics </title>
    </head>
    <body>
        <form action="/jspforms/jspshow" method="post"
              id="formid" name="formname" method="post">
            <div class="group">
                <input id="inp1" name="myfield" placeholder="words words">
                <button id="btn1" name="jstlbtn" type="submit" value="jstl">JSTL
</button>
                <button id="btn2" name="jspbtn" type="submit" value="jsp">JSP</b
utton>
            </div>
        </form>
    </body>
</html>
```

So, what is happening is it is generating 2 tables and in each table, it is generating 2 different rows.

(Refer Slide Time: 14:50)



Parameters

Parameters Table

Parameter	Value
myfield	abc dfp
jspbtn	jsp

Attributes

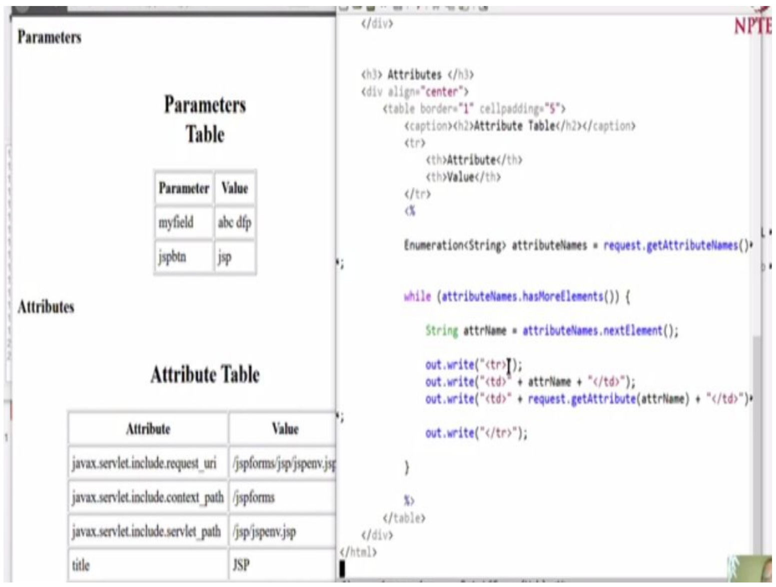
Attribute Table

Attribute	Value
javax.servlet.include.request_uri	/jspforms/jsp/jspenv.jsp
javax.servlet.include.context_path	/jspforms
javax.servlet.include.servlet_path	/jsp/jspenv.jsp
title	JSP

```
<head>
    <title> <%= request.getAttribute("title") %> </title>
</head>

<h3> Parameters </h3>

<div align="center">
    <table border="1" cellpadding="5">
        <caption><h2>Parameters Table</h2></caption>
        <tr>
            <th>Parameter</th>
            <th>Value</th>
        </tr>
        <%

        Enumeration<String> parameterNames = request.getParameterNames()
;

        while (parameterNames.hasMoreElements()) {

            String paramName = parameterNames.nextElement();

            out.write("<tr>");
            out.write("<td>" + paramName + "</td>");
            out.write("<td>" + request.getParameter(paramName) + "</td>"
);

            out.write("</tr>");

        }
        %>
    </table>
</div>
1\--- jspenv.jsp    12% L29    (Web) <V>
```

Java code is similar to before, it gets the request object, get parameter names, etc.

(Refer Slide Time: 16:35)



Parameters

Parameters Table

Parameter	Value
myfield	abc dfp
jspbtn	jsp

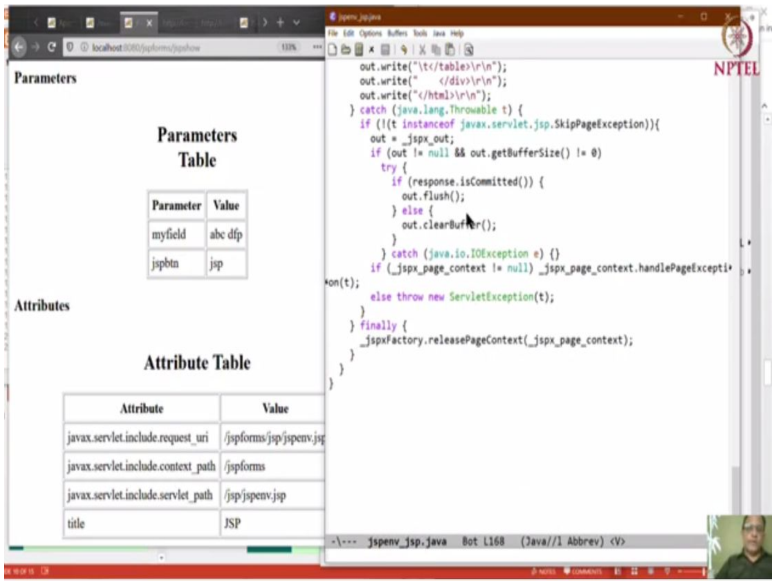
Attributes

Attribute Table

Attribute	Value
javax.servlet.include.request_uri	/jspforms/jsp/jspenv.jsp
javax.servlet.include.context_path	/jspforms
javax.servlet.include.servlet_path	/jsp/jspenv.jsp
title	JSP

```
    </div>


    <h3> Attributes </h3>
    <div align="center">
        <table border="1" cellpadding="5">
            <caption><h2>Attribute Table</h2></caption>
            <tr>
                <th>Attribute</th>
                <th>Value</th>
            </tr>
            <%

            Enumeration<String> attributeNames = request.getAttributeNames()
;


            while (attributeNames.hasMoreElements()) {

                String attrName = attributeNames.nextElement();

                out.write("<tr>");
                out.write("<td>" + attrName + "</td>");
                out.write("<td>" + request.getAttribute(attrName) + "</td>")
;

                out.write("</tr>");

            }

            %>
        </table>
    </div>
</html>
```

(Refer Slide Time: 17:02)



localhost:8080/jspforms/jspshow

Parameters

Parameters Table

Parameter	Value
myfield	abc dfp
jspbtn	jsp

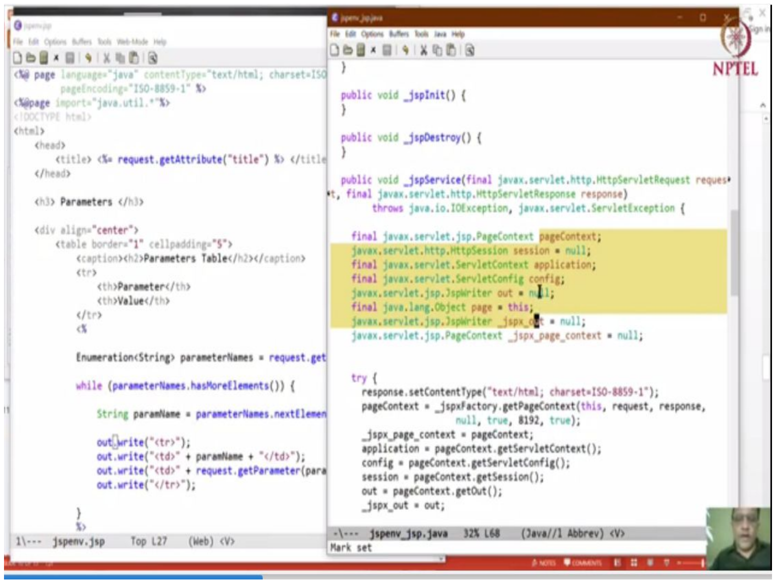
Attributes

Attribute Table

Attribute	Value
javax.servlet.include.request_uri	/jspforms/jsp/jspenv.jsp
javax.servlet.include.context_path	/jspforms
javax.servlet.include.servlet_path	/jsp/jspenv.jsp
title	JSP

```
      out.write("\t</table>\r\n");
      out.write("    </div>\r\n");
      out.write("</html>\r\n");
    } catch (java.lang.Throwable t) {
      if (!(t instanceof javax.servlet.jsp.SkipPageException)){
        out = _jspx_out;
        if (out != null && out.getBufferSize() != 0)
          try {
            if (response.isCommitted()) {
              out.flush();
            } else {
              out.clearBuffer();
            }
          } catch (java.io.IOException e) {}
        if (_jspx_page_context != null) _jspx_page_context.handlePageExcepti
on(t);
        else throw new ServletException(t);
      }
    } finally {
      _jspxFactory.releasePageContext(_jspx_page_context);
    }
  }
}
-\--- jspenv_jsp.java   Bot L168   (Java//l Abbrev) <V>
```

(Refer Slide Time: 18:01)



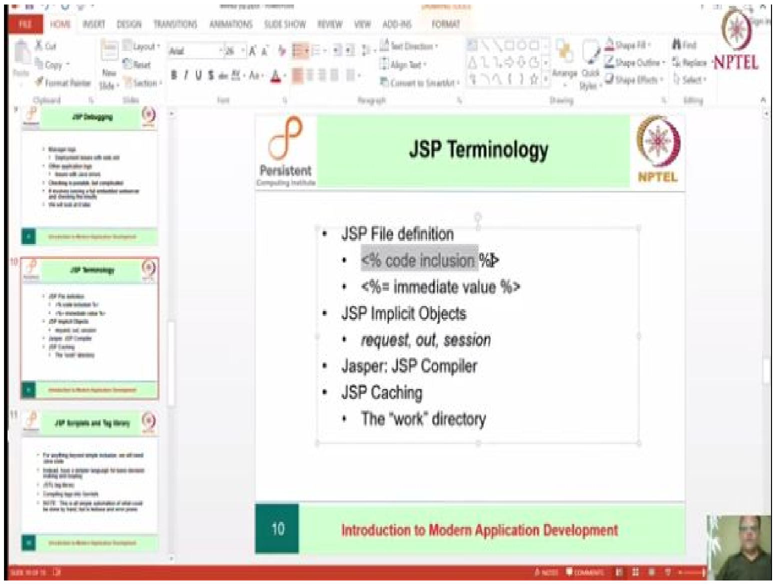
```
<%@ page language="java" contentType="text/html; charset=ISO
        pageEncoding="ISO-8859-1" %>
<%@page import="java.util.*"%>
<!DOCTYPE html>
<html>
    <head>
        <title> <%= request.getAttribute("title") %> </title
    </head>

    <h3> Parameters </h3>

    <div align="center">
        <table border="1" cellpadding="5">
            <caption><h2>Parameters Table</h2></caption>
            <tr>
                <th>Parameter</th>
                <th>Value</th>
            </tr>
            <%

            Enumeration<String> parameterNames = request.get

            while (parameterNames.hasMoreElements()) {

                String paramName = parameterNames.nextElemen

                out.write("<tr>");
                out.write("<td>" + paramName + "</td>");
                out.write("<td>" + request.getParameter(para
                out.write("</tr>");

            }
            %>
1\--- jspenv.jsp     Top L27     (Web) <V>
```

```
  }

  public void _jspInit() {
  }

  public void _jspDestroy() {
  }

  public void _jspService(final javax.servlet.http.HttpServletRequest reques
t, final javax.servlet.http.HttpServletResponse response)
        throws java.io.IOException, javax.servlet.ServletException {

    final javax.servlet.jsp.PageContext pageContext;
    javax.servlet.http.HttpSession session = null;
    final javax.servlet.ServletContext application;
    final javax.servlet.ServletConfig config;
    javax.servlet.jsp.JspWriter out = null;
    final java.lang.Object page = this;
    javax.servlet.jsp.JspWriter _jspx_out = null;
    javax.servlet.jsp.PageContext _jspx_page_context = null;


    try {
      response.setContentType("text/html; charset=ISO-8859-1");
      pageContext = _jspxFactory.getPageContext(this, request, response,
                  null, true, 8192, true);
      _jspx_page_context = pageContext;
      application = pageContext.getServletContext();
      config = pageContext.getServletConfig();
      session = pageContext.getSession();
      out = pageContext.getOut();
      _jspx_out = out;

-\--- jspenv_jsp.java    32% L68    (Java//l Abbrev) <V>
Mark set
```

Since the previous code is added to the context of a full servlet, terms like request carry the same meaning that they did for the servlet.

SUMMARY: Starting with web.xml which points out that entry point which is this URL for jspforms takes us to the jsp file which contains an action which calls jspforms and in turn invokes jspenv.jsp, inside the class by binding and that choice in turn is based on the button.

(Refer Slide Time: 20:29)



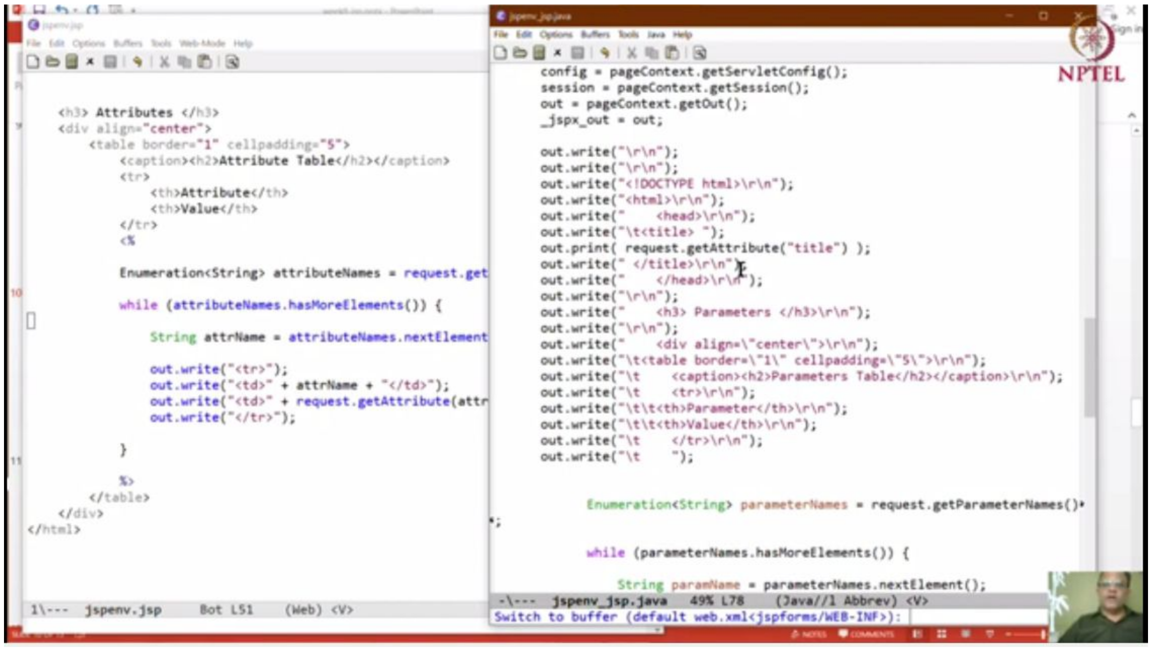
JSP Terminology

- JSP File definition
  - <% code inclusion %>
  - <%= immediate value %>
- JSP Implicit Objects
  - *request, out, session*
- Jasper: JSP Compiler
- JSP Caching
  - The "work" directory

10 Introduction to Modern Application Development

In the JSP file definition, you can straight away include code or you can include a value as shown above.

(Refer Slide Time: 20:41)



```
<h3> Attributes </h3>
<div align="center">
    <table border="1" cellpadding="5">
        <caption><h2>Attribute Table</h2></caption>
        <tr>
            <th>Attribute</th>
            <th>Value</th>
        </tr>
        <%

        Enumeration<String> attributeNames = request.get

        while (attributeNames.hasMoreElements()) {

            String attrName = attributeNames.nextElement

            out.write("<tr>");
            out.write("<td>" + attrName + "</td>");
            out.write("<td>" + request.getAttribute(attr
            out.write("</tr>");

        }

        %>
    </table>
</div>
</html>
```

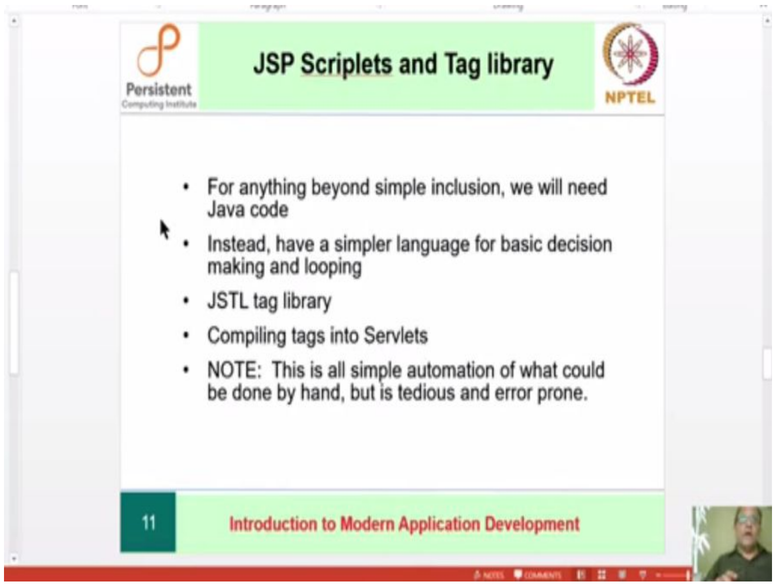
```
1\---  jspenv.jsp     Bot L51     (Web) <V>
```

```
    config = pageContext.getServletConfig();
    session = pageContext.getSession();
    out = pageContext.getOut();
    _jspx_out = out;

    out.write("\r\n");
    out.write("\r\n");
    out.write("<!DOCTYPE html>\r\n");
    out.write("<html>\r\n");
    out.write("    <head>\r\n");
    out.write("\t<title> ");
    out.print( request.getAttribute("title") );
    out.write(" </title>\r\n");
    out.write("    </head>\r\n");
    out.write("\r\n");
    out.write("    <h3> Parameters </h3>\r\n");
    out.write("\r\n");
    out.write("    <div align=\"center\">\r\n");
    out.write("\t<table border=\"1\" cellpadding=\"5\">\r\n");
    out.write("\t    <caption><h2>Parameters Table</h2></caption>\r\n");
    out.write("\t    <tr>\r\n");
    out.write("\t\t<th>Parameter</th>\r\n");
    out.write("\t\t<th>Value</th>\r\n");
    out.write("\t    </tr>\r\n");
    out.write("\t    ");


        Enumeration<String> parameterNames = request.getParameterNames()
;

        while (parameterNames.hasMoreElements()) {

            String paramName = parameterNames.nextElement();
```

```
-\---  jspenv_jsp.java   49% L78    (Java//l Abbrev) <V>
Switch to buffer (default web.xml<jspforms/WEB-INF>):
```

Consider the above figure, writing using out.write ourselves does not make sense when the example is complicated. Instead of that with the code on the left window, which are called **scriplets** (little pieces of Java code) can make it more readable and elegant.

(Refer Slide Time: 21:38)



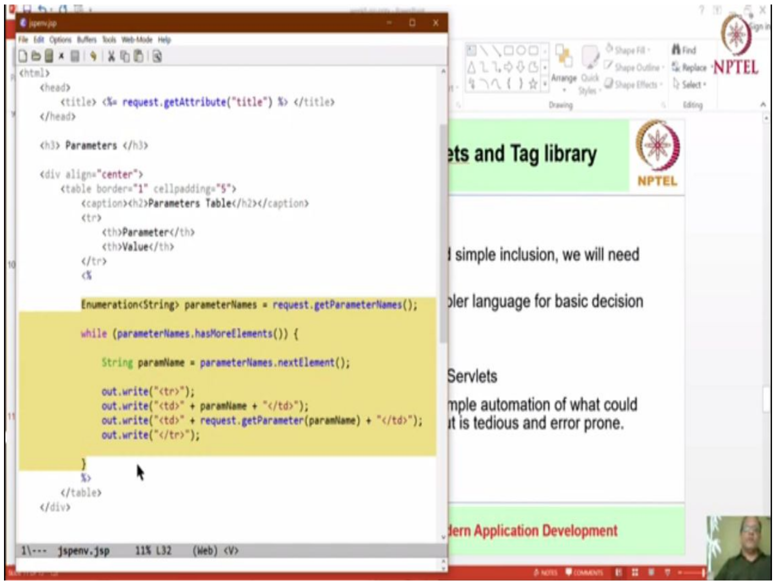
JSP Scriplets and Tag library

- For anything beyond simple inclusion, we will need Java code
- Instead, have a simpler language for basic decision making and looping
- JSTL tag library
- Compiling tags into Servlets
- NOTE: This is all simple automation of what could be done by hand, but is tedious and error prone.

11 Introduction to Modern Application Development

So for anything beyond simple inclusion, it is better to use Java code.

(Refer Slide Time: 21:49)



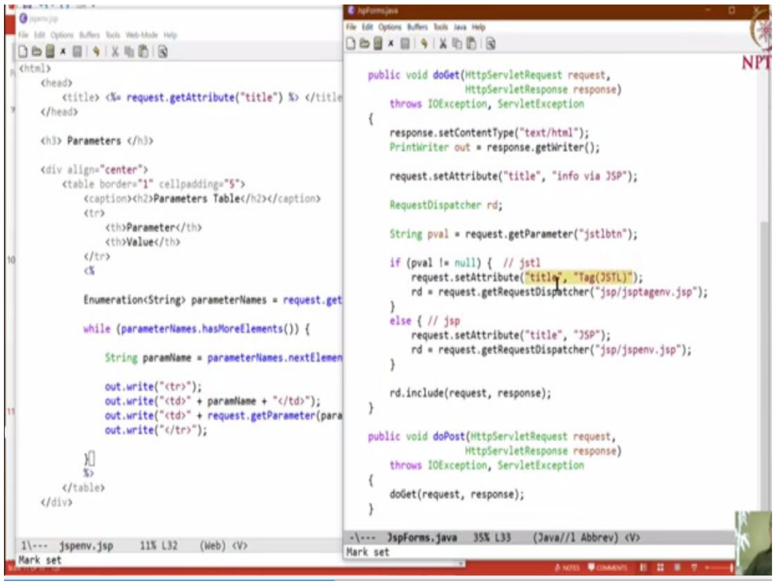
```
<html>
    <head>
        <title> <%= request.getAttribute("title") %> </title>
    </head>

    <h3> Parameters </h3>

    <div align="center">
        <table border="1" cellpadding="5">
            <caption><h2>Parameters Table</h2></caption>
            <tr>
                <th>Parameter</th>
                <th>Value</th>
            </tr>
            <%

            Enumeration<String> parameterNames = request.getParameterNames();

            while (parameterNames.hasMoreElements()) {

                String paramName = parameterNames.nextElement();

                out.write("<tr>");
                out.write("<td>" + paramName + "</td>");
                out.write("<td>" + request.getParameter(paramName) + "</td>");
                out.write("</tr>");

            }
            %>
        </table>
    </div>

1\---  jspenv.jsp     11% L32     (Web) <V>
```

ets and Tag library
NPTEL
simple inclusion, we will need
pler language for basic decision
Servlets
mple automation of what could
it is tedious and error prone.
dern Application Development

Tag library.

Tag libraries are a way of simplifying loops, decision making etc., that arise while designing systems. One goal behind things like tag libraries was that since the front end i.e. forms and layouts are done by designers who are not necessarily programmers, we want a much simpler language over there which is easier for designers to understand. It is lot more like html.

(Refer Slide Time: 22:50)

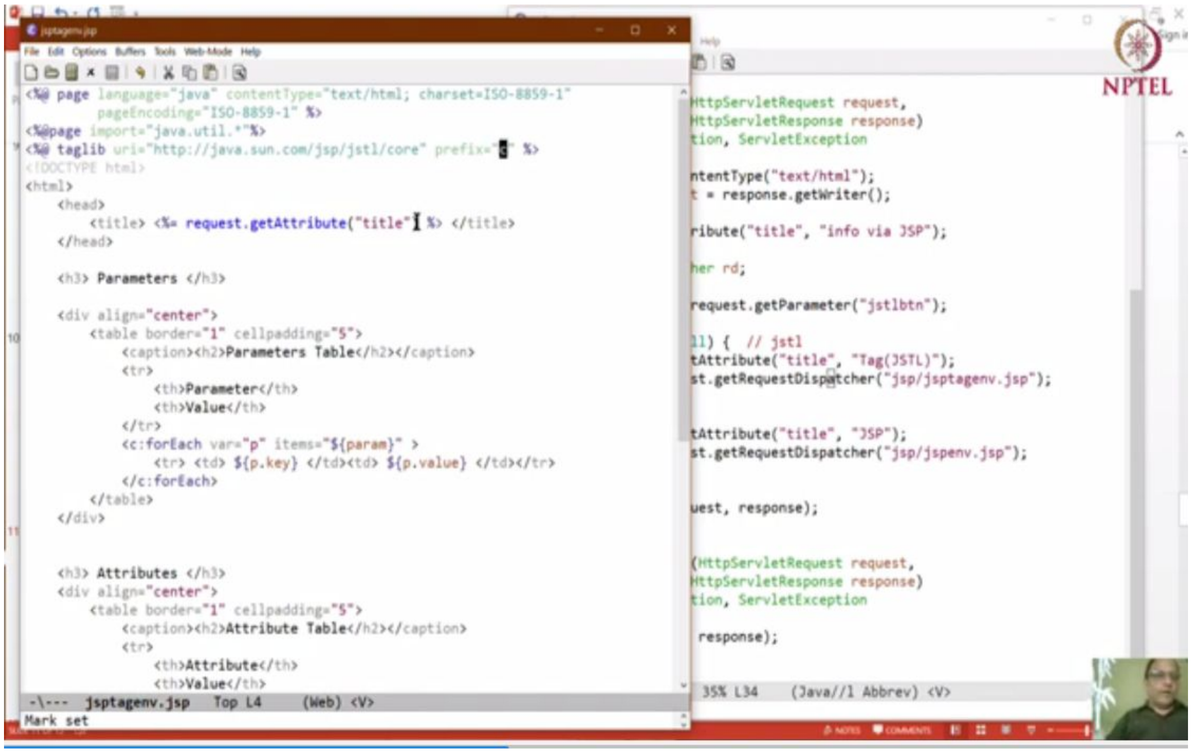


```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws IOException, ServletException
{
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    request.setAttribute("title", "info via JSP");

    RequestDispatcher rd;

    String pval = request.getParameter("jstlbtn");

    if (pval != null) {  // jstl
        request.setAttribute("title", "Tag(JSTL)");
        rd = request.getRequestDispatcher("jsp/jsptagenv.jsp");
    }
    else { // jsp
        request.setAttribute("title", "JSP");
        rd = request.getRequestDispatcher("jsp/jspenv.jsp");
    }

    rd.include(request, response);
}

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

-\---  JspForms.java    35% L33     (Java//l Abbrev) <V>
Mark set
```

The second scenario : JSTL

While we learnt about implementing using jsp through the jsp button which involves these things called scriplets, tag libraries are a simplified version but achieves the same.

(Refer Slide Time: 23:09)

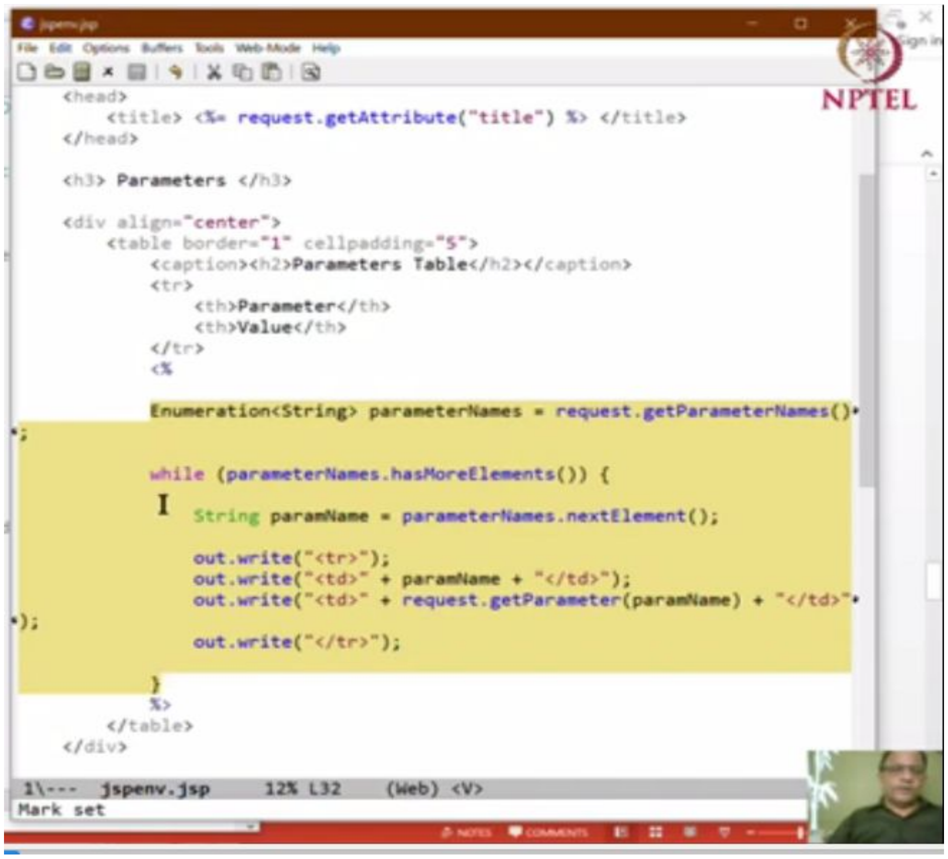


```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
         pageEncoding="ISO-8859-1" %>
<%@page import="java.util.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
    <head>
        <title> <%= request.getAttribute("title") %> </title>
    </head>

    <h3> Parameters </h3>

    <div align="center">
        <table border="1" cellpadding="5">
            <caption><h2>Parameters Table</h2></caption>
            <tr>
                <th>Parameter</th>
                <th>Value</th>
            </tr>
            <c:forEach var="p" items="${param}" >
                <tr> <td> ${p.key} </td><td> ${p.value} </td></tr>
            </c:forEach>
        </table>
    </div>


    <h3> Attributes </h3>
    <div align="center">
        <table border="1" cellpadding="5">
            <caption><h2>Attribute Table</h2></caption>
            <tr>
                <th>Attribute</th>
                <th>Value</th>
-\---  jsptagenv.jsp    Top L4     (Web) <V>
Mark set
```

```
HttpServletRequest request,
HttpServletResponse response)
tion, ServletException

ntentType("text/html");
t = response.getWriter();

ribute("title", "info via JSP");

her rd;

request.getParameter("jstlbtn");

ll) {  // jstl
tAttribute("title", "Tag(JSTL)");
st.getRequestDispatcher("jsp/jsptagenv.jsp");


tAttribute("title", "JSP");
st.getRequestDispatcher("jsp/jspenv.jsp");


uest, response);


(HttpServletRequest request,
HttpServletResponse response)
tion, ServletException

 response);
 35% L34    (Java//l Abbrev) <V>
```

Tag origin: Need to include the first two lines in the above figure which specifies that tag library will be used.

(Refer Slide Time: 23:37)

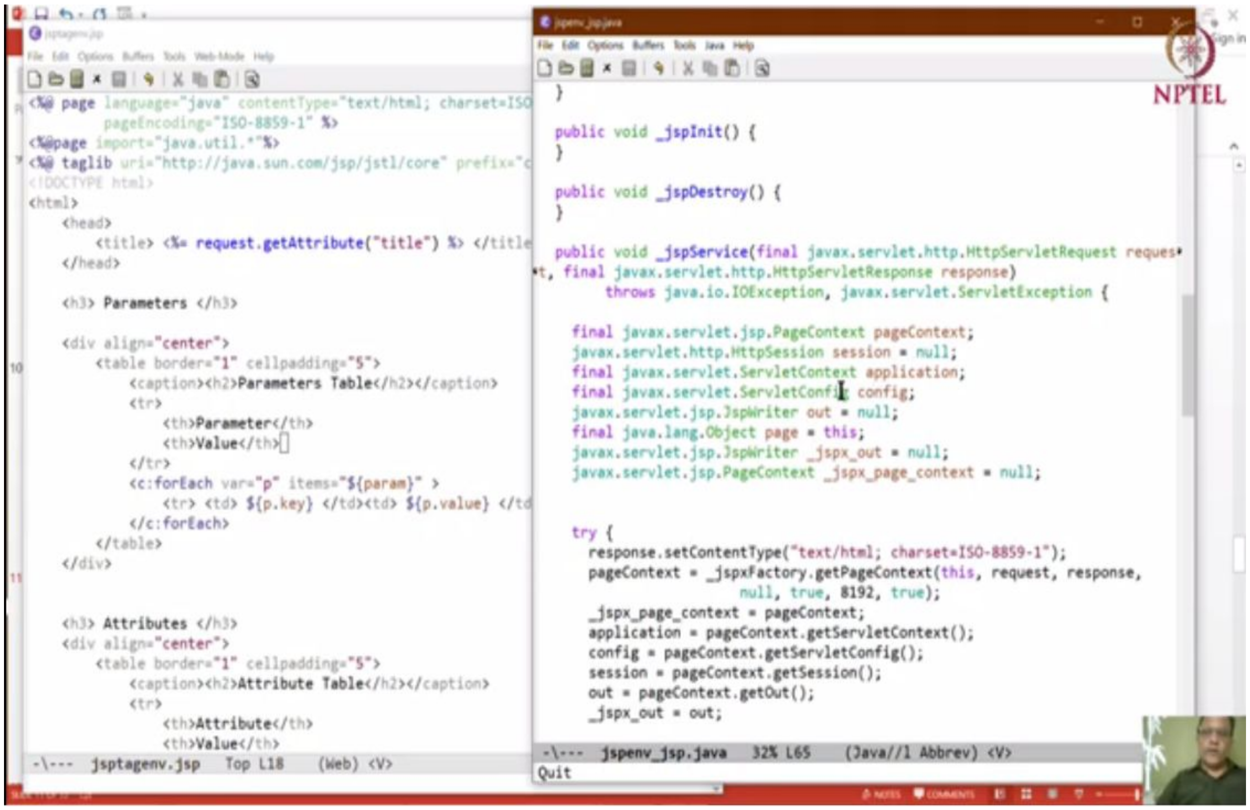


```
    <head>
        <title> <%= request.getAttribute("title") %> </title>
    </head>

    <h3> Parameters </h3>

    <div align="center">
        <table border="1" cellpadding="5">
            <caption><h2>Parameters Table</h2></caption>
            <tr>
                <th>Parameter</th>
                <th>Value</th>
            </tr>
            <%

            Enumeration<String> parameterNames = request.getParameterNames()•
•;

            while (parameterNames.hasMoreElements()) {

                String paramName = parameterNames.nextElement();

                out.write("<tr>");
                out.write("<td>" + paramName + "</td>");
                out.write("<td>" + request.getParameter(paramName) + "</td>"•
•);

                out.write("</tr>");

            }
            %>
        </table>
    </div>
```

```
1\---  jspenv.jsp    12% L32    (Web) <V>
Mark set
```

tr, td is used in a loop to generate a row and column respectively. *$param refers to the parameters of the request is an implicit object for jstl.*

(Refer Slide Time: 24:40)

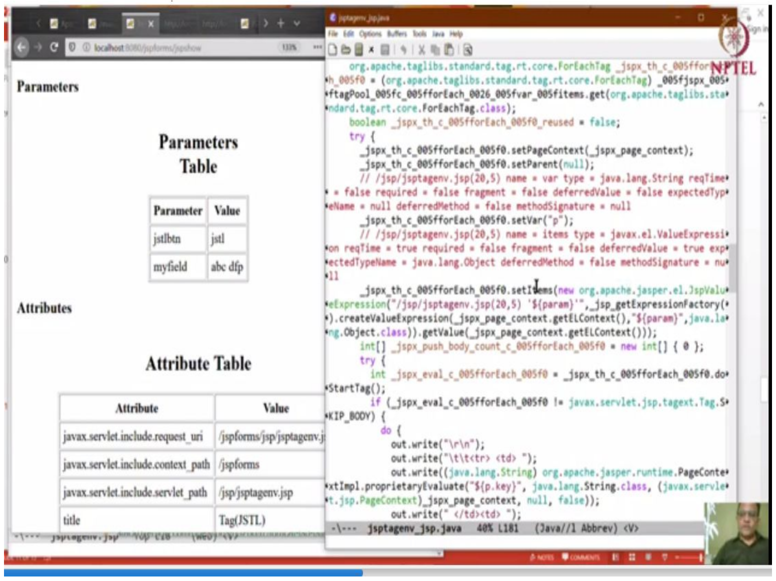


```
<%@ page language="java" contentType="text/html; charset=ISO
        pageEncoding="ISO-8859-1" %>
<%@page import="java.util.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c
<!DOCTYPE html>
<html>
    <head>
        <title> <%= request.getAttribute("title") %> </title
    </head>

    <h3> Parameters </h3>

    <div align="center">
        <table border="1" cellpadding="5">
            <caption><h2>Parameters Table</h2></caption>
            <tr>
                <th>Parameter</th>
                <th>Value</th>
            </tr>
            <c:forEach var="p" items="${param}" >
                <tr> <td> ${p.key} </td><td> ${p.value} </td
            </c:forEach>
        </table>
    </div>


    <h3> Attributes </h3>
    <div align="center">
        <table border="1" cellpadding="5">
            <caption><h2>Attribute Table</h2></caption>
            <tr>
                <th>Attribute</th>
                <th>Value</th>
-\---  jsptagenv.jsp    Top L18    (Web) <V>
```

```
  }

  public void _jspInit() {
  }

  public void _jspDestroy() {
  }

  public void _jspService(final javax.servlet.http.HttpServletRequest reques
t, final javax.servlet.http.HttpServletResponse response)
        throws java.io.IOException, javax.servlet.ServletException {

    final javax.servlet.jsp.PageContext pageContext;
    javax.servlet.http.HttpSession session = null;
    final javax.servlet.ServletContext application;
    final javax.servlet.ServletConfig config;
    javax.servlet.jsp.JspWriter out = null;
    final java.lang.Object page = this;
    javax.servlet.jsp.JspWriter _jspx_out = null;
    javax.servlet.jsp.PageContext _jspx_page_context = null;


    try {
      response.setContentType("text/html; charset=ISO-8859-1");
      pageContext = _jspxFactory.getPageContext(this, request, response,
                  null, true, 8192, true);
      _jspx_page_context = pageContext;
      application = pageContext.getServletContext();
      config = pageContext.getServletConfig();
      session = pageContext.getSession();
      out = pageContext.getOut();
      _jspx_out = out;
-\---  jspenv_jsp.java   32% L65    (Java//l Abbrev) <V>
Quit
```

The generated code for the jstl contains predefined variables. Param is a java map which is a key and a value pair.

(Refer Slide Time: 25:12)



Parameters

Parameters Table

Parameter	Value
jstlbtn	jstl
myfield	abc dfp

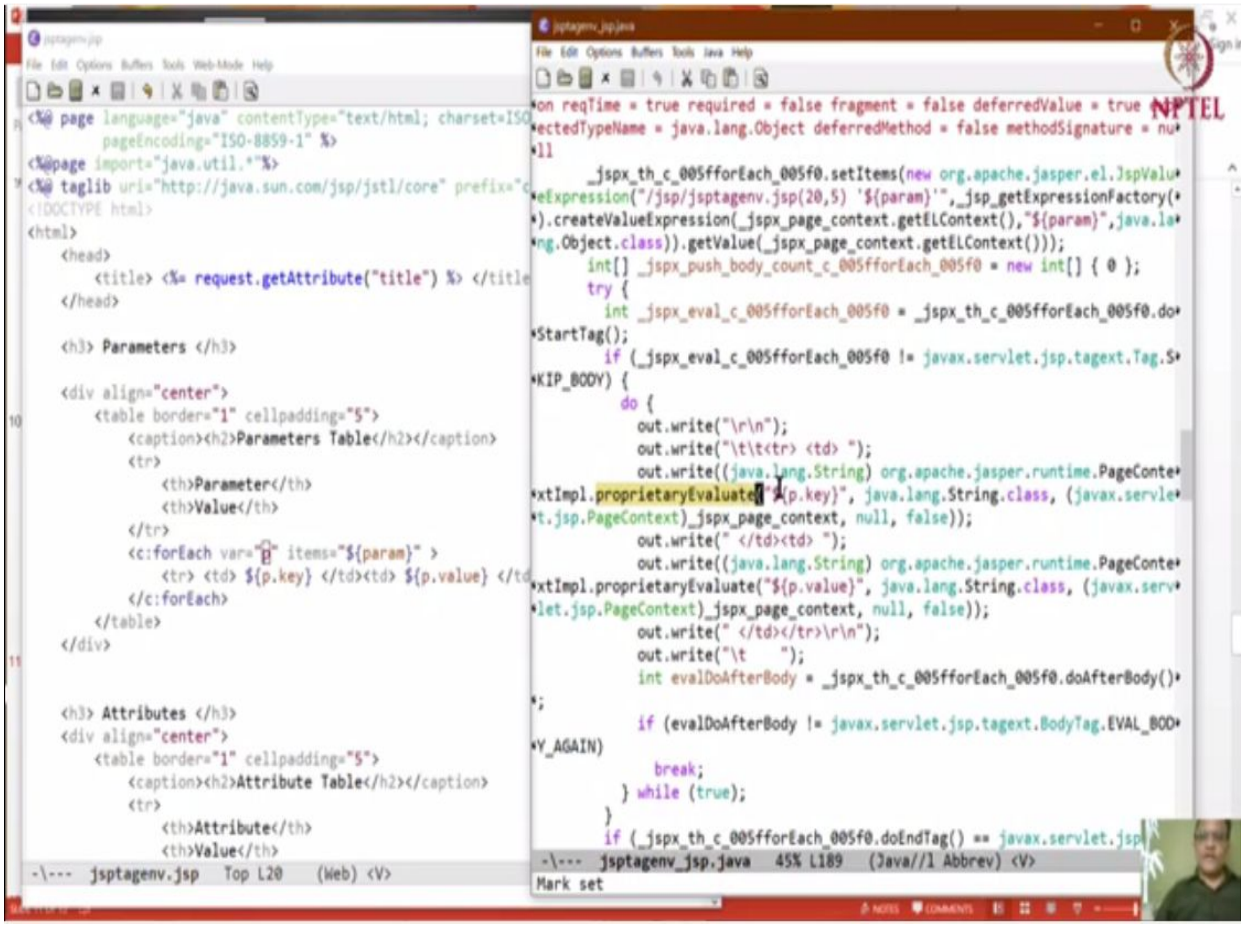
Attributes

Attribute Table

Attribute	Value
javax.servlet.include.request_uri	/jspforms/jsp/jsptagenv.j
javax.servlet.include.context_path	/jspforms
javax.servlet.include.servlet_path	/jsp/jsptagenv.jsp
title	Tag(JSTL)

```
    org.apache.taglibs.standard.tag.rt.core.ForEachTag _jspx_th_c_005ffor
h_005f0 = (org.apache.taglibs.standard.tag.rt.core.ForEachTag) _005fjspx_005
ftagPool_005fc_005fforEach_0026_005fvar_005fitems.get(org.apache.taglibs.sta
ndard.tag.rt.core.ForEachTag.class);
    boolean _jspx_th_c_005fforEach_005f0_reused = false;
    try {
      _jspx_th_c_005fforEach_005f0.setPageContext(_jspx_page_context);
      _jspx_th_c_005fforEach_005f0.setParent(null);
      // /jsp/jsptagenv.jsp(20,5) name = var type = java.lang.String reqTime
 = false required = false fragment = false deferredValue = false expectedTyp
eName = null deferredMethod = false methodSignature = null
      _jspx_th_c_005fforEach_005f0.setVar("p");
      // /jsp/jsptagenv.jsp(20,5) name = items type = javax.el.ValueExpressi
on reqTime = true required = false fragment = false deferredValue = true exp
ectedTypeName = java.lang.Object deferredMethod = false methodSignature = nu
ll
      _jspx_th_c_005fforEach_005f0.setItems(new org.apache.jasper.el.JspValu
eExpression("/jsp/jsptagenv.jsp(20,5) '${param}'",_jsp_getExpressionFactory(
).createValueExpression(_jspx_page_context.getELContext(),"${param}",java.la
ng.Object.class)).getValue(_jspx_page_context.getELContext()));
      int[] _jspx_push_body_count_c_005fforEach_005f0 = new int[] { 0 };
      try {
        int _jspx_eval_c_005fforEach_005f0 = _jspx_th_c_005fforEach_005f0.do
StartTag();
        if (_jspx_eval_c_005fforEach_005f0 != javax.servlet.jsp.tagext.Tag.S
KIP_BODY) {
          do {
            out.write("\r\n");
            out.write("\t\t<tr> <td> ");
            out.write((java.lang.String) org.apache.jasper.runtime.PageConte
xtImpl.proprietaryEvaluate("${p.key}", java.lang.String.class, (javax.servle
t.jsp.PageContext)_jspx_page_context, null, false));
            out.write(" </td><td> ");
-\---  jsptagenv_jsp.java   40% L181    (Java//l Abbrev) <V>
```

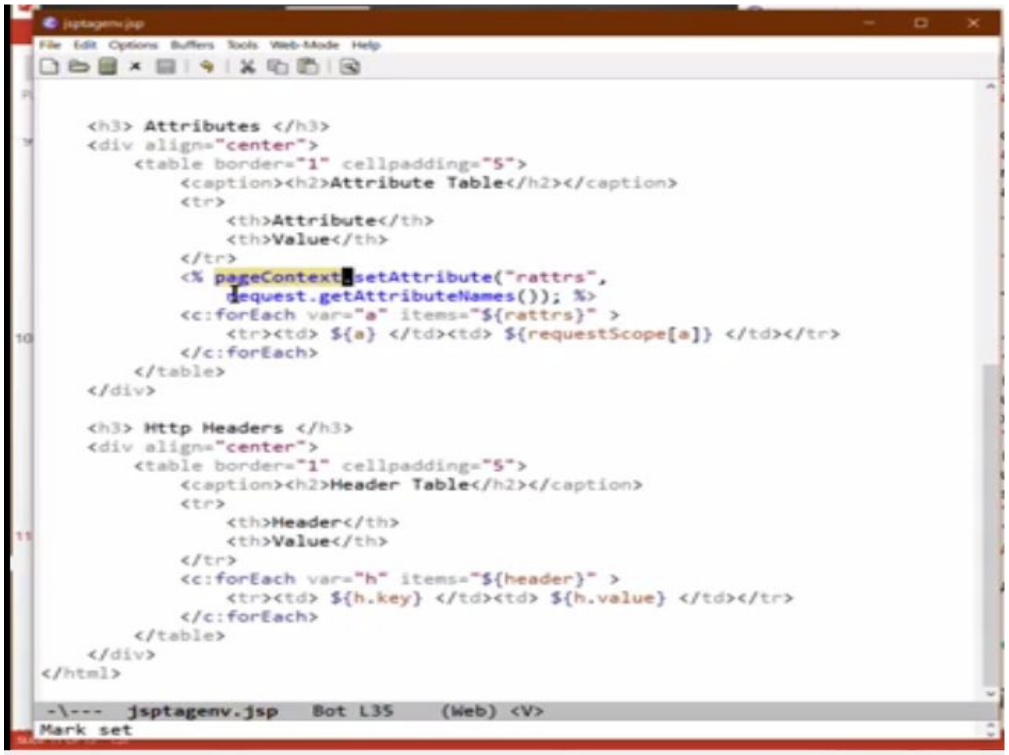
(Refer Slide Time: 26:30)



```
<%@ page language="java" contentType="text/html; charset=ISO
        pageEncoding="ISO-8859-1" %>
<%@page import="java.util.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c
<!DOCTYPE html>
<html>
    <head>
        <title> <%= request.getAttribute("title") %> </title
    </head>

    <h3> Parameters </h3>

    <div align="center">
        <table border="1" cellpadding="5">
            <caption><h2>Parameters Table</h2></caption>
            <tr>
                <th>Parameter</th>
                <th>Value</th>
            </tr>
            <c:forEach var="p" items="${param}" >
                <tr> <td> ${p.key} </td><td> ${p.value} </td
            </c:forEach>
        </table>
    </div>


    <h3> Attributes </h3>
    <div align="center">
        <table border="1" cellpadding="5">
            <caption><h2>Attribute Table</h2></caption>
            <tr>
                <th>Attribute</th>
                <th>Value</th>
-\---  jsptagenv.jsp   Top L20    (Web) <V>
```

```
on reqTime = true required = false fragment = false deferredValue = true
ectedTypeName = java.lang.Object deferredMethod = false methodSignature = nu
ll
        _jspx_th_c_005fforEach_005f0.setItems(new org.apache.jasper.el.JspValu
eExpression("/jsp/jsptagenv.jsp(20,5) '${param}'",_jsp_getExpressionFactory(
).createValueExpression(_jspx_page_context.getELContext(),"${param}",java.la
ng.Object.class)).getValue(_jspx_page_context.getELContext()));
        int[] _jspx_push_body_count_c_005fforEach_005f0 = new int[] { 0 };
        try {
          int _jspx_eval_c_005fforEach_005f0 = _jspx_th_c_005fforEach_005f0.do
StartTag();
          if (_jspx_eval_c_005fforEach_005f0 != javax.servlet.jsp.tagext.Tag.S
KIP_BODY) {
            do {
              out.write("\r\n");
              out.write("\t\t<tr> <td> ");
              out.write((java.lang.String) org.apache.jasper.runtime.PageConte
xtImpl.proprietaryEvaluate("${p.key}", java.lang.String.class, (javax.servle
t.jsp.PageContext)_jspx_page_context, null, false));
              out.write(" </td><td> ");
              out.write((java.lang.String) org.apache.jasper.runtime.PageConte
xtImpl.proprietaryEvaluate("${p.value}", java.lang.String.class, (javax.serv
let.jsp.PageContext)_jspx_page_context, null, false));
              out.write(" </td></tr>\r\n");
              out.write("\t    ");
              int evalDoAfterBody = _jspx_th_c_005fforEach_005f0.doAfterBody()
;

              if (evalDoAfterBody != javax.servlet.jsp.tagext.BodyTag.EVAL_BOD
Y_AGAIN)
                break;
            } while (true);
          }
          if (_jspx_th_c_005fforEach_005f0.doEndTag() == javax.servlet.jsp
-\---  jsptagenv_jsp.java   45% L189   (Java//l Abbrev) <V>
Mark set
```

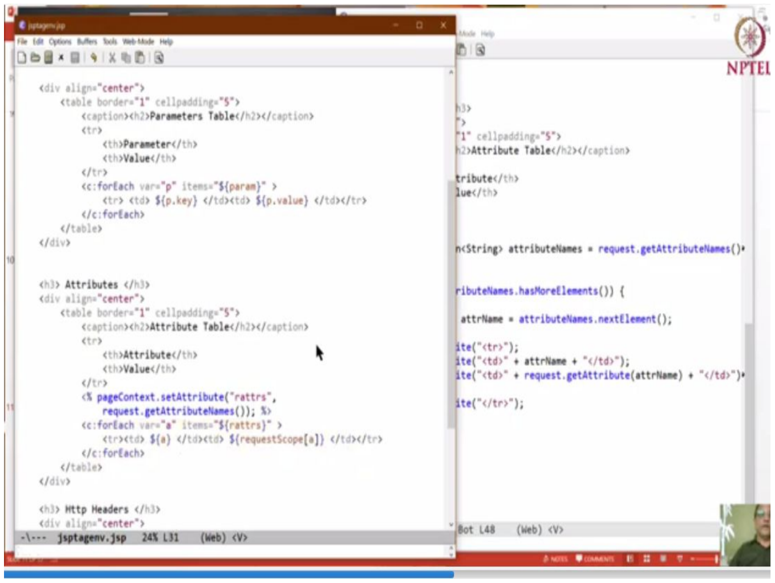
Proprietary evaluate corresponds to the entry of the map, which is p.key where the runtime is actually evaluating these things in the page context.**(Refer Slide Time: 27:05)**



```
<h3> Attributes </h3>
<div align="center">
    <table border="1" cellpadding="5">
        <caption><h2>Attribute Table</h2></caption>
        <tr>
            <th>Attribute</th>
            <th>Value</th>
        </tr>
        <% pageContext.setAttribute("rattrs",
            request.getAttributeNames()); %>
        <c:forEach var="a" items="${rattrs}" >
            <tr><td> ${a} </td><td> ${requestScope[a]} </td></tr>
        </c:forEach>
    </table>
</div>

<h3> Http Headers </h3>
<div align="center">
    <table border="1" cellpadding="5">
        <caption><h2>Header Table</h2></caption>
        <tr>
            <th>Header</th>
            <th>Value</th>
        </tr>
        <c:forEach var="h" items="${header}" >
            <tr><td> ${h.key} </td><td> ${h.value} </td></tr>
        </c:forEach>
    </table>
</div>
</html>
-\---  jsptagenv.jsp   Bot L35    (Web) <V>
Mark set
```

Just as param is a built-in or an implicit object, http headers are also available as implicit objects and the loop over those is equally straightforward but the loop over attributes is a bit funky, this time the tag library alone does not do the job, instead we have to use old style jsp code and in this object something called page context, you have to actually get all the attributes and set it over here.

(Refer Slide Time: 27:46)



```
<div align="center">
    <table border="1" cellpadding="5">
        <caption><h2>Parameters Table</h2></caption>
        <tr>
            <th>Parameter</th>
            <th>Value</th>
        </tr>
        <c:forEach var="p" items="${param}" >
            <tr> <td> ${p.key} </td><td> ${p.value} </td></tr>
        </c:forEach>
    </table>
</div>


<h3> Attributes </h3>
<div align="center">
    <table border="1" cellpadding="5">
        <caption><h2>Attribute Table</h2></caption>
        <tr>
            <th>Attribute</th>
            <th>Value</th>
        </tr>
        <% pageContext.setAttribute("rattrs",
            request.getAttributeNames()); %>
        <c:forEach var="a" items="${rattrs}" >
            <tr><td> ${a} </td><td> ${requestScope[a]} </td></tr>
        </c:forEach>
    </table>
</div>

<h3> Http Headers </h3>
<div align="center">
-\---  jsptagenv.jsp   24% L31    (Web) <V>
```

```
h3>
">
"1" cellpadding="5">
h2>Attribute Table</h2></caption>

tribute</th>
lue</th>



n<String> attributeNames = request.getAttributeNames()

ributeNames.hasMoreElements()) {

 attrName = attributeNames.nextElement();

ite("<tr>");
ite("<td>" + attrName + "</td>");
ite("<td>" + request.getAttribute(attrName) + "</td>")

ite("</tr>");

Bot L48    (Web) <V>
```

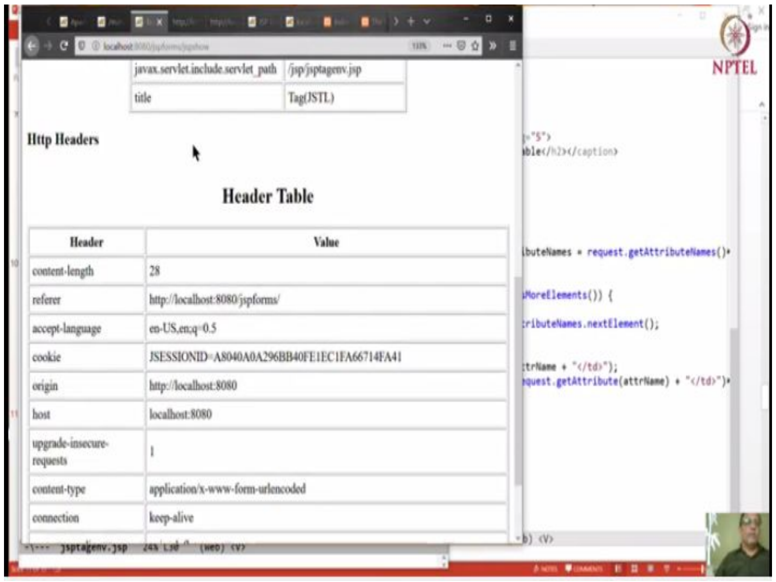
In short implicit objects of jsp are also available within jstl but jstl is a more elaborated version of jsp and so what we have is you have to actually generate one of these attributes which then becomes available as a variable.

So, somewhere in that generated code, this param variable is also created in the same style but it is implicit in the code rather than we have in to do it. Now, all of these complexities you will become more familiar with them in time, I do not expect that this 1 lecture is going to; you should not worry too much, if you do not understand all the nuances, see the template idea begin small, simple jsp is look nice and when jstl's and all come in all sorts of new things start happening but it is just a layered extension of these ideas.

And by the way to get started doing something like this is perfectly fine. Once you get used to it, then you can decide whether something like a tag library is this what you need, now as we will see these kinds of complicated server side creations are not necessarily that common nowadays, there are other alternatives but in practice you will encounter a lot of code like this, if you go to work in industry and happen to work in a Java shop, furthermore these are not limited to Java as we will see later.

Bottom line for now, all you have to remember is this, there is a simple way to do for loops, instead of doing it through code, for some types of maps over, which you can do, they are already built in, for other types of maps, you will have to create them yourselves and the same is true if you have user generated maps in which case also you can use these kind of objects in order to set, create your own map and then you can use the for loop to go over it.

(Refer Slide Time: 30:56)



javax.servlet.include.servlet_path	/jsp/jsptagenv.jsp
title	Tag(JSTL)

Http Headers

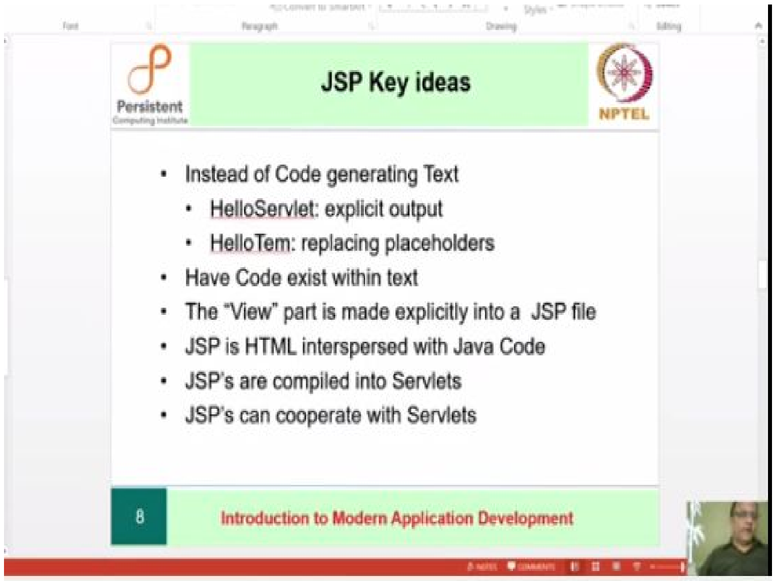
Header Table

Header	Value
content-length	28
referer	http://localhost:8080/jspforms/
accept-language	en-US,en;q=0.5
cookie	JSESSIONID=A8040A0A296BB40FE1EC1FA66714FA41
origin	http://localhost:8080
host	localhost:8080
upgrade-insecure-requests	1
content-type	application/x-www-form-urlencoded
connection	keep-alive

```
buteNames = request.getAttributeNames()+
MoreElements()) {
ributeNames.nextElement();
trName + "</td>");
quest.getAttribute(attrName) + "</td>")+
```

That is why I showed you all these variations and so with the tag library, you can generate these tables very easily, whereas with JSP generating those tables with the sufficient pain that I just stop after parameters and attributes and did not really bother to do the JSTL style tables with http and all that but I could have use the same kind of code as should be obvious to you, okay, now let us consolidate a little bit.

(Refer Slide Time: 31:40)



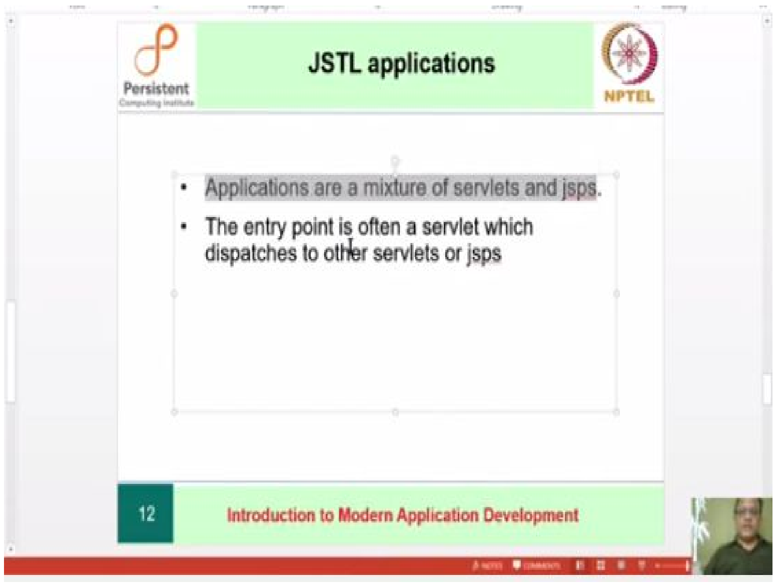
JSP Key ideas

- Instead of Code generating Text
  - HelloServlet: explicit output
  - HelloTem: replacing placeholders
- Have Code exist within text
- The "View" part is made explicitly into a JSP file
- JSP is HTML interspersed with Java Code
- JSP's are compiled into Servlets
- JSP's can cooperate with Servlets

8 Introduction to Modern Application Development

So, what we have now is key ideas instead of code generating text, code exists within text, this works okay but you get these problems where you have for anything more than beyond simple inclusion in the Java code and instead you use tag libraries to have simpler methods and then the tag themselves get compiled into servlets, the most important thing is this is simple automation of what could I have been done by hand, but would have been erroneous or error prone in tedious.

(Refer Slide Time: 32:17)



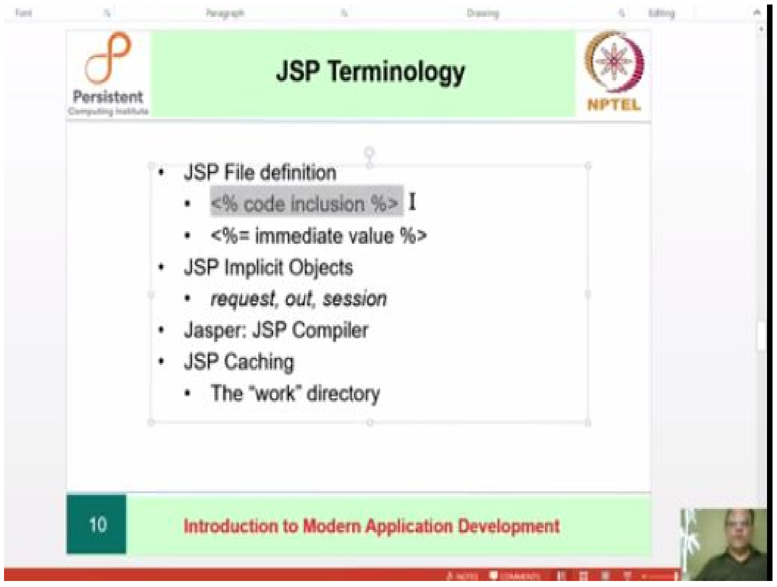
JSTL applications

- Applications are a mixture of servlets and jsps.
- The entry point is often a servlet which dispatches to other servlets or jsps

12 Introduction to Modern Application Development

And as we saw applications or a mixture of servlets in JSP's, the entry point can; the very first thing is can is often in JSP but after that you might go into a servlet which dispatches off to other JSP's. so, we have 2 JSP's, there is the entry point JSP and a JSP which is generated by servlet for our simple application.

(Refer Slide Time: 32:44)



JSP Terminology

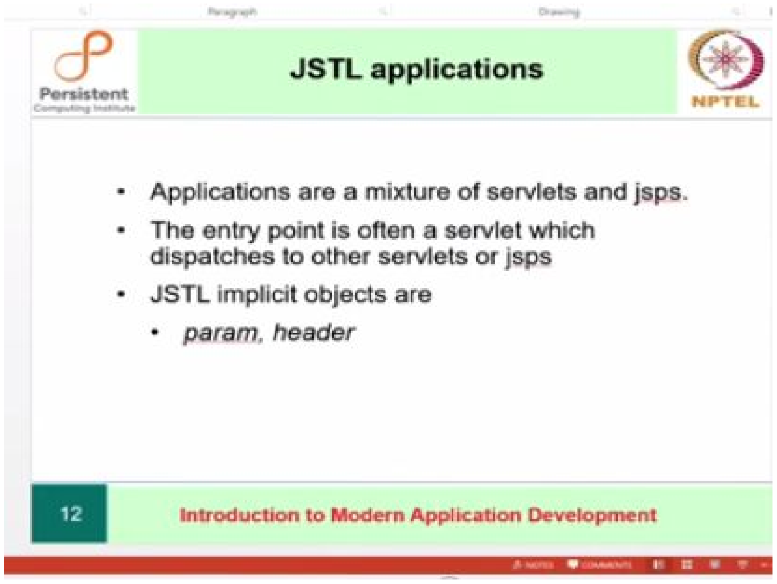
- JSP File definition
  - <% code inclusion %>
  - <%= immediate value %>
- JSP Implicit Objects
  - *request, out, session*
- Jasper: JSP Compiler
- JSP Caching
  - The "work" directory

10 Introduction to Modern Application Development

And as far as this JSP is concerned you have either code inclusion or values or the complicated version which is tag libraries which happened to involve syntax like C colon etc., whereby the way to remind you again, the C part at least was our choice where we said we are going to use this prefix for all the things like for each and all which are built-in into the JSTL library. Now, the exact syntax and all the details people have written in books on these things, the core idea I just want you to keep in mind is this is basic string jugglery, no more no less.

But you need access to some data and so you have to work with Java objects in order to get that access. What happens then is certain objects become implicitly available such as request out sessions and for the JSTL, there are implicit objects.

(Refer Slide Time: 33:49)



JSTL applications

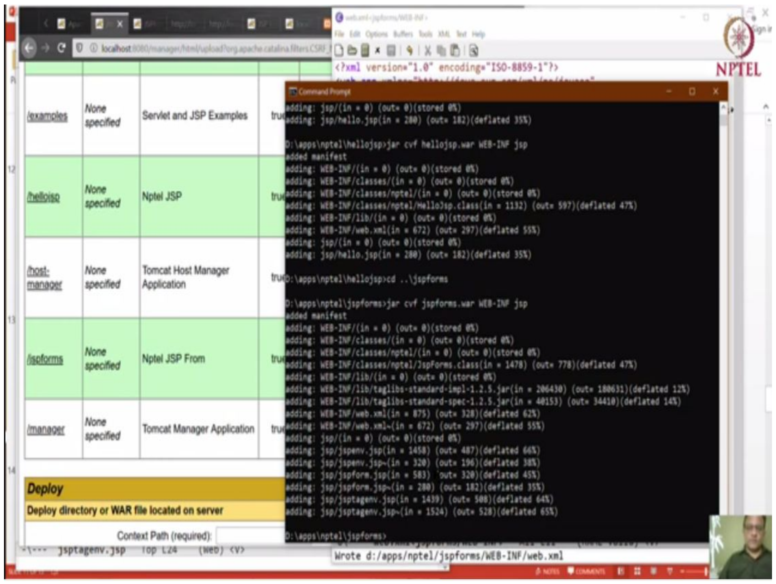
- Applications are a mixture of servlets and jsps.
- The entry point is often a servlet which dispatches to other servlets or jsps
- JSTL implicit objects are
  - *param, header*

12 Introduction to Modern Application Development

So, **JSTL implicit objects are params, headers**, which is the collection of objects available in JSTL. Jasper is a JSP compiler and the compiled version is cached by putting it into the work directory.

Summary: We know how to make forms using JSP and smartly achieve the same using JSTL.

(Refer Slide Time: 34:54)



/examples	None specified	Servlet and JSP Examples	true
/hellojsp	None specified	Nptel JSP	true
/host-manager	None specified	Tomcat Host Manager Application	true
/jspforms	None specified	Nptel JSP From	true
/manager	None specified	Tomcat Manager Application	true

Deploy

Deploy directory or WAR file located on server

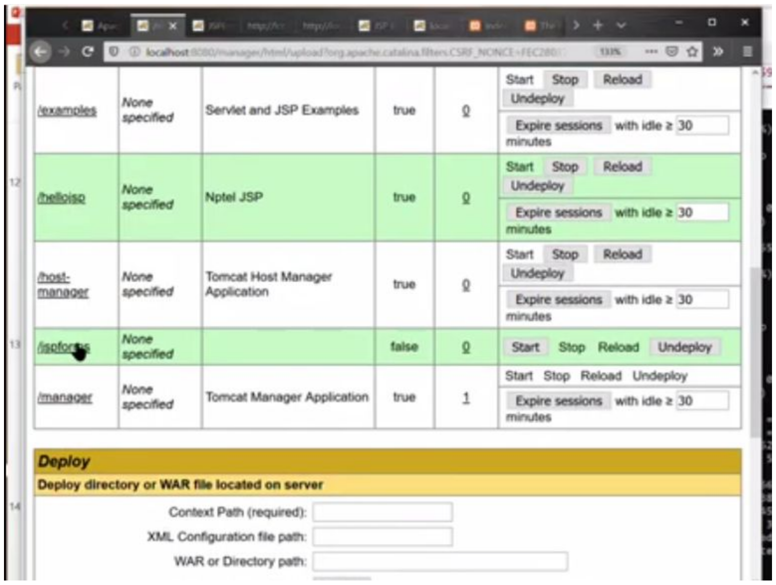
Context Path (required):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
adding: jsp/(in = 0) (out= 0)(stored 0%)
adding: jsp/hello.jsp(in = 280) (out= 182)(deflated 35%)

D:\apps\nptel\hellojsp>jar cvf hellojsp.war WEB-INF jsp
added manifest
adding: WEB-INF/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/nptel/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/nptel/HelloJsp.class(in = 1132) (out= 597)(deflated 47%)
adding: WEB-INF/lib/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/web.xml(in = 672) (out= 297)(deflated 55%)
adding: jsp/(in = 0) (out= 0)(stored 0%)
adding: jsp/hello.jsp(in = 280) (out= 182)(deflated 35%)

D:\apps\nptel\hellojsp>cd ..\jspforms

D:\apps\nptel\jspforms>jar cvf jspforms.war WEB-INF jsp
added manifest
adding: WEB-INF/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/nptel/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/nptel/JspForms.class(in = 1478) (out= 778)(deflated 47%)
adding: WEB-INF/lib/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/lib/taglibs-standard-impl-1.2.5.jar(in = 206430) (out= 180631)(deflated 12%)
adding: WEB-INF/lib/taglibs-standard-spec-1.2.5.jar(in = 40153) (out= 34410)(deflated 14%)
adding: WEB-INF/web.xml(in = 875) (out= 328)(deflated 62%)
adding: WEB-INF/web.xml~(in = 672) (out= 297)(deflated 55%)
adding: jsp/(in = 0) (out= 0)(stored 0%)
adding: jsp/jspenv.jsp(in = 1458) (out= 487)(deflated 66%)
adding: jsp/jspenv.jsp~(in = 320) (out= 196)(deflated 38%)
adding: jsp/jspform.jsp(in = 583) (out= 320)(deflated 45%)
adding: jsp/jspform.jsp~(in = 280) (out= 182)(deflated 35%)
adding: jsp/jsptagenv.jsp(in = 1439) (out= 508)(deflated 64%)
adding: jsp/jsptagenv.jsp~(in = 1524) (out= 528)(deflated 65%)

D:\apps\nptel\jspforms>
```

```
Wrote d:/apps/nptel/jspforms/WEB-INF/web.xml
```

(Refer Slide Time: 36:39)



/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/hellojsp	None specified	Nptel JSP	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/jspforms	None specified		false	0	Start Stop Reload Undeploy
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

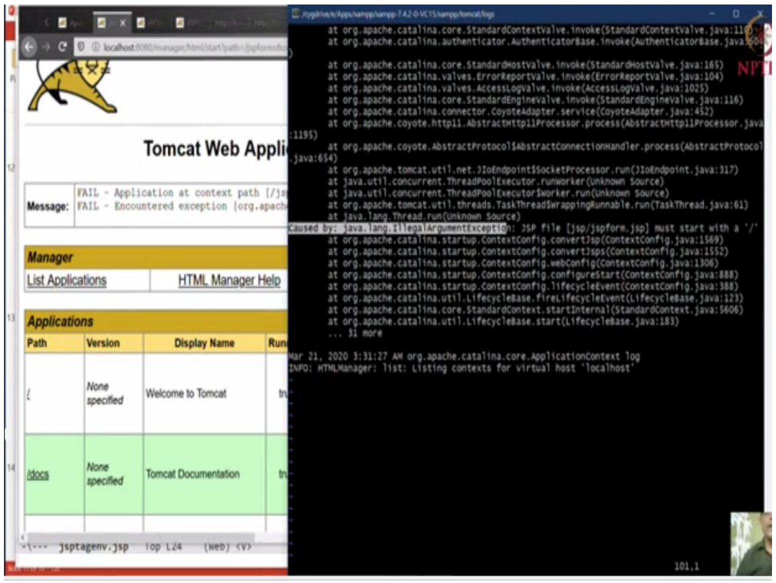
Deploy directory or WAR file located on server

Context Path (required):

XML Configuration file path:

WAR or Directory path:

(Refer Slide Time: 37:16)



Tomcat Web Appli

Message:	FAIL - Application at context path [/js FAIL - Encountered exception [org.apach

Manager

List Applications	HTML Manager Help

Applications

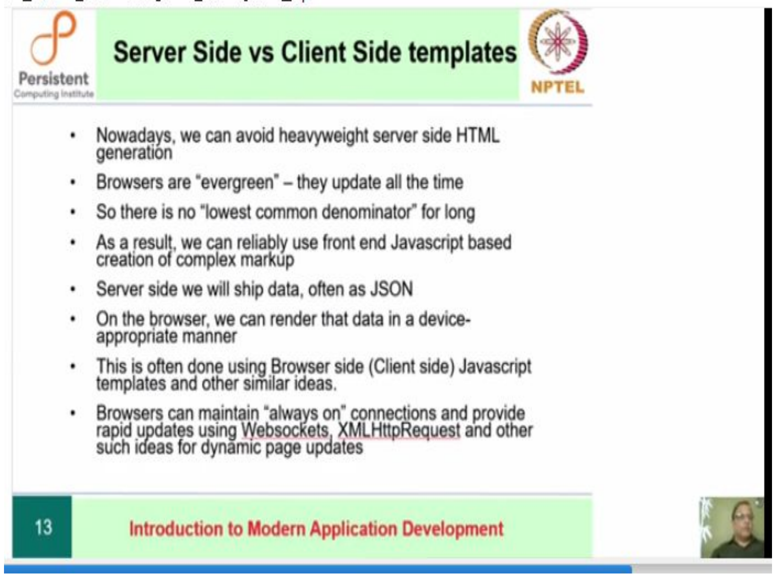
Path	Version	Display Name	Run
/	None specified	Welcome to Tomcat	tr
/docs	None specified	Tomcat Documentation	tr

```
-\---  jsptagenv.jsp   Top L24    (Web) <V>
```

```
        at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:11
        at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:
)
        at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:165)
        at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:104)
        at org.apache.catalina.valves.AccessLogValve.invoke(AccessLogValve.java:1025)
        at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:116)
        at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:452)
        at org.apache.coyote.http11.AbstractHttp11Processor.process(AbstractHttp11Processor.java
:1195)
        at org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.process(AbstractProtocol
.java:654)
        at org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run(JIoEndpoint.java:317)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
        at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
        at java.lang.Thread.run(Unknown Source)
Caused by: java.lang.IllegalArgumentException: JSP file [jsp/jspform.jsp] must start with a '/'
        at org.apache.catalina.startup.ContextConfig.convertJsp(ContextConfig.java:1569)
        at org.apache.catalina.startup.ContextConfig.convertJsps(ContextConfig.java:1552)
        at org.apache.catalina.startup.ContextConfig.webConfig(ContextConfig.java:1306)
        at org.apache.catalina.startup.ContextConfig.configureStart(ContextConfig.java:888)
        at org.apache.catalina.startup.ContextConfig.lifecycleEvent(ContextConfig.java:388)
        at org.apache.catalina.util.LifecycleBase.fireLifecycleEvent(LifecycleBase.java:123)
        at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5606)
        at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
        ... 31 more

Mar 21, 2020 3:31:27 AM org.apache.catalina.core.ApplicationContext log
INFO: HTMLManager: list: Listing contexts for virtual host 'localhost'
```

NOTE: If you observe the above error, go to Tomcat logs, it says very clearly that jsp/jspform must start with the slash.

(Refer Slide Time: 43:22)

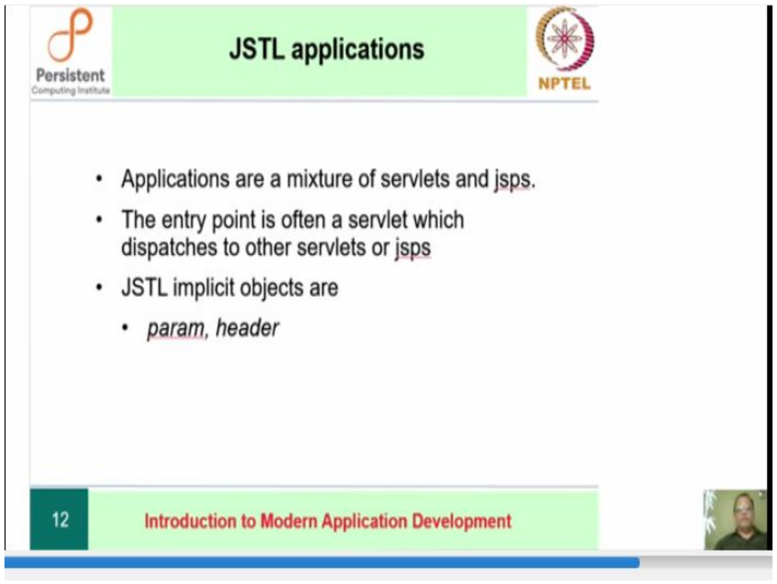


Server Side vs Client Side templates

- Nowadays, we can avoid heavyweight server side HTML generation
- Browsers are "evergreen" – they update all the time
- So there is no "lowest common denominator" for long
- As a result, we can reliably use front end Javascript based creation of complex markup
- Server side we will ship data, often as JSON
- On the browser, we can render that data in a device-appropriate manner
- This is often done using Browser side (Client side) Javascript templates and other similar ideas.
- Browsers can maintain "always on" connections and provide rapid updates using Websockets, XMLHttpRequest and other such ideas for dynamic page updates

13 Introduction to Modern Application Development

(Refer Slide Time: 43:31)



JSTL applications

- Applications are a mixture of servlets and jsps.
- The entry point is often a servlet which dispatches to other servlets or jsps
- JSTL implicit objects are
  - *param, header*

12 Introduction to Modern Application Development

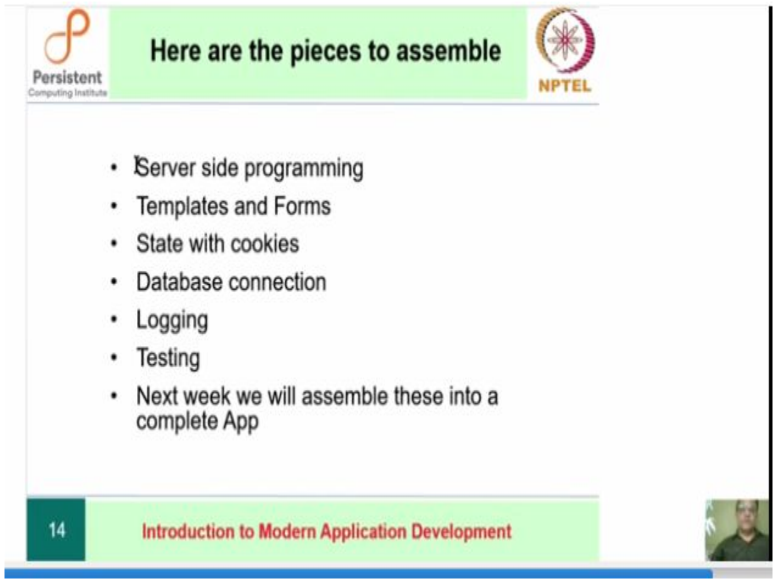
JSP TAKEAWAY:

There is basic JSP and there are tag libraries and both of them are essentially ways of generating web pages which are dominated by mark up.

But by and large for most of the sleek consumer sites you avoid heavy weight html server generation, back end these kinds of things were generated on servers, one reason was that browsers was were still evolving and kind of flaky and it took time for people to update these things but for about the last decade or more may be the last 12, 13 years browsers have become self-updating, they are updated very, very quickly and reliably.

And so we say these are evergreen browsers and so most of them when they acquire a capability, all of them copy each other and acquire the capability very, very fast. So, as a result you can reliably use front and JavaScript based mark up and on the server side, what you do is you shift data often as JSON and on the browser, so JSON is this data format which comes with JavaScript.

(Refer Slide Time: 45:50)



Here are the pieces to assemble

- Server side programming
- Templates and Forms
- State with cookies
- Database connection
- Logging
- Testing
- Next week we will assemble these into a complete App

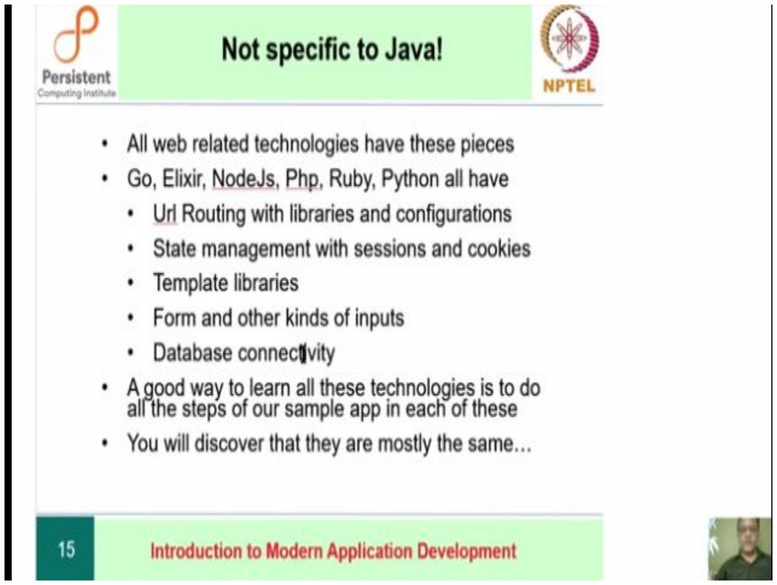
14 Introduction to Modern Application Development

TO DO:

Finished server side programming, templates and forms, state with cookies, database connections, logging.

1. Testing using embedded Tomcat
2. Assemble all the above into a complete App.

(Refer Slide Time: 46:16)



Not specific to Java!

- All web related technologies have these pieces
- Go, Elixir, NodeJs, Php, Ruby, Python all have
  - Url Routing with libraries and configurations
  - State management with sessions and cookies
  - Template libraries
  - Form and other kinds of inputs
  - Database connectivity
- A good way to learn all these technologies is to do all the steps of our sample app in each of these
- You will discover that they are mostly the same…

15 Introduction to Modern Application Development

The important thing to remember is, these problems are universal to web applications, they are not Java specific, so all the ultra-modern things like Go, and Elixir and even so not the ultra-modern things like NodeJs and Php, Ruby, Python etc., all of them have URL Routing with libraries and configuration, state management with sessions and cookies, template libraries, forms and other kinds of input, database connectivity and so on.

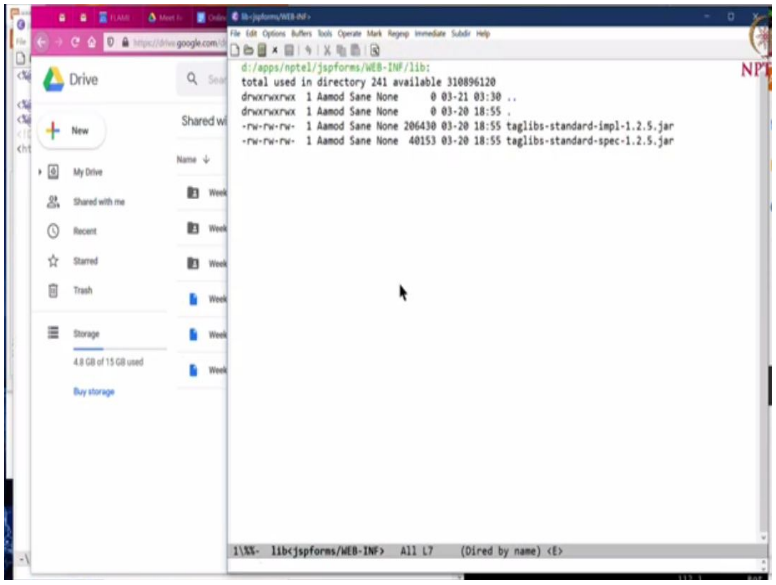
A good way to learn all these things is basically to take previous sample apps especially, hello with servlets to hello with jsp and jspforms and implement them in each of these technologies, we will find out that things and that different actually, there is no clear cut superiority over Java except perhaps Java tends to verbose and others are often perceived or actually are friendlier.

One important difference between the Java technologies and some of the other ones like Ruby especially, Ruby on the rails and some of its what shall we say; cousins in the Java world, like Spring Boot etc., is that they remove a lot of the complexity of detailed configuration of the kind we saw in web xml etc., by means of what is called convention over configuration. So, if you agree to design your app in a way that is prescribed by the designer of the framework, then you get a lot of these configuration type things for free.

And in practice it means a great deal, so there is real value in following the prescriptions of a framework, some of the further developments along these lines are newer systems that try to do even more wrapping and what this wrapping means we will eventually discussed in a lecture where we will talk about what is called the rest architecture of the web or representational state transfer.

And at that point, we will be able to see a much higher level view of the web, right now we are in the weeds with all the details and the tag libraries and so on and so forth but by and large, once you get used to it and once you are familiar with one set of these things, it is not really that hard, just tedious and annoying and you have to keep track of or great many details.

(Refer Slide Time: 49:08)



```
d:/apps/nptel/jspforms/WEB-INF/lib:
total used in directory 241 available 310896120
drwxrwxrwx  1 Aamod Sane None      0 03-21 03:30 ..
drwxrwxrwx  1 Aamod Sane None      0 03-20 18:55 .
-rw-rw-rw-  1 Aamod Sane None 206430 03-20 18:55 taglibs-standard-impl-1.2.5.jar
-rw-rw-rw-  1 Aamod Sane None  40153 03-20 18:55 taglibs-standard-spec-1.2.5.jar
```

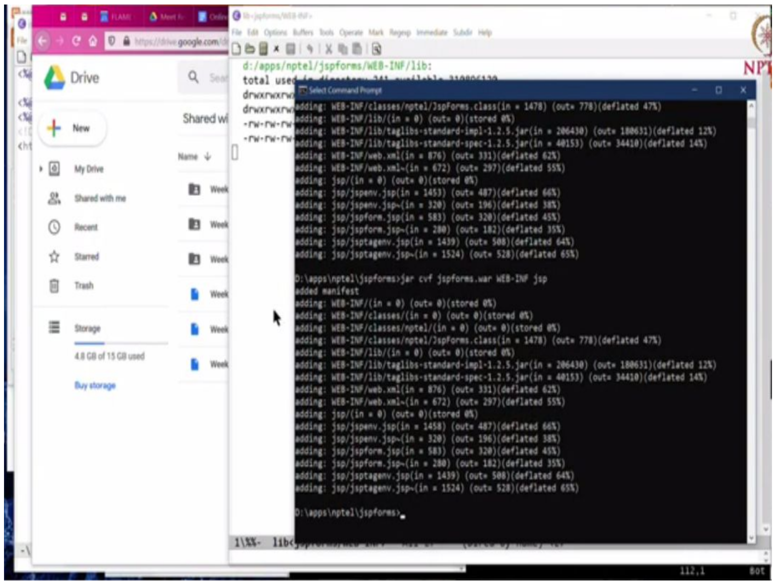
```
1\%%-  lib<jspforms/WEB-INF>   All L7     (Dired by name) <E>
```

Important difference between the JSP versus the JSTL:

Tere is just the JSP syntax with percentages versus the jsptagenv which mentions tag library. Now, JSP is in some sense built into Tomcat in the sense that the JSP processing is for free.

For the tag lib, we need to import certain libraries which are there in the examples that come with tomcat (look up in the examples directory) but these are the tag library, so this is called taglibs,impl and taglibs spec, these libraries you have to move into the lib's directory over here, WEB-INF.

(Refer Slide Time: 50:42)

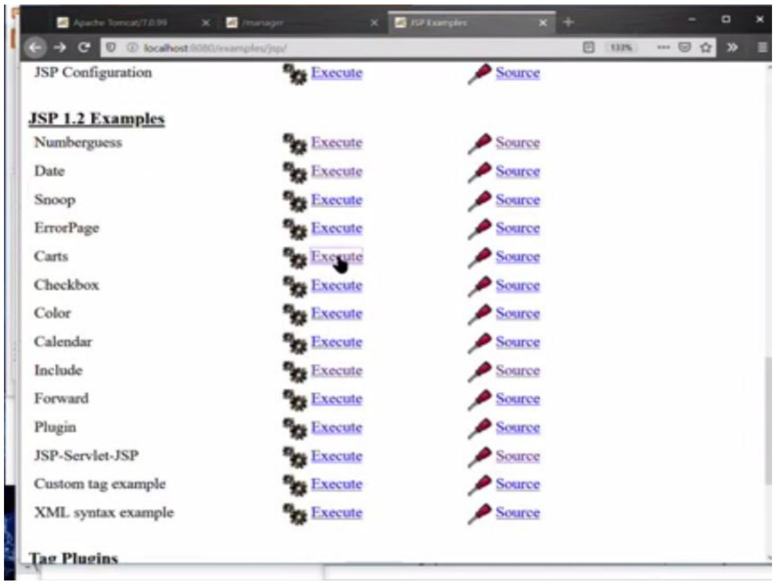


```
d:/apps/nptel/jspforms/WEB-INF/lib:
total used
```

```
adding: WEB-INF/classes/nptel/JspForms.class(in = 1478) (out= 778)(deflated 47%)
adding: WEB-INF/lib/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/lib/taglibs-standard-impl-1.2.5.jar(in = 206430) (out= 180631)(deflated 12%)
adding: WEB-INF/lib/taglibs-standard-spec-1.2.5.jar(in = 40153) (out= 34410)(deflated 14%)
adding: WEB-INF/web.xml(in = 876) (out= 331)(deflated 62%)
adding: WEB-INF/web.xml~(in = 672) (out= 297)(deflated 55%)
adding: jsp/(in = 0) (out= 0)(stored 0%)
adding: jsp/jspenv.jsp(in = 1453) (out= 487)(deflated 66%)
adding: jsp/jspenv.jsp~(in = 320) (out= 196)(deflated 38%)
adding: jsp/jspform.jsp(in = 583) (out= 320)(deflated 45%)
adding: jsp/jspform.jsp~(in = 280) (out= 182)(deflated 35%)
adding: jsp/jsptagenv.jsp(in = 1439) (out= 508)(deflated 64%)
adding: jsp/jsptagenv.jsp~(in = 1524) (out= 528)(deflated 65%)

D:\apps\nptel\jspforms>jar cvf jspforms.war WEB-INF jsp
added manifest
adding: WEB-INF/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/nptel/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/nptel/JspForms.class(in = 1478) (out= 778)(deflated 47%)
adding: WEB-INF/lib/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/lib/taglibs-standard-impl-1.2.5.jar(in = 206430) (out= 180631)(deflated 12%)
adding: WEB-INF/lib/taglibs-standard-spec-1.2.5.jar(in = 40153) (out= 34410)(deflated 14%)
adding: WEB-INF/web.xml(in = 876) (out= 331)(deflated 62%)
adding: WEB-INF/web.xml~(in = 672) (out= 297)(deflated 55%)
adding: jsp/(in = 0) (out= 0)(stored 0%)
adding: jsp/jspenv.jsp(in = 1458) (out= 487)(deflated 66%)
adding: jsp/jspenv.jsp~(in = 320) (out= 196)(deflated 38%)
adding: jsp/jspform.jsp(in = 583) (out= 320)(deflated 45%)
adding: jsp/jspform.jsp~(in = 280) (out= 182)(deflated 35%)
adding: jsp/jsptagenv.jsp(in = 1439) (out= 508)(deflated 64%)
adding: jsp/jsptagenv.jsp~(in = 1524) (out= 528)(deflated 65%)

D:\apps\nptel\jspforms>
```

And when you create the war file here, WEB-INF also contains the lib, so lib only shows up in this case if you compare with any of the earlier ones such as here as you will see, lib is empty and this is why we have lib for the few cases where a certain standard jar files must be included, so in order to get this kind of code working, you will need to include the tag lib's in the lib directory.

(Refer Slide Time: 51:36)



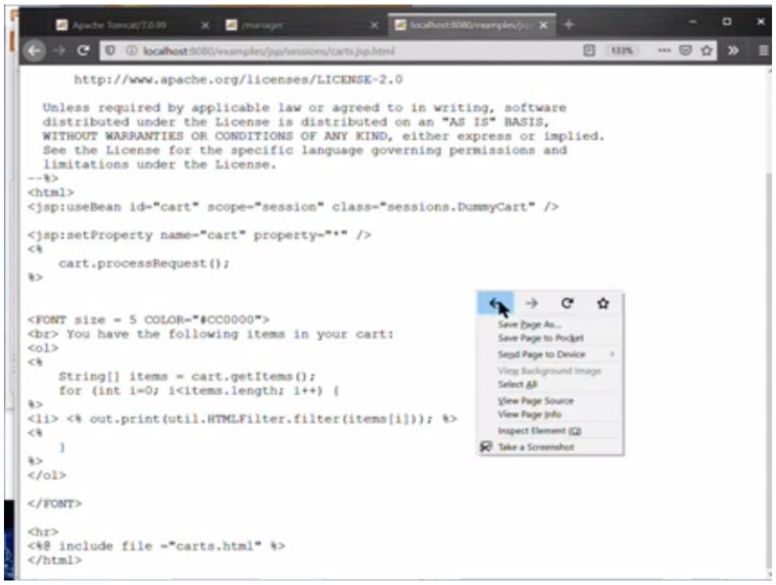
JSP Configuration	Execute	Source

JSP 1.2 Examples

Numberguess	Execute	Source
Date	Execute	Source
Snoop	Execute	Source
ErrorPage	Execute	Source
Carts	Execute	Source
Checkbox	Execute	Source
Color	Execute	Source
Calendar	Execute	Source
Include	Execute	Source
Forward	Execute	Source
Plugin	Execute	Source
JSP-Servlet-JSP	Execute	Source
Custom tag example	Execute	Source
XML syntax example	Execute	Source

Tag Plugins

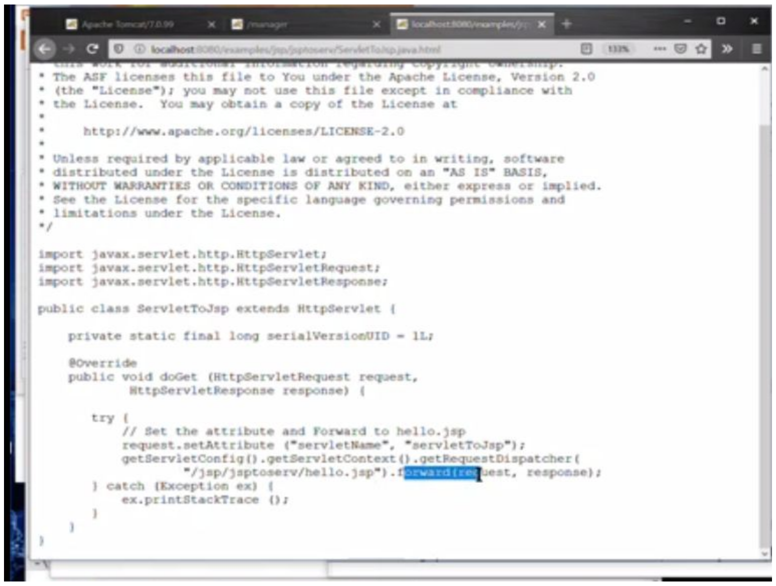
(Refer Slide Time: 52:19)



```
    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
--%>
<html>
<jsp:useBean id="cart" scope="session" class="sessions.DummyCart" />

<jsp:setProperty name="cart" property="*" />
<%
    cart.processRequest();
%>


<FONT size = 5 COLOR="#CC0000">
<br> You have the following items in your cart:
<ol>
<%
    String[] items = cart.getItems();
    for (int i=0; i<items.length; i++) {
%>
<li> <% out.print(util.HTMLFilter.filter(items[i])); %>
<%
    }
%>
</ol>

</FONT>

<hr>
<%@ include file ="carts.html" %>
</html>
```

And in their source, they have also shown how the cart itself is implemented with for loops in a different style explicitly, they do not use JSTL, it is JSP only but it is still interesting to see how these things are put together.

(Refer Slide Time: 52:45)

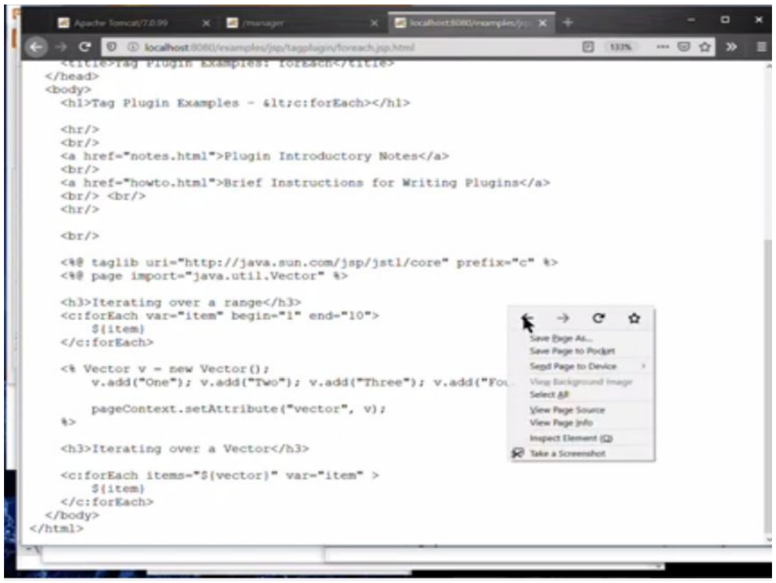


Apache Tomcat/7.0.99
/manager
localhost:8080/examples/jsp
localhost:8080/examples/jsp/jsptoserv/ServletToJsp.java.html
133%

```
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ServletToJsp extends HttpServlet {

    private static final long serialVersionUID = 1L;

    @Override
    public void doGet (HttpServletRequest request,
            HttpServletResponse response) {

       try {
           // Set the attribute and Forward to hello.jsp
           request.setAttribute ("servletName", "servletToJsp");
           getServletConfig().getServletContext().getRequestDispatcher(
                   "/jsp/jsptoserv/hello.jsp").forward(request, response);
       } catch (Exception ex) {
           ex.printStackTrace ();
       }
    }
}
```

Here is another example of JSP servlet JSP where they show how the JSP page can call the servlet and how the servlet can call JSP in turn using a different method, this one we will revisit later but I wanted to point out that here are some examples that you should take a look at.

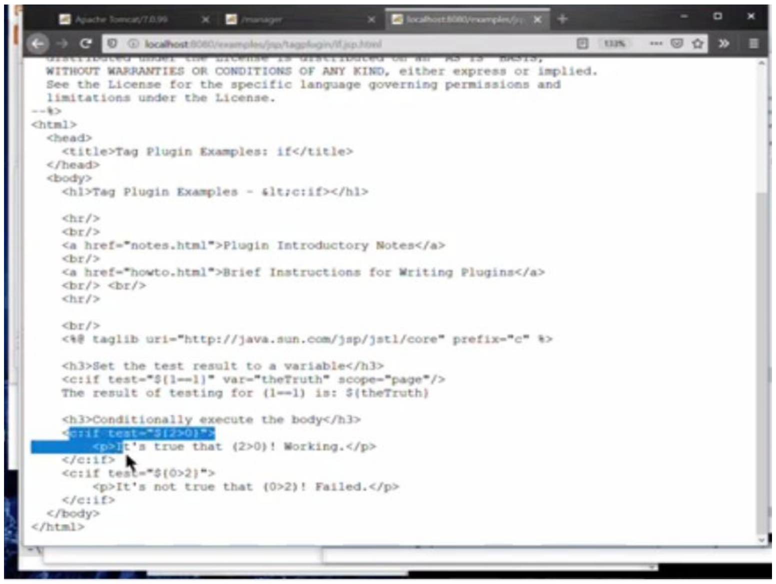
(Refer Slide Time: 53:13)



```
  </head>
  <body>
    <h1>Tag Plugin Examples - &lt;c:forEach></h1>

    <hr/>
    <br/>
    <a href="notes.html">Plugin Introductory Notes</a>
    <br/>
    <a href="howto.html">Brief Instructions for Writing Plugins</a>
    <br/> <br/>
    <hr/>

    <br/>

    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
    <%@ page import="java.util.Vector" %>

    <h3>Iterating over a range</h3>
    <c:forEach var="item" begin="1" end="10">
        ${item}
    </c:forEach>

    <% Vector v = new Vector();
        v.add("One"); v.add("Two"); v.add("Three"); v.add("Fo

        pageContext.setAttribute("vector", v);
    %>

    <h3>Iterating over a Vector</h3>

    <c:forEach items="${vector}" var="item" >
        ${item}
    </c:forEach>
  </body>
</html>
```

Similarly, there is for each and there is chose a bunch of other things, so here they show another way, where you use page contexts set attribute kind of the same way we did and you can loop over for loops like this, so those are some of the other examples about simple JSP things that you can find.

(Refer Slide Time: 53:39)



```
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
--%>
<html>
  <head>
    <title>Tag Plugin Examples: if</title>
  </head>
  <body>
    <h1>Tag Plugin Examples - &lt;c:if></h1>

    <hr/>
    <br/>
    <a href="notes.html">Plugin Introductory Notes</a>
    <br/>
    <a href="howto.html">Brief Instructions for Writing Plugins</a>
    <br/> <br/>
    <hr/>

    <br/>
    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

    <h3>Set the test result to a variable</h3>
    <c:if test="${1==1}" var="theTruth" scope="page"/>
    The result of testing for (1==1) is: ${theTruth}

    <h3>Conditionally execute the body</h3>
    <c:if test="${2>0}">
        <p>It's true that (2>0)! Working.</p>
    </c:if>
    <c:if test="${0>2}">
        <p>It's not true that (0>2)! Failed.</p>
    </c:if>
  </body>
</html>
```

Here is another one where some decision making is going on and so we have C colon if just like we have C colon for and there is syntax for testing things and so on, we may find these useful as we go ahead in the application.