JSTL

JSP Standard Tag Libraries

What is JavaServer Pages?

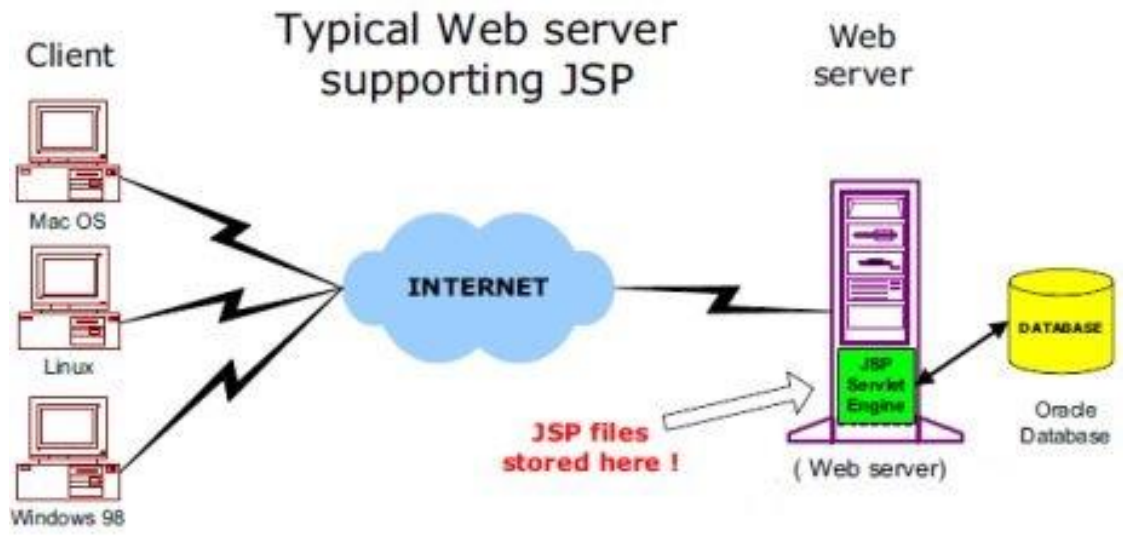
- JavaServer Pages (JSP) is a technology for developing web pages that support dynamic content which helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.
- A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.
- Using JSP, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.
- JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages and sharing information between requests, pages etc.

Advantages of JSP

- Following is the list of other advantages of using JSP over other technologies:
- **vs. Active Server Pages (ASP):** The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.
- **vs. Pure Servlets:** It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.
- **vs. Server-Side Includes (SSI):** SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.
- **vs. JavaScript:** JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.
- **vs. Static HTML:** Regular HTML, of course, cannot contain dynamic information.

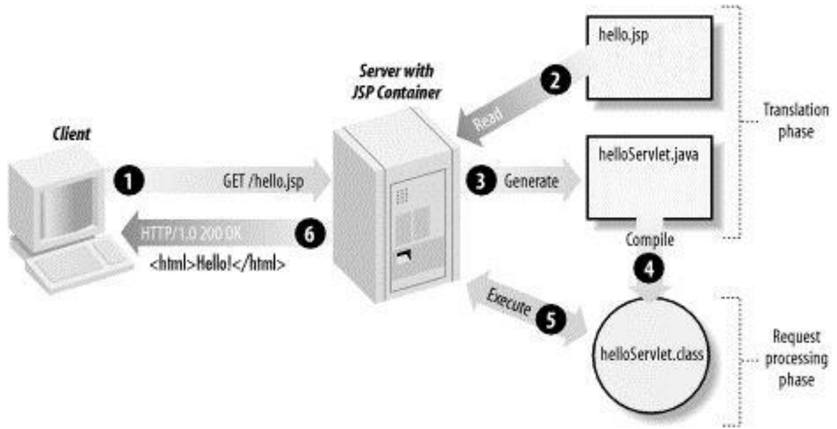
JSP Architecture

- The web server needs a JSP engine ie. container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages. This chapter makes use of Apache which has built-in JSP container to support JSP pages development.
- A JSP container works with the Web server to provide the runtime environment and other services a JSP needs. It knows how to understand the special elements that are part of JSPs.



JSP Processing

- The following steps explain how the web server creates the web page using JSP:
- As with a normal page, your browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with **.jsp** instead of .html.
- The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to println( ) statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior of the page.
- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format, which the servlet engine passes to the web server inside an HTTP response.
- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page.



Scriptlet

- A scriptlet can contain any number of JAVA language statements, variable or method declarations, or expressions that are valid in the page scripting language.
- Following is the syntax of Scriptlet:

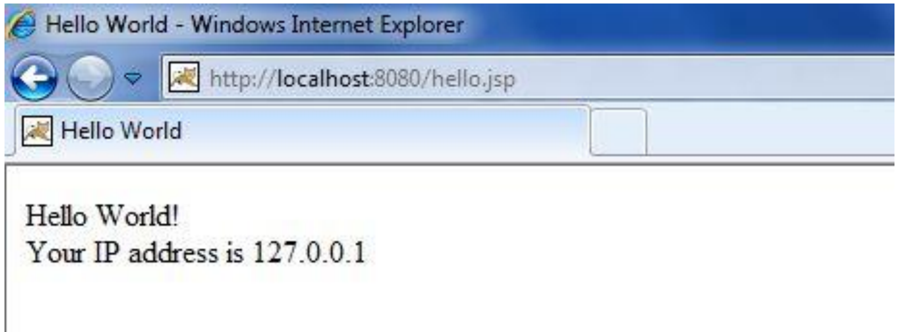
```
<% code fragment %>
```

- You can write XML equivalent of the above syntax as follows:

```
<jsp:scriptlet>
   code fragment
</jsp:scriptlet>
```

- Any text, HTML tags, or JSP elements you write must be outside the scriptlet. Following is the simple and first example for JSP:

```
<html><head><title>Hello World</title></head><body>
Hello World!<br/>
<%
out.println("Your IP address is " + request.getRemoteAddr());
%>
</body>
</html>
```


Hello World - Windows Internet Explorer
http://localhost:8080/hello.jsp
Hello World

```
Hello World!
Your IP address is 127.0.0.1
```

JSP Declarations

- A declaration declares one or more variables or methods that you can use in Java code later in the JSP file. You must declare the variable or method before you use it in the JSP file.
- Following is the syntax of JSP Declarations:

```
<%! declaration; [ declaration; ]+ ... %>
```

- You can write XML equivalent of the above syntax as follows:

```
<jsp:declaration> code fragment </jsp:declaration>
```

- Following is the simple example for JSP Declarations:

```
<%! int i = 0; %>
<%! int a, b, c; %>
<%! Circle a = new Circle(2.0); %>
```

JSP Expression

```
<html>
<head><title>A Comment Test</title></head>
<body>
<p>
  Today's date: <%= (new
  java.util.Date()).toLocaleString()%>
</p>
</body>
</html>
```

```
Today's date: 11-Sep-2010 21:24:25
```

JSP Comments

```
<html>
<head><title>A Comment
  Test</title></head>
<body>
<h2>A Test of Comments</h2>
<%-- This comment will not be visible in the
  page source --%>
</body>
</html>
```

A Test of Comments

JSP Directive

- A JSP directive affects the overall structure of the servlet class. It usually has the following form:

  <%@ directive attribute="value" %>
- There are three types of directive tag:

Directive	Description
<%@ page ... %>	Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.
<%@ include ... %>	Includes a file during the translation phase.
<%@ taglib ... %>	Declares a tag library, containing custom actions, used in the page

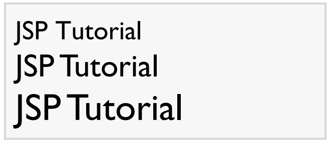
JSP switch..case

```
<%! int day = 3; %>
<html> <head><title>SWITCH...CASE Example</title></head> <body>
<%
switch(day) {
    case 0:   out.println("It\'s Sunday.");   break;
    case 1:   out.println("It\'s Monday.");   break;
    case 2:   out.println("It\'s Tuesday.");   break;
    case 3:   out.println("It\'s Wednesday.");   break;
    case 4:   out.println("It\'s Thursday.");   break;
    case 5:   out.println("It\'s Friday.");   break;
    default:   out.println("It's Saturday.");
}
%></body> </html>
```

```
It's Wednesday.
```

JSP Loop

```
<%! int fontSize; %>
<html> <head><title>FOR LOOP Example</title></head> <body>
<%for ( fontSize = 1; fontSize <= 3; fontSize++){ %>
   <font color="green" size="<%= fontSize %>">
    JSP Tutorial
   </font><br />
<%}%>
</body>
</html>
==========================================
<%! int fontSize; %>
<html> <head><title>WHILE LOOP Example</title></head> <body>
<%while ( fontSize <= 3){ %>
   <font color="green" size="<%= fontSize %>">
    JSP Tutorial
   </font><br />
<%fontSize++;%>
<%}%>
</body>
</html>
```


```
JSP Tutorial
JSP Tutorial
JSP Tutorial
```

JSP taglib directive

- The JavaServer Pages API allows you to define custom JSP tags that look like HTML or XML tags and a tag library is a set of user-defined tags that implement custom behavior.
- The taglib directive declares that your JSP page uses a set of custom tags, identifies the location of the library, and provides a means for identifying the custom tags in your JSP page.
- The taglib directive follows the following syntax:

```
<%@ taglib uri="uri" prefix="prefixOfTag" >
```

- Where the uri attribute value resolves to a location the container understands and the prefix attribute informs a container what bits of markup are custom actions.
- When you use a custom tag, it is typically of the form <prefix:tagname>.
- The prefix is the same as the prefix you specify in the taglib directive, and the tagname is the name of a tag implemented in the tag library

- For example, suppose the custlib tag library contains a tag called hello. If you wanted to use the hello tag with a prefix of mytag, your tag would be <mytag:hello> and it will be used in your JSP file as follows:

```
<%@ taglib uri="http://www.example.com/custlib" prefix="mytag" %>
<html><body>
<mytag:hello/>
</body></html>
```

JSP Form Processing-GET

- The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ? character as follows:

  http://www.test.com/hello?key1=value1&key2=value2
- The GET method is the default method to pass information from browser to web server and it produces a long string that appears in your browser's Location:box. Never use the GET method if you have password or other sensitive information to pass to the server.
- The GET method has size limitation: only 1024 characters can be in a request string.
- This information is passed using QUERY_STRING header and will be accessible through QUERY_STRING environment variable which can be handled using getQueryString() and getParameter() methods of request object.

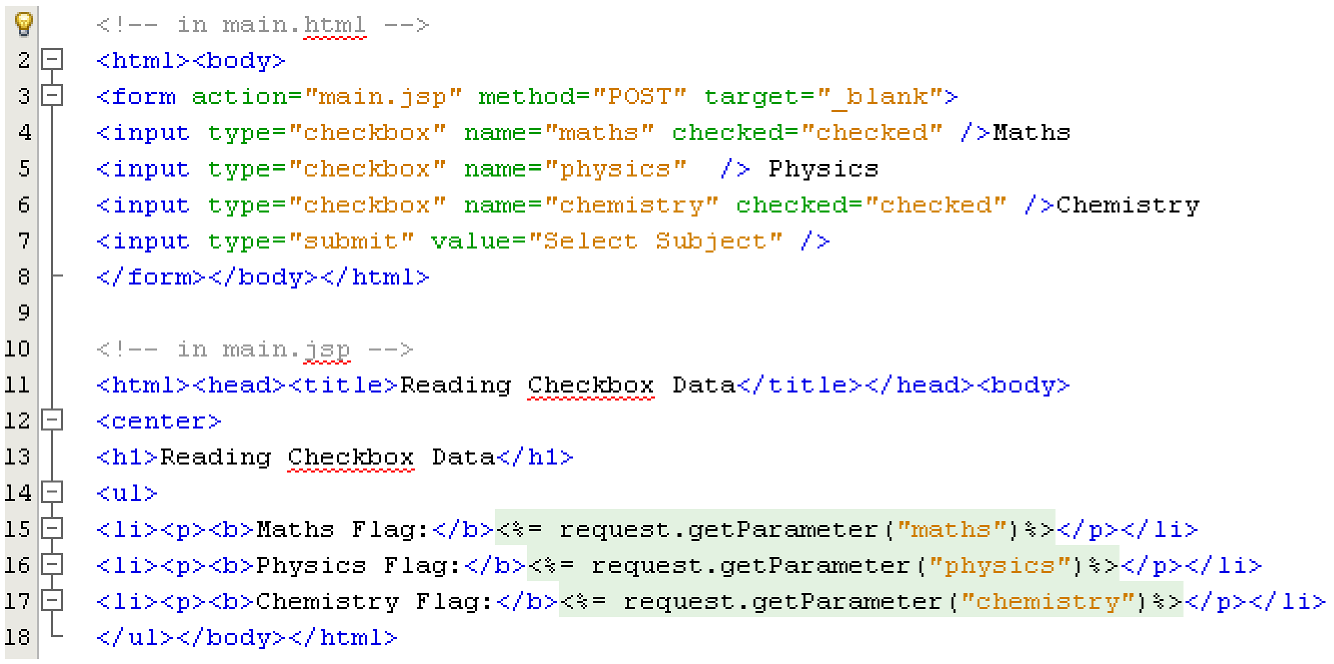
JSP Form Processing-POST

- A generally more reliable method of passing information to a backend program is the POST method.
- This method packages the information in exactly the same way as GET methods, but instead of sending it as a text string after a ? in the URL it sends it as a separate message. This message comes to the backend program in the form of the standard input which you can parse and use for your processing.
- JSP handles this type of requests using getParameter() method to read simple parameters and getInputStream() method to read binary data stream coming from the client.

Reading From Data Using JSP

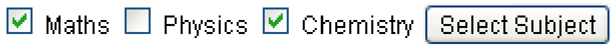
- JSP handles form data parsing automatically using the following methods depending on the situation:
- **getParameter():** You call request.getParameter() method to get the value of a form parameter.
- **getParameterValues():** Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
- **getParameterNames():** Call this method if you want a complete list of all parameters in the current request.
- **getInputStream():** Call this method to read binary data stream coming from the client.

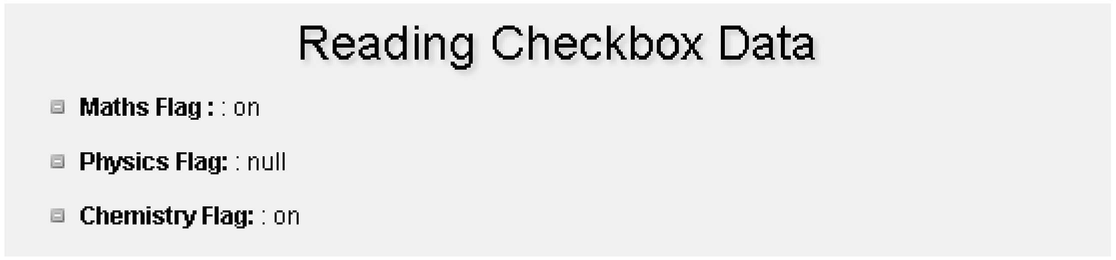
Passing Checkbox Data to JSP Program



```
<!-- in main.html -->
<html><body>
<form action="main.jsp" method="POST" target="_blank">
<input type="checkbox" name="maths" checked="checked" />Maths
<input type="checkbox" name="physics"  /> Physics
<input type="checkbox" name="chemistry" checked="checked" />Chemistry
<input type="submit" value="Select Subject" />
</form></body></html>

<!-- in main.jsp -->
<html><head><title>Reading Checkbox Data</title></head><body>
<center>
<h1>Reading Checkbox Data</h1>
<ul>
<li><p><b>Maths Flag:</b><%= request.getParameter("maths")%></p></li>
<li><p><b>Physics Flag:</b><%= request.getParameter("physics")%></p></li>
<li><p><b>Chemistry Flag:</b><%= request.getParameter("chemistry")%></p></li>
</ul></body></html>
```

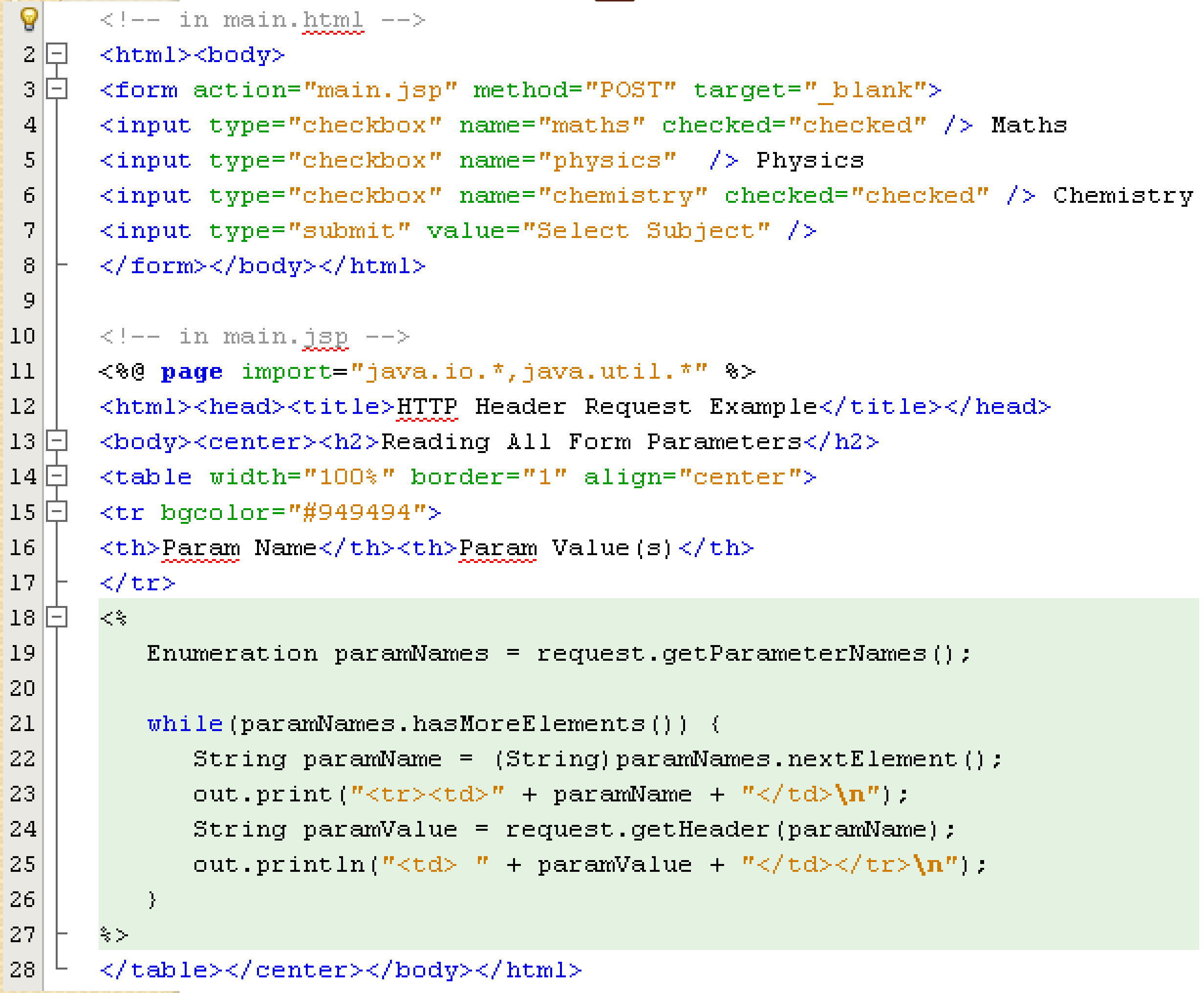




Reading Checkbox Data

- **Maths Flag :** : on
- **Physics Flag:** : null
- **Chemistry Flag:** : on

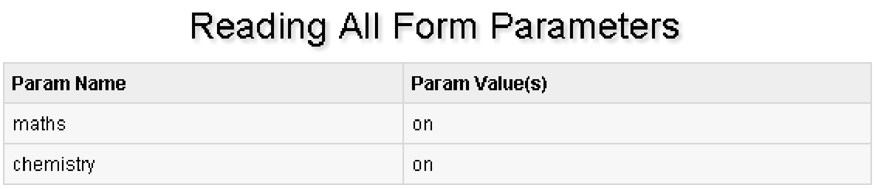
Reading All Form Parameters



```
<!-- in main.html -->
<html><body>
<form action="main.jsp" method="POST" target="_blank">
<input type="checkbox" name="maths" checked="checked" /> Maths
<input type="checkbox" name="physics"  /> Physics
<input type="checkbox" name="chemistry" checked="checked" /> Chemistry
<input type="submit" value="Select Subject" />
</form></body></html>

<!-- in main.jsp -->
<%@ page import="java.io.*,java.util.*" %>
<html><head><title>HTTP Header Request Example</title></head>
<body><center><h2>Reading All Form Parameters</h2>
<table width="100%" border="1" align="center">
<tr bgcolor="#949494">
<th>Param Name</th><th>Param Value(s)</th>
</tr>
<%
   Enumeration paramNames = request.getParameterNames();

   while(paramNames.hasMoreElements()) {
      String paramName = (String)paramNames.nextElement();
      out.print("<tr><td>" + paramName + "</td>\n");
      String paramValue = request.getHeader(paramName);
      out.println("<td> " + paramValue + "</td></tr>\n");
   }
%>
</table></center></body></html>
```



Reading All Form Parameters

Param Name	Param Value(s)
maths	on
chemistry	on

- Following is the generic example which uses **getParameterNames()** method of HttpServletRequest to read all the available form parameters. This method returns an Enumeration that contains the parameter names in an unspecified order.
- Once we have an Enumeration, we can loop down the Enumeration in the standard manner, using*hasMoreElements()* method to determine when to stop and using *nextElement()* method to get each parameter name.

JSTL

- The JavaServer Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates core functionality common to many JSP applications.
- JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags.
- The JSTL tags can be classified, according to their functions, into following JSTL tag library groups that can be used when creating a JSP page:
  - **Core Tags**
  - **Formatting tags**
  - **SQL tags**
  - **XML tags**
  - **JSTL Functions**

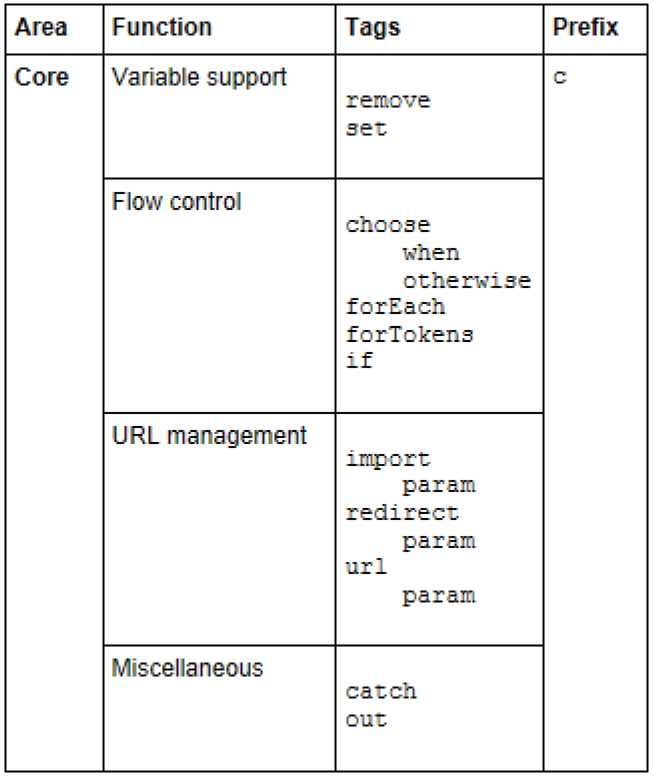
Core Tags

- The core group of tags are the most frequently used JSTL tags.
- Following is the syntax to include JSTL Core library in your JSP:

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

- There are following Core JSTL Tags:

Tag	Description
<c:out >	Like <%= ... >, but for expressions.
<c:set >	Sets the result of an expression evaluation in a 'scope'
<c:remove >	Removes a scoped variable (from a particular scope, if specified).
<c:catch>	Catches any Throwable that occurs in its body and optionally exposes it.
<c:if>	Simple conditional tag which evalutes its body if the supplied condition is true.
<c:choose>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise>
<c:when>	Subtag of <choose> that includes its body if its condition evalutes to 'true'.
<c:otherwise >	Subtag of <choose> that follows <when> tags and runs only if all of the prior conditions evaluated to 'false'.
<c:import>	Retrieves an absolute or relative URL and exposes its contents to either the page, a String in 'var', or a Reader in 'varReader'.
<c:forEach >	The basic iteration tag, accepting many different collection types and supporting subsetting and other functionality .
<c:forTokens>	Iterates over tokens, separated by the supplied delimeters.
<c:param>	Adds a parameter to a containing 'import' tag's URL.
<c:redirect >	Redirects to a new URL.
<c:url>	Creates a URL with optional query parameters



Area	Function	Tags	Prefix
Core	Variable support	remove set	c
	Flow control	choose when otherwise forEach forTokens if	
	URL management	import param redirect param url param	
	Miscellaneous	catch out	

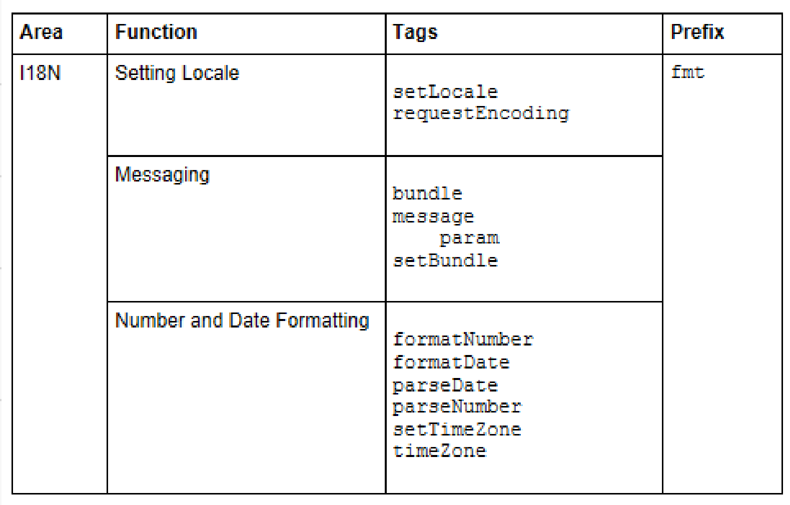
Formatting Tags

- The JSTL formatting tags are used to format and display text, the date, the time, and numbers for internationalized Web sites.
- Following is the syntax to include Formatting library in your JSP:

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
```

- Following is the list of Formatting JSTL Tags:

Tag	Description
<fmt:formatNumber>	To render numerical value with specific precision or format.
<fmt:parseNumber>	Parses the string representation of a number, currency, or percentage.
<fmt:formatDate>	Formats a date and/or time using the supplied styles and pattern
<fmt:parseDate>	Parses the string representation of a date and/or time
<fmt:bundle>	Loads a resource bundle to be used by its tag body.
<fmt:setLocale>	Stores the given locale in the locale configuration variable.
<fmt:setBundle>	Loads a resource bundle and stores it in the named scoped variable or the bundle configuration variable.
<fmt:timeZone>	Specifies the time zone for any time formatting or parsing actions nested in its body.
<fmt:setTimeZone>	Stores the given time zone in the time zone configuration variable
<fmt:message>	To display an internationalized message.
<fmt:requestEncoding>	Sets the request character encoding



Area	Function	Tags	Prefix
I18N	Setting Locale	`setLocale` `requestEncoding`	`fmt`
	Messaging	`bundle` `message` `param` `setBundle`	
	Number and Date Formatting	`formatNumber` `formatDate` `parseDate` `parseNumber` `setTimeZone` `timeZone`	

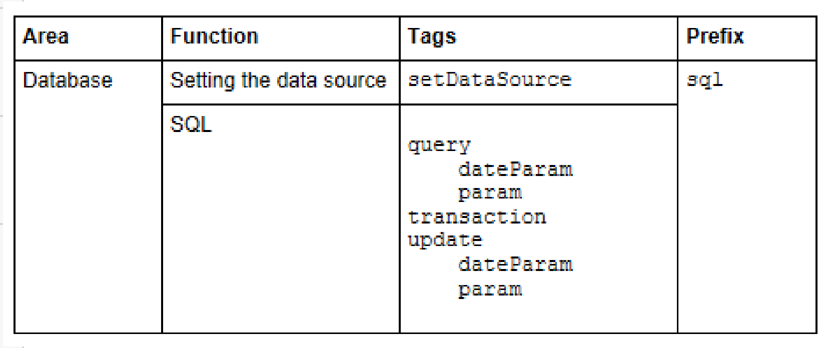
SQL Tags

- The JSTL SQL tag library provides tags for interacting with relational databases (RDBMSs) such as Oracle, mySQL, or Microsoft SQL Server.
- Following is the syntax to include JSTL SQL library in your JSP:

```
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
```

- Following is the list of SQL JSTL Tags:

Tag	Description
<sql:setDataSource>	Creates a simple DataSource suitable only for prototyping
<sql:query>	Executes the SQL query defined in its body or through the sql attribute.
<sql:update>	Executes the SQL update defined in its body or through the sql attribute.
<sql:param>	Sets a parameter in an SQL statement to the specified value.
<sql:dateParam>	Sets a parameter in an SQL statement to the specified java.util.Date value.
<sql:transaction >	Provides nested database action elements with a shared Connection, set up to execute all statements as one transaction.



Area	Function	Tags	Prefix
Database	Setting the data source	setDataSource	sql

	SQL	query
		    dateParam
		    param
		transaction
		update
		    dateParam
		    param	

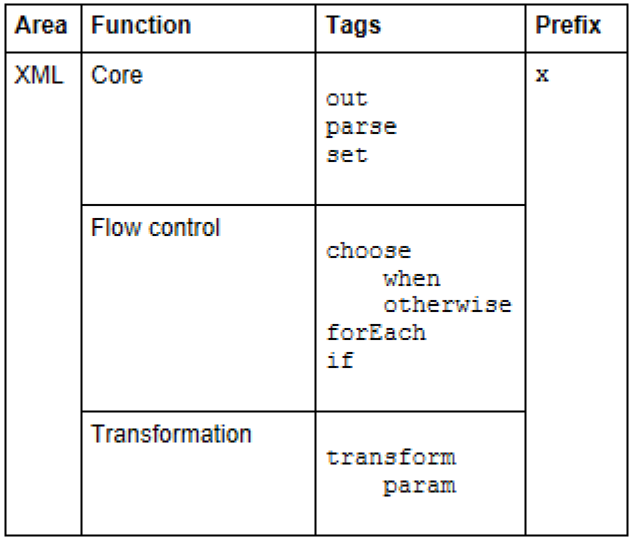
XML Tags

- The JSTL XML tags provide a JSP-centric way of creating and manipulating XML documents. Following is the syntax to include JSTL XML library in your JSP.
- The JSTL XML tag library has custom tags for interacting with XML data. This includes parsing XML, transforming XML data, and flow control based on XPath expressions.

  <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

- Before you proceed with the examples, you would need to copy following two XML and XPath related libraries into your <Tomcat Installation Directory>\lib:
  - **XercesImpl.jar:** Download it from http://www.apache.org/dist/xerces/j/
  - **xalan.jar:** Download it from http://xml.apache.org/xalan-j/index.html
- Following is the list of XML JSTL Tags:

Tag	Description
<x:out>	Like <%= ... >, but for XPath expressions.
<x:parse>	Use to parse XML data specified either via an attribute or in the tag body.
<x:set >	Sets a variable to the value of an XPath expression.
<x:if >	Evaluates a test XPath expression and if it is true, it processes its body. If the test condition is false, the body is ignored.
<x:forEach>	To loop over nodes in an XML document.
<x:choose>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise>
<x:when >	Subtag of <choose> that includes its body if its expression evalutes to 'true'
<x:otherwise >	Subtag of <choose> that follows <when> tags and runs only if all of the prior conditions evaluated to 'false'
<x:transform >	Applies an XSL transformation on a XML document
<x:param >	Use along with the transform tag to set a parameter in the XSLT stylesheet



Area	Function	Tags	Prefix
XML	Core	out parse set	x
	Flow control	choose when otherwise forEach if	
	Transformation	transform param	

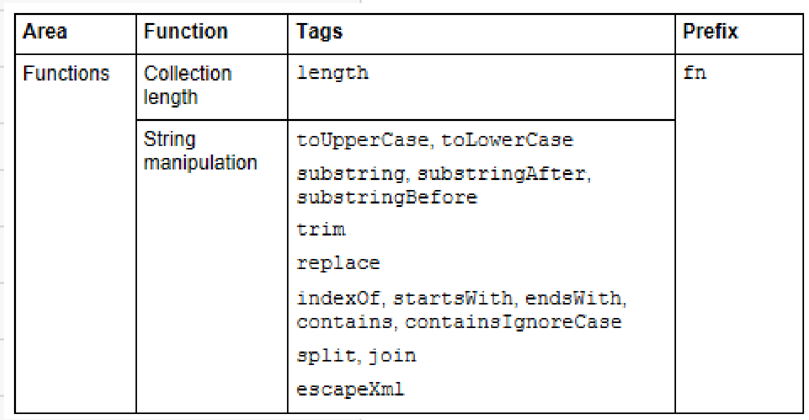
JSTL Functions

- JSTL includes a number of standard functions, most of which are common string manipulation functions.
- Following is the syntax to include JSTL Functions library in your JSP:

```
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
```

- Following is the list of JSTL Functions:

Function	Description
fn:contains()	Tests if an input string contains the specified substring.
fn:containsIgnoreCase()	Tests if an input string contains the specified substring in a case insensitive way.
fn:endsWith()	Tests if an input string ends with the specified suffix.
fn:escapeXml()	Escapes characters that could be interpreted as XML markup.
fn:indexOf()	Returns the index withing a string of the first occurrence of a specified substring.
fn:join()	Joins all elements of an array into a string.
fn:length()	Returns the number of items in a collection, or the number of characters in a string.
fn:replace()	Returns a string resulting from replacing in an input string all occurrences with a given string.
fn:split()	Splits a string into an array of substrings.
fn:startsWith()	Tests if an input string starts with the specified prefix.
fn:substring()	Returns a subset of a string.
fn:substringAfter()	Returns a subset of a string following a specific substring.
fn:substringBefore()	Returns a subset of a string before a specific substring.
fn:toLowerCase()	Converts all of the characters of a string to lower case.
fn:toUpperCase()	Converts all of the characters of a string to upper case.
fn:trim()	Removes white spaces from both ends of a string.



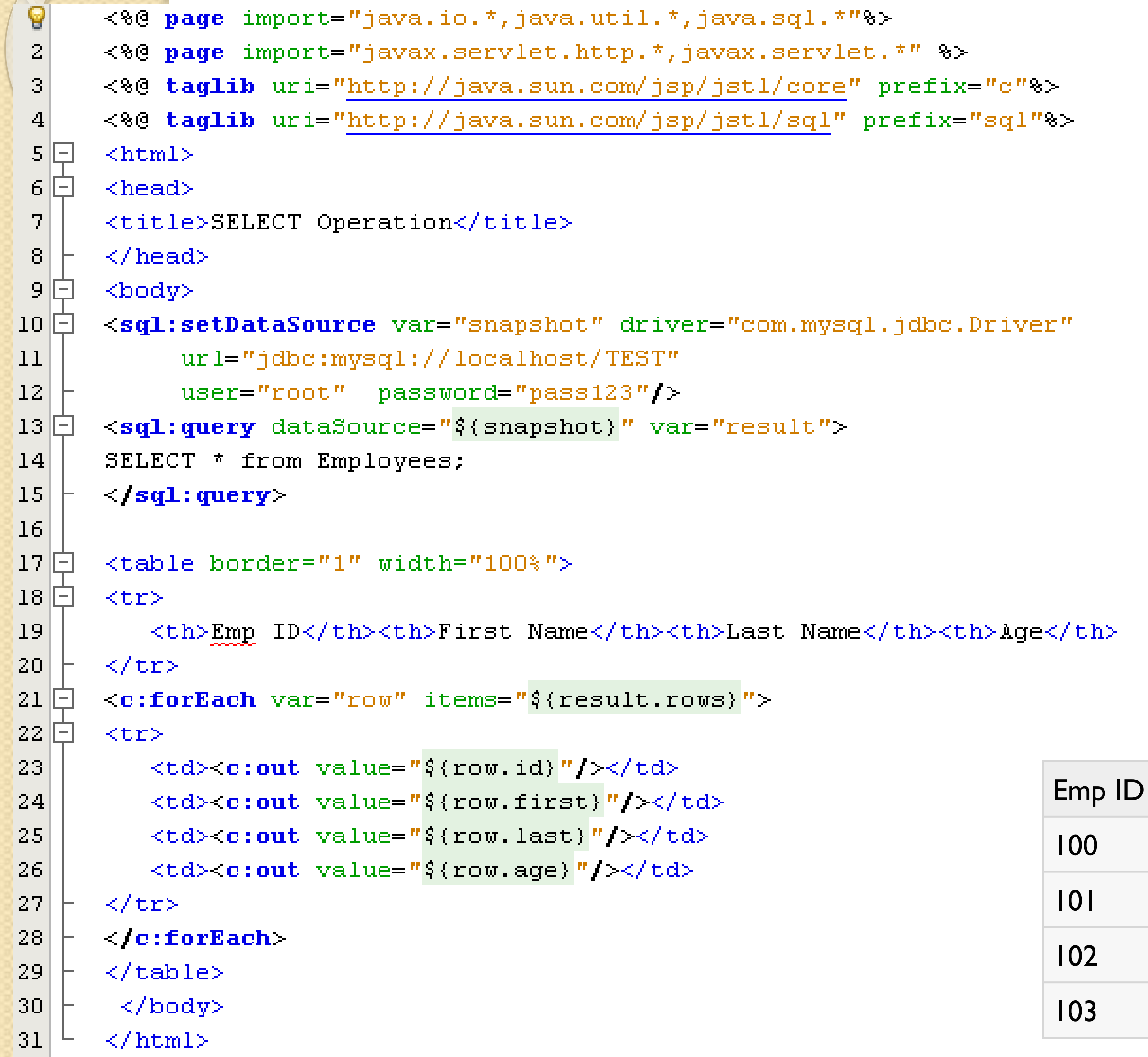
Area	Function	Tags	Prefix
Functions	Collection length	`length`	`fn`

	String manipulation	`toUpperCase`, `toLowerCase`
		`substring`, `substringAfter`, `substringBefore`
		`trim`
		`replace`
		`indexOf`, `startsWith`, `endsWith`, `contains`, `containsIgnoreCase`
		`split`, `join`
		`escapeXml`	

MySQL Environment

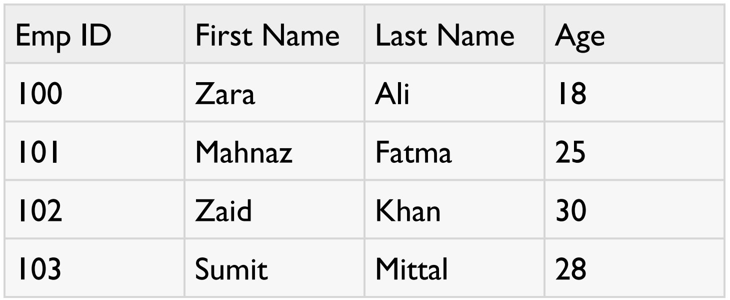
```
mysql> use TEST;
mysql> create table Employees
    (
     id int not null,
     age int not null,
     first varchar (255),
     last varchar (255)
    );
Query OK, 0 rows affected (0.08 sec)
mysql>
mysql> INSERT INTO Employees VALUES (100, 18, 'Zara', 'Ali');
Query OK, 1 row affected (0.05 sec)
 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)
 mysql>
```

SELECT Operation



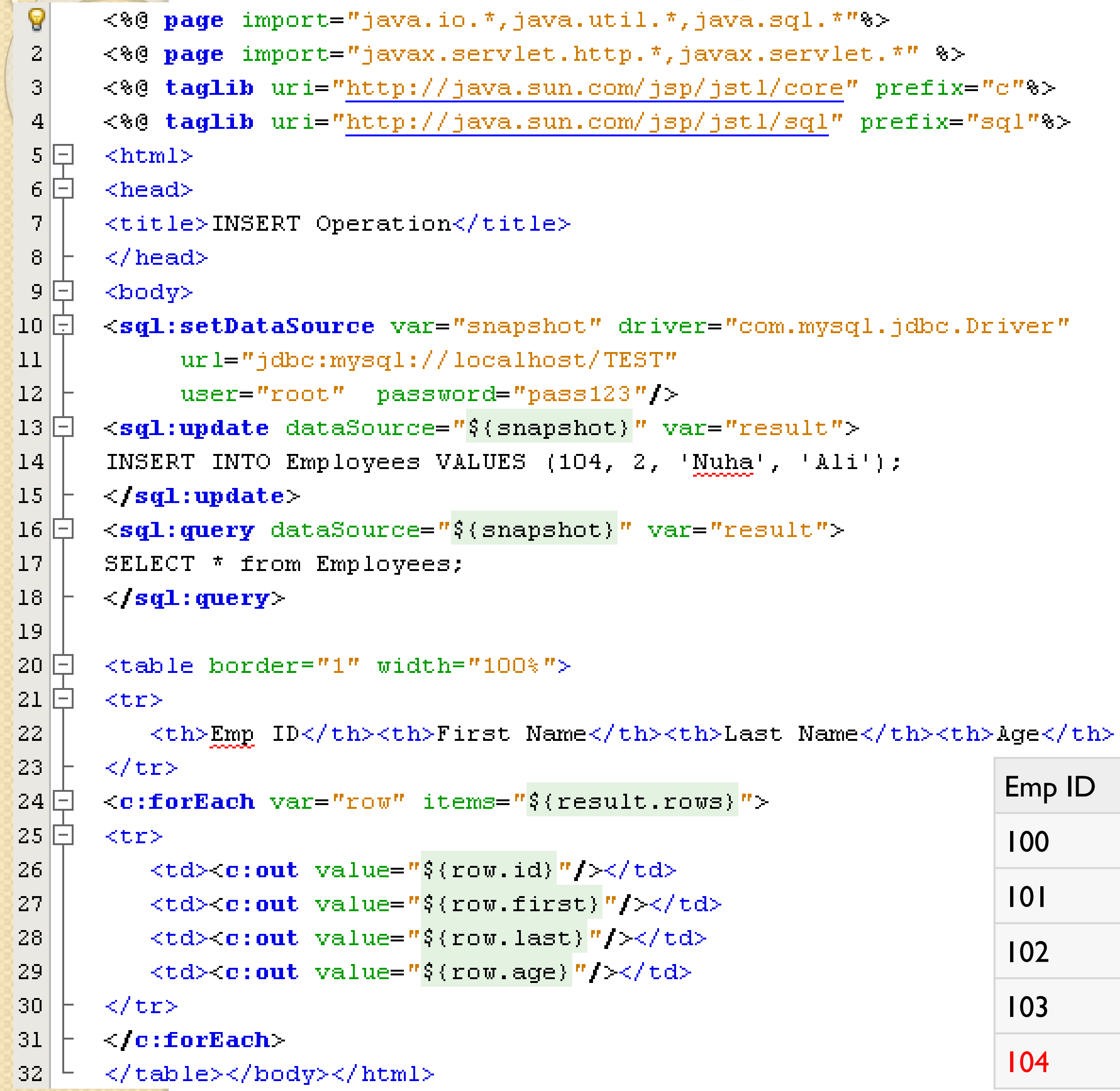
```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
<head>
<title>SELECT Operation</title>
</head>
<body>
<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
     url="jdbc:mysql://localhost/TEST"
     user="root"  password="pass123"/>
<sql:query dataSource="${snapshot}" var="result">
SELECT * from Employees;
</sql:query>

<table border="1" width="100%">
<tr>
   <th>Emp ID</th><th>First Name</th><th>Last Name</th><th>Age</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
   <td><c:out value="${row.id}"/></td>
   <td><c:out value="${row.first}"/></td>
   <td><c:out value="${row.last}"/></td>
   <td><c:out value="${row.age}"/></td>
</tr>
</c:forEach>
</table>
 </body>
</html>
```



Emp ID	First Name	Last Name	Age
100	Zara	Ali	18
101	Mahnaz	Fatma	25
102	Zaid	Khan	30
103	Sumit	Mittal	28

INSERT Operation

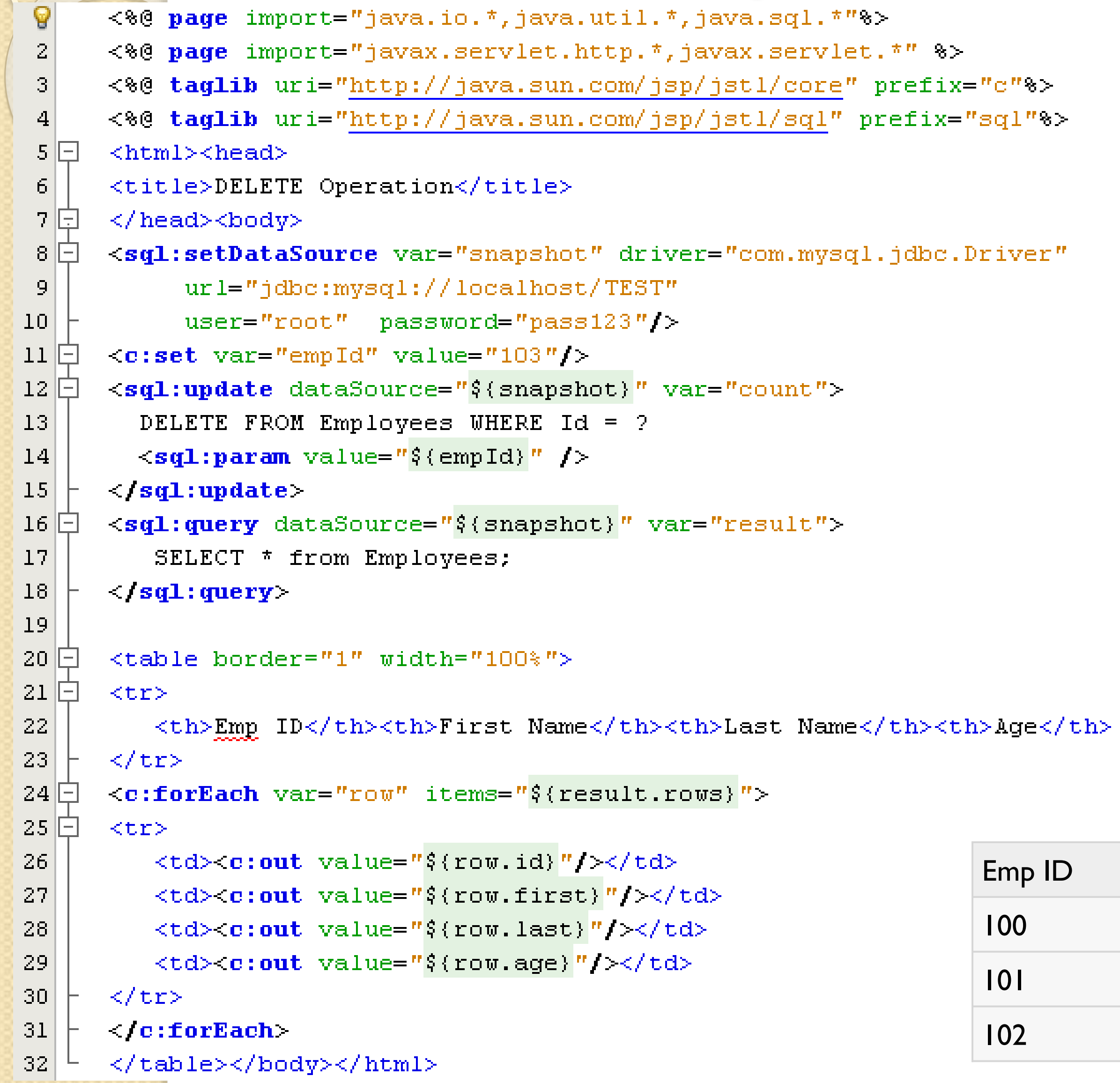


```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
<head>
<title>INSERT Operation</title>
</head>
<body>
<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
     url="jdbc:mysql://localhost/TEST"
     user="root"  password="pass123"/>
<sql:update dataSource="${snapshot}" var="result">
INSERT INTO Employees VALUES (104, 2, 'Nuha', 'Ali');
</sql:update>
<sql:query dataSource="${snapshot}" var="result">
SELECT * from Employees;
</sql:query>

<table border="1" width="100%">
<tr>
   <th>Emp ID</th><th>First Name</th><th>Last Name</th><th>Age</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
   <td><c:out value="${row.id}"/></td>
   <td><c:out value="${row.first}"/></td>
   <td><c:out value="${row.last}"/></td>
   <td><c:out value="${row.age}"/></td>
</tr>
</c:forEach>
</table></body></html>
```

Emp ID	First Name	Last Name	Age
100	Zara	Ali	18
101	Mahnaz	Fatma	25
102	Zaid	Khan	30
103	Sumit	Mittal	28
104	Nuha	Ali	2

DELETE Operation

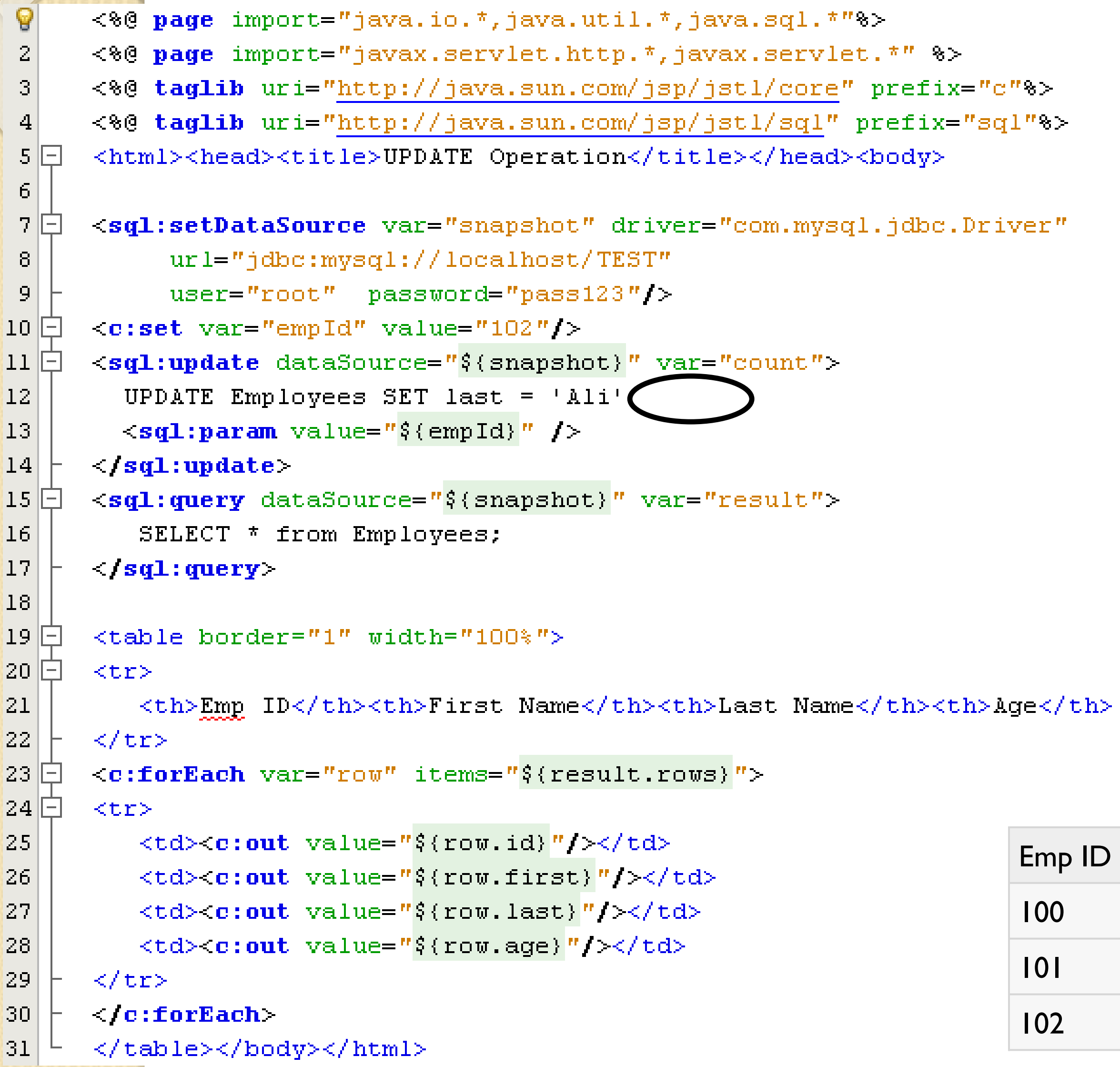


```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html><head>
<title>DELETE Operation</title>
</head><body>
<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
     url="jdbc:mysql://localhost/TEST"
     user="root"  password="pass123"/>
<c:set var="empId" value="103"/>
<sql:update dataSource="${snapshot}" var="count">
  DELETE FROM Employees WHERE Id = ?
  <sql:param value="${empId}" />
</sql:update>
<sql:query dataSource="${snapshot}" var="result">
   SELECT * from Employees;
</sql:query>

<table border="1" width="100%">
<tr>
   <th>Emp ID</th><th>First Name</th><th>Last Name</th><th>Age</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
   <td><c:out value="${row.id}"/></td>
   <td><c:out value="${row.first}"/></td>
   <td><c:out value="${row.last}"/></td>
   <td><c:out value="${row.age}"/></td>
</tr>
</c:forEach>
</table></body></html>
```

Emp ID	First Name	Last Name	Age
100	Zara	Ali	18
101	Mahnaz	Fatma	25
102	Zaid	Khan	30

UPDATE Operation

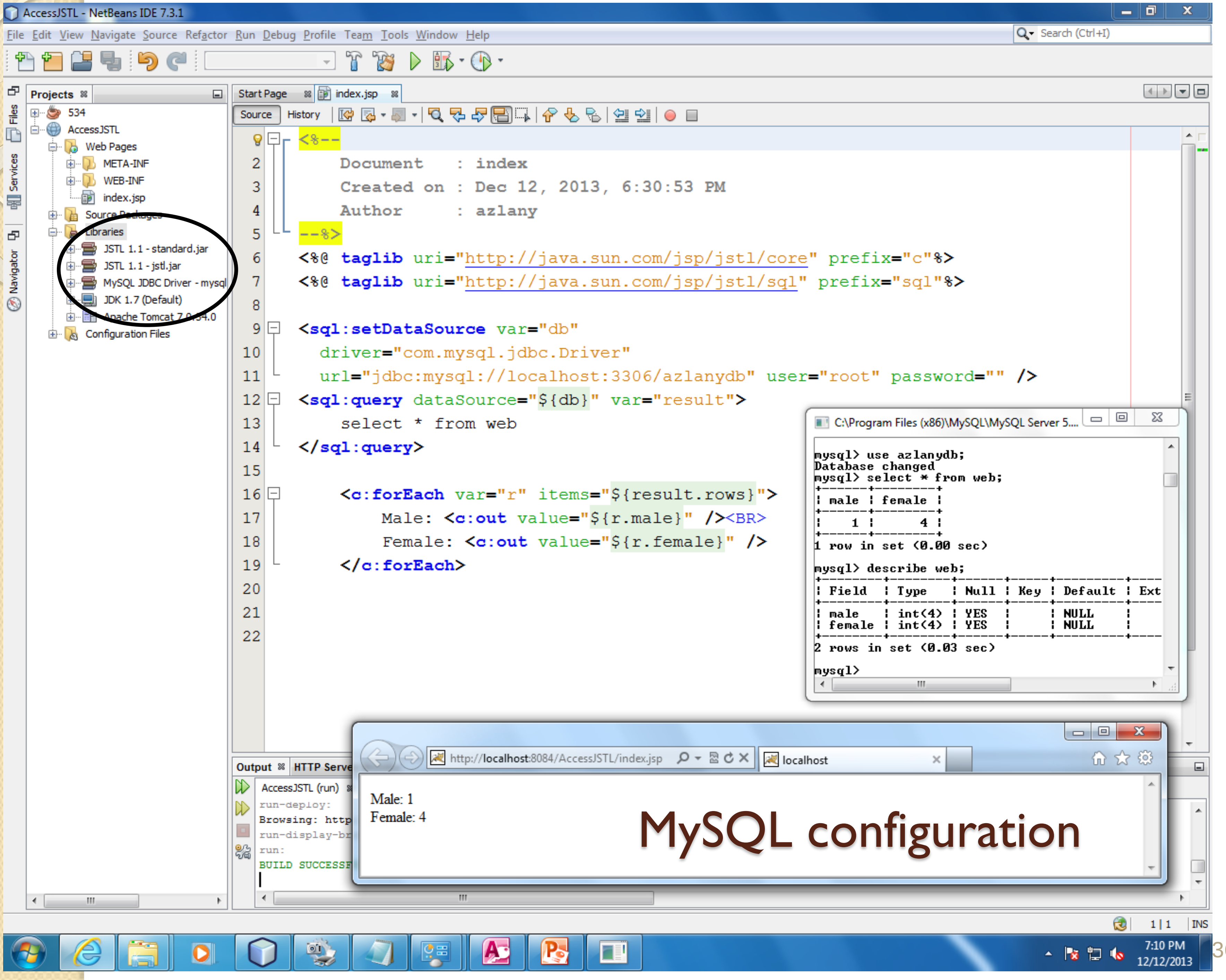


```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html><head><title>UPDATE Operation</title></head><body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
     url="jdbc:mysql://localhost/TEST"
     user="root"  password="pass123"/>
<c:set var="empId" value="102"/>
<sql:update dataSource="${snapshot}" var="count">
  UPDATE Employees SET last = 'Ali'
  <sql:param value="${empId}" />
</sql:update>
<sql:query dataSource="${snapshot}" var="result">
   SELECT * from Employees;
</sql:query>

<table border="1" width="100%">
<tr>
   <th>Emp ID</th><th>First Name</th><th>Last Name</th><th>Age</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
   <td><c:out value="${row.id}"/></td>
   <td><c:out value="${row.first}"/></td>
   <td><c:out value="${row.last}"/></td>
   <td><c:out value="${row.age}"/></td>
</tr>
</c:forEach>
</table></body></html>
```

Emp ID	First Name	Last Name	Age
100	Zara	Ali	18
101	Mahnaz	Fatma	25
102	Zaid	Ali	30



```
<%--
    Document   : index
    Created on : Dec 12, 2013, 6:30:53 PM
    Author     : azlany
--%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<sql:setDataSource var="db"
  driver="com.mysql.jdbc.Driver"
  url="jdbc:mysql://localhost:3306/azlanydb" user="root" password="" />
<sql:query dataSource="${db}" var="result">
    select * from web
</sql:query>

    <c:forEach var="r" items="${result.rows}">
        Male: <c:out value="${r.male}" /><BR>
        Female: <c:out value="${r.female}" />
    </c:forEach>
```

```
run-deploy:
Browsing: http
run-display-br
run:
BUILD SUCCESSF
```

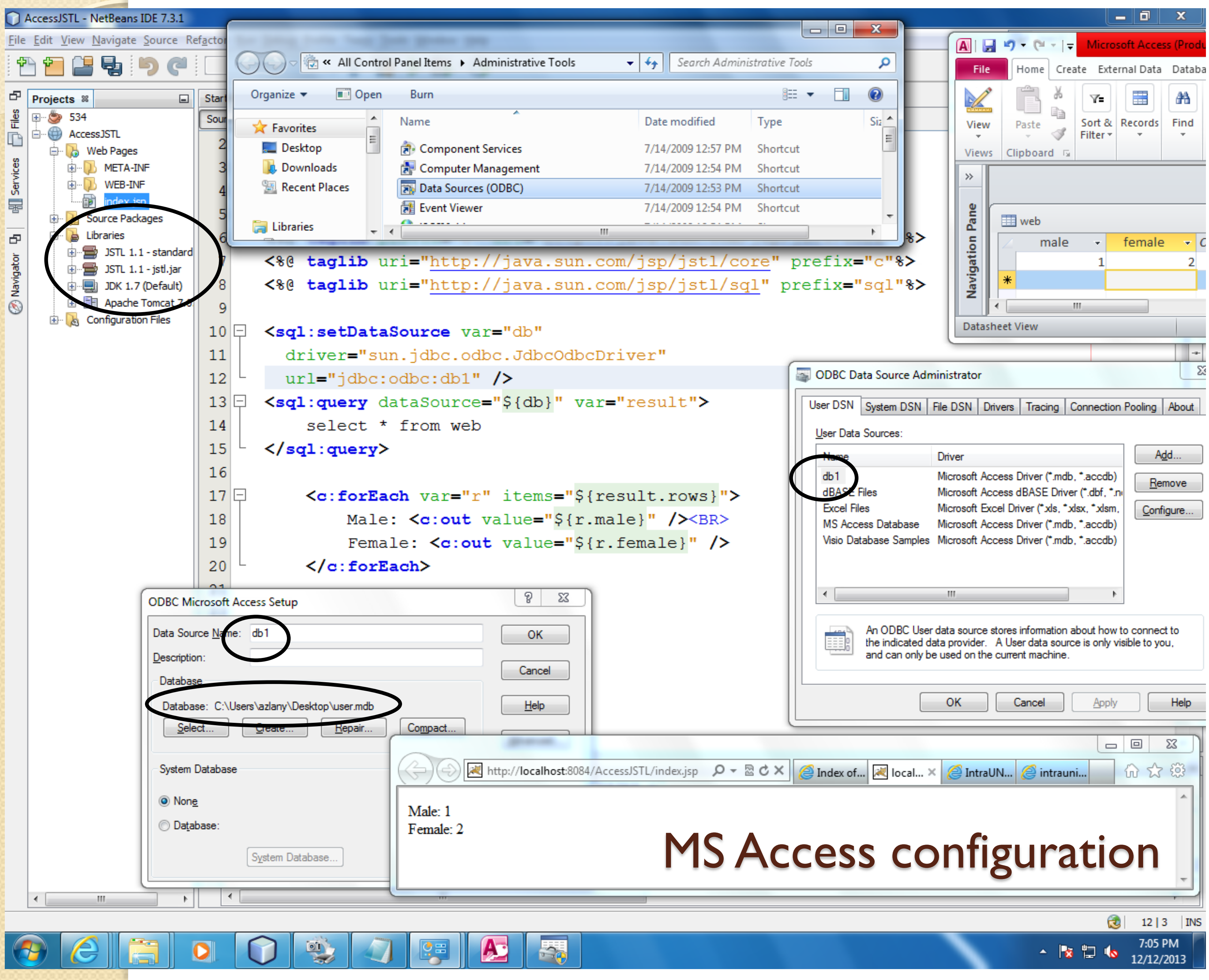
```
mysql> use azlanydb;
Database changed
mysql> select * from web;
+------+--------+
| male | female |
+------+--------+
|    1 |      4 |
+------+--------+
1 row in set (0.00 sec)

mysql> describe web;
+--------+--------+------+-----+---------+----
| Field  | Type   | Null | Key | Default | Ext
+--------+--------+------+-----+---------+----
| male   | int(4) | YES  |     | NULL    |
| female | int(4) | YES  |     | NULL    |
+--------+--------+------+-----+---------+----
2 rows in set (0.03 sec)

mysql>
```

Male: 1
Female: 4

MySQL configuration



MS Access configuration

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

<sql:setDataSource var="db"
  driver="sun.jdbc.odbc.JdbcOdbcDriver"
  url="jdbc:odbc:db1" />
<sql:query dataSource="${db}" var="result">
    select * from web
</sql:query>

    <c:forEach var="r" items="${result.rows}">
        Male: <c:out value="${r.male}" /><BR>
        Female: <c:out value="${r.female}" />
    </c:forEach>
```

male	female
1	2

ODBC Microsoft Access Setup

Data Source Name: db1

Description:

Database: C:\Users\azlany\Desktop\user.mdb

System Database: [✓ None] Database:

ODBC Data Source Administrator

User Data Sources:

Name	Driver
db1	Microsoft Access Driver (*.mdb, *.accdb)
dBASE Files	Microsoft Access dBASE Driver (*.dbf, *.n
Excel Files	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm,
MS Access Database	Microsoft Access Driver (*.mdb, *.accdb)
Visio Database Samples	Microsoft Access Driver (*.mdb, *.accdb)

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

http://localhost:8084/AccessJSTL/index.jsp

```
Male: 1
Female: 2
```

Code sample

```
<c:if test="${not empty param.success}"> //do
   something </c:if>

<c:set var = "TR1" value="TR1"/>
<c:if test = "${class eq TR1}“>//process</c:if>

<option value="<c:out value='AAA'/>" >
<c:if test="${transaction eq 'AAA'}"> selected='selected
   '</c:if>
<c:out value='AAA'/></option>

<c:if test="${str == 'hello'}“>Both string are
   equal;</c:if>
```

Code sample

```
<select>
<c:forEach var="name" items="${names}">
<option value="${name}">${name}</option>
</c:forEach>
</select>

<input type=radio name=gender
  value=male>Male
<c:out value="${param.gender}" />
<c:out value="${gender}" />
```

Code sample

```
<sql:update dataSource="${snapshot}">
  DELETE from employess WHERE Id = '?'
   AND age = '?'
  <sql:param value="${empid}">
  <sql:param value="${age}">
</sql:update>
```