

MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY

Banjara Hills, Hyderabad, Telangana



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Web Programming Laboratory Manual



Academic Year 2016-2017

Table of Contents

I Contents

1.	Vision of the Institution	i
2.	Mission of the Institution	i
3.	Department Vision	ii
4.	Department Mission	ii
5.	Programme Education Objectives	iii
6.	Programme Outcomes	iv
7.	Programme Specific Outcomes	v
8.	Introduction to Web Programming	vi

II Programs

1.	Creation of a static website of at least five pages using XHTML.	1
2.	Validation of static web page using Java script.	4
3.	Demonstration of XML, XSLT.	5
4.	Demonstration of XML, XSLT.	6
5.	Handling Sessions in web applications.	8
6.	Usage of Filters in web applications.	10
7.	Usage of Filters in web applications.	12
8.	Creation of dynamic content in web application using ASP.NET	17
9.	Providing data store support for web site using JDBC	21

Part I
Contents

1. Vision of the Institution

To be part of universal human quest for development and progress by contributing high calibre, ethical and socially responsible engineers who meet the global challenge of building modern society in harmony with nature.

2. Mission of the Institution

- To attain excellence in imparting technical education from undergraduate through doctorate levels by adopting coherent and judiciously coordinated curricular and co-curricular programs.
- To foster partnership with industry and government agencies through collaborative research and consultancy.
- To nurture and strengthen auxiliary soft skills for overall development and improved employability in a multi-cultural work space.
- To develop scientific temper and spirit of enquiry in order to harness the latent innovative talents.
- To develop constructive attitude in students towards the task of nation building and empower them to become future leaders
- To nourish the entrepreneurial instincts of the students and hone their business acumen.
- To involve the students and the faculty in solving local community problems through economical and sustainable solutions.

3. Department Vision

To contribute competent computer science professionals to the global talent pool to meet the constantly evolving societal needs.

4. Department Mission

Mentoring students towards a successful professional career in a global environment through quality education and soft skills in order to meet the evolving societal needs.

5. Programme Education Objectives

1. Graduates will demonstrate technical skills and leadership in their chosen fields of employment by solving real time problems using current techniques and tools.
2. Graduates will communicate effectively as individuals or team members and be successful in the local and global cross cultural working environment.
3. Graduates will demonstrate lifelong learning through continuing education and professional development.
4. Graduates will be successful in providing viable and sustainable solutions within societal, professional, environmental and ethical contexts

6. Programme Outcomes

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Lifelong learning:** Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

7. Programme Specific Outcomes

The graduates will be able to:

- PSO1:** Demonstrate understanding of the principles and working of the hardware and software aspects of computer systems.
- PSO2:** Use professional engineering practices, strategies and tactics for the development, operation and maintenance of software
- PSO3:** Provide effective and efficient real time solutions using acquired knowledge in various domains.

8. Introduction to Web Programming

Web development is a broad term for the work involved in developing a web site for the Internet (World Wide Web) or an intranet (a private network). Web development can range from developing the simplest static single page of plain text to the most complex web-based internet applications, electronic businesses, and social network services. A more comprehensive list of tasks to which web development commonly refers, may include web design, web content development, client liaison, client-side/server-side scripting, and data base access. Among web professionals, "web development" usually refers to the main non-design aspects of building web sites: writing markup and coding.

In this Web programming lab you'll learn how to:

- Create your own Web site using HTML.
- Program the behavior of web pages using client side JavaScript.
- Store and transport data using XML.
- Leverage the power of Java to create dynamic websites
- Deploy your applications for free on the Internet
- Use and understand core server-side Java and .NET web technologies

You will how to take your basic Java knowledge and use it to create websites using the same technologies (HTML, Javascript, XML, servlets, JSPs and JDBC) that everyone from self-employed web developers to huge corporations use to create modern interactive web sites.

Minimum Hardware Requirements

- 1 Ghz Pentium IV
- 1 GB RAM
- 8 GB Disk space
- Connection to the Internet. Note, some of the lab exercises may not work without a working Internet connection.

Minimum Software Requirements

- Microsoft Windows 2000 SP4 or Windows XP
- Internet Explorer 6.0
- WinZip 8.0 or higher
- Apache Tomcat 6 _ 0 _ 26
- Eclipse _JEE _Galileo _SR1
- JDK 6 0 Update 10
- Visual Studio 2005
- DataBase: Mysql, oracle etc.

Part II
Programs

Web Programming Lab Manual

Program 1

Creation of static web site using XHTML.

Problem Definition

Creation of a static website of at least five pages using XHTML.

Problem Description: -

A Static web site using only XHTML is to be created. The pages can resemble any popular website:

The website should consists the following pages.

- Home page (user formatting Tags)
- Registration and user Login (demonstrate Forms)
- Books catalog (demonstrate lists)
- Shopping cart (demonstrate tables)

Procedure:-

1. Creating the first page.

- Open a new file using Notepad.
- Name the file as XXXX.html (Extension for html programs is .html)
- Make the website neat and attractive with relevant text and pictures.
- Use suitable tags wherever necessary.
- Open 'Main page 'in browser to see the result.
- Similarly create the remaining pages.

2. Creating Table using HTML tags

- The basic structure of an HTML table is constructed using the following tags:

Table tags: <TABLE> </TABLE>

Row tags: <TR> </TR>

Cell tags: <TD> </TD>

Add border, title, and column headings

Web Programming Lab Manual

- Commands that enable you to customize your table include: The WIDTH=n% command sets the width of your table as a percentage of the screen.

The CELLPADDING=n command adjusts the vertical dimension of the cells.

The letter n designates the numerical value that you assign to this command.

The CELLSPACING=n command sets the space or border around the cells.

The ALIGN=(LEFT, RIGHT, or CENTER) command will horizontally align the data in a cell.

For The VALIGN=(TOP, MIDDLE, or BOTTOM) command will vertically align the data in a cell.

3. Creating List using HTML tags HTML provides three different types to choose from: unordered, ordered, and description lists.

- Creating an unordered list in HTML is accomplished using the unordered list block-level element, . Each item within an unordered list is individually marked up using the list item element, .
- Creating an ordered list in HTML is accomplished using the ordered list block-level element, . Each item within an unordered list is individually marked up using the list item element,
- Creating a description list in HTML is accomplished using the description list block-level element, <dl>. Instead of using a element to mark up list items, the description list requires two block-level elements: the description term element, <dt>, and the description element, <dd>.

4. Creating form using HTML tags Forms are an essential part of the Internet, as they provide a way for websites to capture information from users and to process requests

- To add a form to a page, we'll use the <form> element. The <form> element identifies where on the page control elements will appear. Additionally, the <form> element will wrap all of the elements included within the form, much like a <div> element.
- **Text Fields:** To obtain text from users is the <input> element. The <input> element uses the type attribute to define what type of information is to be captured within the control. The most popular type attribute value is text, which denotes a single line of text input. The name attribute value is used as the name of the control and is submitted along with the input data to the server.
- **Textarea:** Another element that's used to capture text-based data is the <textarea> element. The <textarea> element differs from the <input> element in that it can accept larger passages of text spanning multiple lines, the name attribute is used same as in text fields.

Web Programming Lab Manual

- **Radio Buttons:** Radio buttons are an easy way to allow users to make a quick choice from a small list of options. Radio buttons permit users to select one option only, as opposed to multiple options.

Check Box: Check boxes are very similar to radio buttons. They use the same attributes and patterns, with the exception of checkbox as their type attribute value. The difference between the two is that check boxes allow users to select multiple values and tie them all to one control name, while radio buttons limit users to one value.

- Check boxes are very similar to radio buttons. They use the same attributes and patterns, with the exception of checkbox as their type attribute value. The difference between the two is that check boxes allow users to select multiple values and tie them all to one control name, while radio buttons limit users to one value.
- **Drop-Down Lists:** To create a drop-down list we'll use the `<select>` and `<option>` elements. The `<select>` element wraps all of the menu options, and each menu option is marked up using the `<option>` element.

The name attribute resides on the `<select>` element, and the value attribute resides on the `<option>` elements that are nested within the `<select>` element. The value attribute on each `<option>` element then corresponds to the name attribute on the `<select>` element. Each `<option>` element wraps the text (which is visible to users) of an individual option within the list.

Much like the checked Boolean attribute for radio buttons and check boxes, drop-down menus can use the selected Boolean attribute to preselect an option for users.

- **Form Buttons:** After a user inputs the requested information, buttons allow the user to put that information into action. Most commonly, a submit/reset input or submit button is used to process the data.

Users click the submit/reset button to process data after filling out a form. The submit/reset button is created using the `<input>` element with a type attribute value of submit/reset. The value attribute is used to specify the text that appears within the button.

- **Hidden Input:** To create a hidden input, you use the hidden value for the type attribute. Additionally, include the appropriate name and value attribute values.

Validation/Result

The Web pages must demonstrate Formatting, Forms, Lists, Tables, Anchor, image tags.

Web Programming Lab Manual
Program 2

Validation of static web page using Java script.

Problem Definition

Validation of static web page using Java script.

Problem Description: -

Create a form using and validate (empty fields, correctness of name, roll no, email password etc.:

Procedure:

Create and HTML form and write JavaScript code to

1. Check for non-empty fields
2. Check for all letters in name field
3. Check for all numbers in roll no Field
4. Check whether Email field contains only letters, numbers and @ , and at the right place
5. Date Validation
6. Phone No. Validation
7. Credit Card No. Validation
8. Password Validation etc.

Validation/Result:

The program must check if any required field is empty and when it is filled, then program must check whether the contents are correct or not.

Web Programming Lab Manual
Program 3

Demonstration of XML, XSLT.

Problem Definition

Demonstration of XML, XSLT.

Problem Description: -

Using XSL the XML data is queries and displayed in the web browser.

Procedure:-

1. Start with a Raw XML Document
2. Suppose we want to transform the following XML document ("cdcatalog.xml") into XHTML
3. Create an XSL Style Sheet
4. Then you create an XSL Style Sheet ("cdcatalog.xsl")
5. Link the XSL Style Sheet to the XML Document
6. Add the XSL style sheet reference to your XML document ("cdcatalog.xml"):

Validation/Result:

The data present in the XML file is presented on the web page using XSL.

Web Programming Lab Manual Program 4

Creation of dynamic content in web application using servlets.

Problem Definition

Creation of dynamic content in web application using servlets.

Problem Description: -

A dynamic web page is to be created using servlet technology.

Problem Explanation: -

A servlet is a small program that runs on a server. The term was coined in the context of the Java applet, a small program that is sent as a separate file along with a Web (HTML) page. Java applets, usually intended for running on a client, can result in such services as performing a calculation for a user or positioning an image based on user interaction.

Procedure:-

1. **Tomcat installation:** We will assume you have a Tomcat distribution. Otherwise download it from jakarta.apache.org.
 - unzip `jakarta-tomcat-x.y.z.zip`. This creates the installation in a directory `jakarta-tomcat- x.y.z` (e.g `jakarta-tomcat-3.2.1`).
 - Ensure you have set the environment variable `JAVA_HOME` to your Java2 *e.g. J2SDK1.2* installation directory.
 - Try executing `jakarta-tomcat x.y.z/bin/startup.bat`.
 - Browse to `localhost:8080` (you should see the `index.html`)
 - Browse to the servlet/JSP examples
 - Execute several examples (`localhost 8080/examples`) and study the source code (note: the Java source code is under `jakarta-tomcat-x.y.z/webapps/Web-inf/classes`)
 - Unzip also `jakarta-servletapi-x.y.zip`. This contains source + API documentation for the `javax.servlet` APIs (Servlet and JSP).

2. **Creating your work environment :**In this lab you will create your own servlet and JSP server environment.
 - create a working directory for your server in your exercises dir under `exercises /servlet-jsp/servlet/myserver`
 - create subdirs 'src' (for all Java source code except servlets), 'webapps' (your WWW root dir) and `webapps/Web-inf` (standard name for dir containing config files and servlet code)

Web Programming Lab Manual

- under webapps/Web-inf create a subdir called 'servlets' for the servlet code you will be writing
- backup the original jakarta-tomcat-x.y.z /conf /server.xml by copying it to server-orig.xml
- add your directory as a context in jakarta tomcat x.y.z/conf/server.xml around line 260 you should see a line with

”===== Special webapps =====”

place your 'myserver' context here similar to the other contexts (e.g. /examples). The easiest way is to copy the Context XML entry for /examples and to rename the attributes 'path' to 'myserver' and docBase to the working dir created in

step 1. An example Context is given below for the working dir d:/java/exercises/servletjsp/servlet/exercises/myserver/webapps.

```
<Context path="/myserver"
          docBase="d:/java/exercises/servlet-jsp/servlet/
exercises/myserver/webapps"
          crossContext="false"
          debug="0"
          reloadable="true" >
</Context>
```

- Run startup.bat (see previous lab)
- Browse to localhost:8080/myserver. You should see your webapps directory.

3. Creating your first servlet

- In this lab you will create your very first servlet. You will learn how to write and register a servlet with the server and how to bind it to a URL.
- Under webapps/Web-inf/classes create a file of java
- Create a file webapps/Web-inf/web.xml with the following lines This registers the servlet and binds it to the URL localhost:8080/myserver/itworks run startserver.bat browse to localhost:8080/myserver/itworks also browse to localhost:8080/myserver/servlet/itworks

Validation/Result:

A web page is created dynamically sent to the client.

Web Programming Lab Manual Program 5

Handling Sessions in web applications.

Problem Definition

Creation of dynamic content in web application using servlets.

Problem Description: -

Using some technique we have to make server remember us, that is identify us when the subsequent request is sent. HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request. So we need a mechanism to make server remember us (our information). In this experiment it will be demonstrated using HttpSession class.

Procedure: -

For Implementations of session tracking use any one of the following Session tracking methods:

1. User authorization
2. Hidden fields
3. URL rewriting
4. Using cookie
5. Using HttpSession object.

Here, instantiate HttpSession object to create the session

HttpSession Interface

You would get HttpSession object by calling the public method `getSession()` of `HttpServletRequest`, as below:

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

You need to call `request.getSession()` before you send any document content to the client. Here is a summary of the important methods available through HttpSession object which can be used to store and retrieve information:

public Object getAttribute(String name)

This method returns the object bound with the specified name in this session, or null if no object is bound under the name.

public Enumeration getAttributeNames()

This method returns an Enumeration of String objects containing the names of all the objects bound to this session.

public long getCreationTime()

This method returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.

public String getId()

This method returns a string containing the unique identifier assigned to this session.

public long getLastAccessedTime()

This method returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.

public int getMaxInactiveInterval()

This method returns the maximum time interval, in seconds that the servlet container will keep this session open between client accesses.

public void invalidate()

This method invalidates this session and unbinds any objects bound to it.

public boolean isNew()

This method returns true if the client does not yet know about the session or if the client chooses not to join the session.

public void removeAttribute(String name)

This method removes the object bound with the specified name from this session.

public void setAttribute(String name, Object value)

This method binds an object to this session, using the name specified.

public void setMaxInactiveInterval(int interval)

This method specifies the time, in seconds, between client requests before the servlet container will invalidate this session.

Validation/Result:-

Now running <http://localhost:8080/SessionTrack> would display the following result when you would run for the first time:

Web Programming Lab Manual
Program 6

Usage of Filters in web applications.

Problem Definition

Usage of Filters in web applications.

Problem Description: -

We need to create a filter which comes in between the user request/response and the recourse. Servlet Filters are Java classes that can be used in Servlet Programming for the following purposes:

- To intercept requests from a client before they access a resource at back end.
- To manipulate responses from server before they are sent back to the client.

There are various types of filters suggested by the specifications:

- Authentication Filters.
- Data compression Filters.
- Encryption Filters.
- Filters that trigger resource access events.
- Image Conversion Filters.
- Logging and Auditing Filters.
- MIME-TYPE Chain Filters.
- Tokenizing Filters.
- XSL/T Filters That Transform XML Content.

Procedure: -

1. Name the servlet filter and assign the corresponding implementation class to the servlet filter.
2. Optionally, assign initialization parameters that get passed to the init method of the servlet filter.
3. After configuring the servlet filter, the web.xml application deployment descriptor contains a servlet filter configuration similar to the following example:

```

<filter id="Filter_1">
    <filter-name>LoginFilter</filter-name>
    <filter-class>LoginFilter</filter-class>
    <description>Performs pre-login and post-login
operation</description>
    <init-param>// optional
        <param-name>ParameterName</param-name>
        <param-value>ParameterName</param-value>
    </init-param>
</filter>

```

1. Map the servlet filter to a URL or a servlet.
2. After mapping the servlet filter to a URL or a servlet, the web.xml application deployment descriptor contains servlet mapping similar to the following example:

```

<filter-mapping>
    <filter-name>LoginFilter</filter-name>
    <url-pattern>/j_security_check</url-pattern>
    // can be servlet <servlet>servletName</servlet>
</filter-mapping>

```

You can use servlet filters to replace the CustomLoginServlet servlet, and to perform additional authentication, auditing, and logging.

Validation/Result:-

The login page is due to filter.

Web Programming Lab Manual
Program 7

Creation of dynamic content in web application using JSP.

Problem Definition

Creation of dynamic content in web application using JSP.

Problem Description: -

Demonstrate various jsp elements and generate dynamic web pages. In JSP, java code can be written inside the jsp page using the scriptlet tag. Let's see what the scripting elements are first.

The scripting elements provide the ability to insert java code inside the jsp. There are three types of scripting elements:

- Scriptlet tag
- Expression tag
- Declaration tag

Procedure:

1. 1. **Expressions** of the form

```
<%= expression %>
```

that are evaluated and inserted into the output,

A JSP expression is used to insert Java values directly into the output. It has the following form:

```
<%= Java expression %>
```

The Java expression is evaluated, converted to a string, and inserted in the page. Current time:

```
<%= new java.util.Date() %>
```

predefined variables that you can use

request, the HttpServletRequest; response, the HttpServletResponse;

session, the HttpSession associated with the request (if any); and out, the PrintWriter (a buffered version of type JspWriter) used to send output to the client.

Example:

```
Your hostname: <%= request.getRemoteHost() %>
```

2. **Scriptlets** of the form

```
<% code %>
```

 that are inserted into the servlet's service method, and

If you want to do something more complex than insert a simple expression, JSP scriptlets insert code into the service() method of the generated servlet.

Scriptlets have the following form:

```
<% Java Code %>
```

Scriptlets have access to the same automatically defined variables as expressions

```
<%  
String name = request.getParameter("name");  
out.println("name: " + name);  
%>
```

3. **Declarations** of the form

```
<%! code %>
```

that are inserted into the body of the servlet class, outside of any existing methods. A JSP declaration define methods or fields that get inserted into the main body of the servlet class (outside of the service method processing the request). It has the following form:

```
<%! Java Code %>
```

```
or  
<%!
```

```
public int getValue(){  
return 78;  
}  
%>
```

The method we can use in Expression also.

```
<%=getValue()%>
```


JSP actions use constructs in XML syntax to control the behavior of the servlet engine. You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin. Available actions include:

- `jsp:include` - Include a file at the time the page is requested.
- `jsp:useBean` - Find or instantiate a JavaBean.
- `jsp:setProperty` - Set the property of a JavaBean.
- `jsp:getProperty` - Insert the property of a JavaBean into the output.
- `jsp:forward` - Forward the requester to a new page.
- `jsp:plugin` - Generate browser-specific code that makes an OBJECT or EMBED tag for the Java plugin.

JSP Directives

There are two types of directive:

page :

The page directive define one or more of the following case-sensitive attributes:

- **import="package.class"** or **import="package.class1,...,package.classN"**. This lets you specify what packages should be imported.

For example:

```
<%@ page import="java.util.*" %>
```

The import attribute is the only one that is allowed to appear multiple times.

- **contentType="MIME-Type"** or **contentType="MIME-Type; charset=Character-Set"**

This specifies the MIME type of the output. The default is `text/html`. For example, the directive

```
<%@ page contentType="text/plain" %>
```

has the same effect as the scriptlet

```
<% response.setContentType("text/plain"); %>
```

- **isThreadSafe="true—false"**. A value of `true` (the default) indicates normal servlet processing, where multiple requests can be processed simultaneously with a single servlet instance, under the assumption that the author synchronized access to instance variables. A value of `false` indicates that the servlet should implement `SingleThreadModel`, with requests either delivered serially or with simultaneous requests being given separate servlet instances.

Web Programming Lab Manual

- **session="true—false"**. A value of true (the default) indicates that the pre-defined variable session (of type HttpSession) should be bound to the existing session if one exists, otherwise a new session should be created and bound to it. A value of false indicates that no sessions will be used, and attempts to access the variable session will result in errors at the time the JSP page is translated into a servlet.
- **buffer="sizekb—none"**. This specifies the buffer size for the JspWriter out. The default is server-specific, but must be at least 8kb.
- **autoflush="true—false"**. A value of true, the default, indicates that the buffer should be flushed when it is full. A value of false, rarely used, indicates that an exception should be thrown when the buffer overflows. A value of false is illegal when also using buffer="none".
- **extends="package.class"**. This indicates the superclass of servlet that will be generated. Use this with extreme caution, since the server may be using a custom superclass already.
- **info= "message"**. This defines a string that can be retrieved via the getServletInfo method.
- **errorPage= "url"**. This specifies a JSP page that should process any Throwables thrown but not caught in the current page.
- **isErrorPage="true—false"**. This indicates whether or not the current page can act as the error page for another JSP page. The default is false.
- **language="java"**. At some point, this is intended to specify the underlying language being used. For now, don't bother with this since java is both the default and the only legal choice.
- **include :** This directive include files at the time the JSP page is translated into a servlet. The directive looks like this:
 - `<%@ include file="relative url" %>`
 - The URL specified is normally interpreted relative to the JSP page that refers to it, but, as with relative URLs in general, you can tell the system to interpret the URL relative to the home directory of the Web server by starting the URL with a forward slash. The contents of the included file are parsed as regular JSP , and thus can include static HTML, scripting elements, directives, and actions.
 - `<%@ include file="header.jsp" %>`

JavaServer Pages (JSP) separates the dynamic part of your pages from the static HTML. You simply write the regular HTML and then enclose the code for the dynamic parts in special tags, most of which start with

"<%>" and end with "%>".

There are five main types of JSP constructs that you embed in a page: Scriptlet, expression, declaration, directives, and actions.

Scripting elements (Scriptlet, expression, declaration) specify Java code that will become part of the resultant servlet, directives let you control the overall structure of the servlet, and actions let you specify existing components that should be used, and otherwise control the behavior of the JSP engine.

Example: test.jsp

```
<html>
<body>
<%
out.println("Hello This is your First JSP");
%>
</body>
</html>
```

This is your First JSP ready

In the browser you can <http://localhost:8080/test.jsp>

In the browser you can see "Hello This is your First JSP"

Validation/Result:-

Dynamic web pages using JSP (scriptlets, expressions) is created.

Web Programming Lab Manual
Program 8

Creation of dynamic content in web application using ASP.NET

Problem Definition

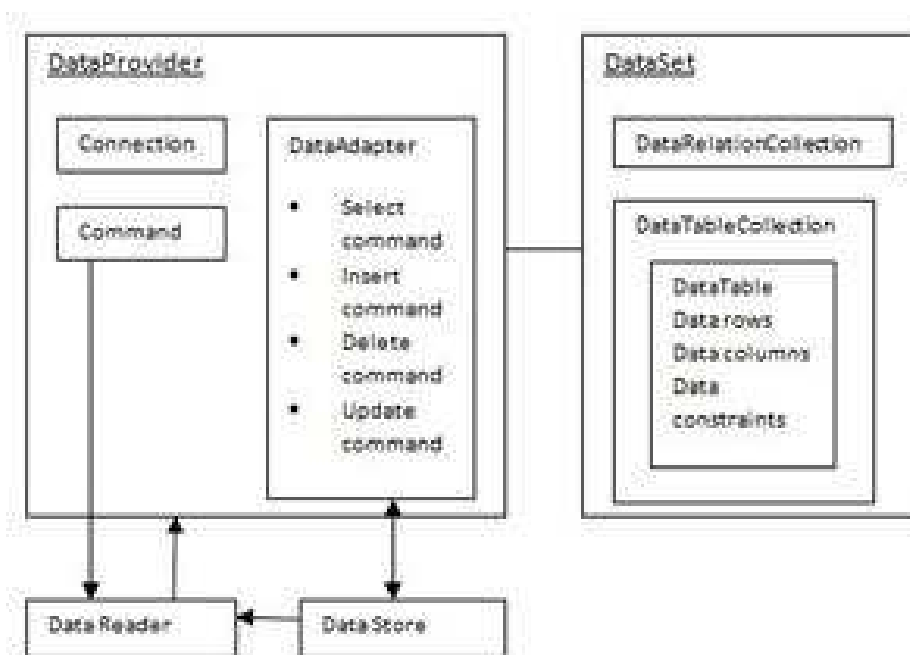
Creation of dynamic content in web application using JSP.

Problem Description: -

ASP.NET is a web application framework developed and marketed by Microsoft to allow programmers to build dynamic web sites. It allows you to use a full featured programming language such as C# or VB.NET to build web applications easily.

ADO.NET provides a bridge between the front end controls and the back end database. The ADO.NET objects encapsulate all the data access operations and the controls interact with these objects to display data, thus hiding the details of movement of data.

The following figure shows the ADO.NET objects at a glance:



Code:-

ASP .NET file

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeBehind="Default.aspx.cs"
    Inherits="createdatabase._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

    <head runat="server">
        <title>
            Untitled Page
        </title>
    </head>

    <body>
        <form id="form1" runat="server">

            <div>
                <asp:GridView ID="GridView1" runat="server">
                    </asp:GridView>
            </div>

        </form>
    </body>

</html>
```

C# file:

```
namespace createdatabase
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                DataSet ds = CreateDataSet();
                GridView1.DataSource = ds.Tables["Student"];
                GridView1.DataBind();
            }
        }
    }
}
```

```
private DataSet CreateDataSet()
{
    //creating a DataSet object for tables
    DataSet dataset = new DataSet();

    // creating the student table
    DataTable Students = CreateStudentTable();
    dataset.Tables.Add(Students);
    return dataset;
}

private DataTable CreateStudentTable()
{
    DataTable Students = new DataTable("Student");

    // adding columns
    AddNewColumn(Students, "System.Int32", "StudentID");
    AddNewColumn(Students, "System.String", "StudentName");
    AddNewColumn(Students, "System.String", "StudentCity");

    // adding rows
    AddNewRow(Students, 1, "M H Kabir", "Kolkata");
    AddNewRow(Students, 1, "Shreya Sharma", "Delhi");
    AddNewRow(Students, 1, "Rini Mukherjee", "Hyderabad");
    AddNewRow(Students, 1, "Sunil Dubey", "Bikaner");
    AddNewRow(Students, 1, "Rajat Mishra", "Patna");

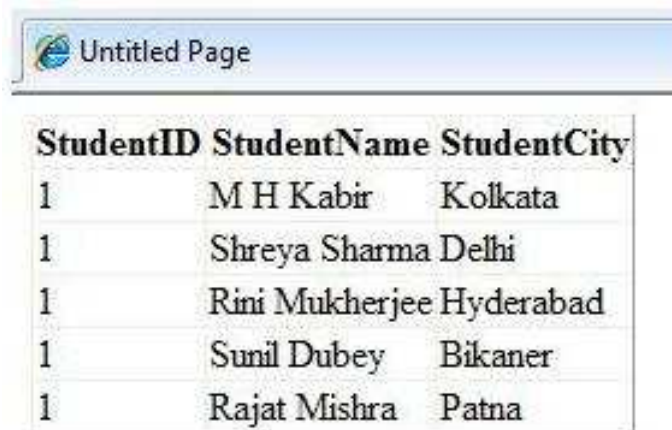
    return Students;
}

private void AddNewColumn(DataTable table,
    string columnType, string columnName)
{
    DataColumn column = table.Columns.Add(
        columnName,Type.GetType(columnType));
}

//adding data into the table
private void AddNewRow(
    DataTable table, int id, string name, string city)
{
    DataRow newrow = table.NewRow();
    newrow["StudentID"] = id;
    newrow["StudentName"] = name;
    newrow["StudentCity"] = city;
    table.Rows.Add(newrow);
}
```

```
}  
  }  
}
```

Validation/Result:-



The screenshot shows a web browser window with the title 'Untitled Page'. Inside the browser, there is a table with three columns: 'StudentID', 'StudentName', and 'StudentCity'. The table contains five rows of data:

StudentID	StudentName	StudentCity
1	M H Kabir	Kolkata
1	Shreya Sharma	Delhi
1	Rini Mukherjee	Hyderabad
1	Sunil Dubey	Bikaner
1	Rajat Mishra	Patna

Conclusion:-

A Output from ASP is generated dynamically after retrieving data form database.

Web Programming Lab Manual

Program 9

Providing data store support for web site using JDBC

Problem Definition

Providing data store support for web site using JDBC

Problem Description: -

This is a simple JSP program to connect to MSSQL database. This example JSP program shows how to connect to a MSSQL database from your JSP program. You also need to download the appropriate driver to connect to MSSQL server from your JSP page. In this tutorial we are using the JTDS driver which can be downloaded from <http://jtds.sourceforge.net/>, once you have downloaded the jar file you will have to copy it to your common lib folder in your tomcat (or any other servlet container you are using).

The database server can be residing anywhere in the network. You just need to get the IP address or the domain name of the server together with the database name, username and password.

Just remember to construct the right url. This sample JSP page assumes that there is a table named employees in your database and it has fields with

`names,cust_id,rdate and email.`

In your case you will have to change the names according to your requirement.

Prcedure/Code:-

1. Register the driver class The `forName()` method of `Class` class is used to register the driver class. This method is used to dynamically load the driver class.

Syntax of `forName()` method

```
public static void forName(String className)throws  
                        ClassNotFoundException
```

Example to register the `OracleDriver` class

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

2. Create the connection object

The `getConnection()` method of `DriverManager` class is used to establish connection with the database. Syntax of `getConnection()` method

- a. `public static Connection getConnection(String url)
throws SQLException`
- b. `public static Connection getConnection(String url,String name,
String password) throws SQLException`

Example to establish connection with the Oracle database

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

- c. Create the Statement object

The `createStatement()` method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Syntax of `createStatement()` method

```
public Statement createStatement()throws SQLException  
Example to create the statement object
```

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

- d. Execute the query

The `executeQuery()` 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 of `executeQuery()` method

```
public ResultSet executeQuery(String sql)throws SQLException  
Example to execute query
```

```
ResultSet rs=stmt.executeQuery("select * from emp");
```

```
while(rs.next()){  
System.out.println(rs.getInt(1)+" "+rs.getString(2));  
}
```

- e. Close the connection object

By closing connection object statement and ResultSet will be closed automatically. The `close()` method of Connection interface is used to close the connection.

Syntax of close() method

```
public void close()throws SQLException
```

Example to close connection

```
con.close();
```

f. Create sample Table in Database

To create the Employees table in EMP database, use the following steps:

Step 1:

Open a Command Prompt and change to the installation directory as follows:

```
C:\>
C:\>cd Program Files\MySQL\bin
C:\Program Files\MySQL\bin>
```

Step 2:

Login to database as follows

```
C:\Program Files\MySQL\bin>mysql -u root -p
Enter password: *****
mysql>
```

Step 3:

Create the table Employee in TEST database as follows:

```
mysql> use TEST;
mysql> create table Employees
(
    id int not null,
    age int not null,
    first varchar (255),
    last varchar (255)
);
```

Query OK, 0 rows affected (0.08 sec)

```
mysql;
```

Create Data Records

Finally you create few records in Employee table as follows:

```
mysql> INSERT INTO Employees VALUES (100, 18, 'Zara', 'Ali');  
Query OK, 1 row affected (0.05 sec)
```

```
mysql> INSERT INTO Employees VALUES (101, 25, 'Mahnaz', 'Fatma');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO Employees VALUES (102, 30, 'Zaid', 'Khan');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO Employees VALUES (103, 28, 'Sumit', 'Mittal');  
Query OK, 1 row affected (0.00 sec)
```

Validation/Result:-

The Database is accessed through a Java program using JDBC.

Annexure – I List of experiments according to the curriculum

CS 381 (Part) WEB PROGRAMMING LAB

Instruction	3	Periods per week
Duration of University Examination	3	Hours
University Examination	50	Marks
Sessional	25	Marks

Web Programming Experiments:

1. Creation of static web site using XHTML.
2. Demonstration of XML, XSLT.
3. Validation of static web page using Java script.
4. Creation of dynamic content in web application using servlets.
5. Handling Sessions in web applications.
6. Usage of Filters in web applications.
7. Creation of dynamic content in web application using JSP.
8. Creation of dynamic content in web application using ASP.NET
9. Providing data store support for web site using JDBC

References:

1. HTML & CSS: The Complete Reference, Fifth Edition Mar 2010 by Thomas Powell
Java Script for Absolute Beginners by Mcnavage T
2. PROFESSIONAL JAVA SERVER PROGRAMMING: J2EE 1.3 EDITION (APRESS series) by SUBRAHMANYAM ALLAMARAJU, CEDRIC BUEST, Edition : 2007
3. <http://www.w3schools.com/>