|  |  |
| --- | --- |
| **Contents** | |
| **Slno** | **Topic** |
| **JDBC(Java Database Connectivity)** | |
| 1 | JDBC Inroduction |
| 2 | Creating JDBC Applications |
| 3 | Statement,PreparedStatement,CallableStatement |
| 4 | Resultset types,BatchUpdates |
| 5 | Transaction Management |
| 6 | MetaData,Types of Jdbc drivers |
| **Servlets** | |
| 1 | Introduction to Webaplications |
| 2 | Http protocol |
| 3 | Creating Webbased Applications |
| 4 | Servlet API |
| 5 | Developing formbased applications |
| 6 | CGI ,Advantages of Servlets over CGI |
| 7 | Listeners |
| 8 | Filters |
| 9 | Servlet lifecycle |
| **JSP(Java Server Pages)** | |
| 1 | Introduction to Jsp |
| 2 | Jsp Architecture and lifecycle methods |
| 3 | Jsp Elements |
| 4 | Jsp implicit objects |
| 5 | Session tracking techniques |
| 6 | RequestDispatchers |
| 7 | Scopped variables |
| 8 | Jsp actiontags |
| 9 | Java beans |
| 10 | Design patterns(MVC1,MVC2) |
| 11 | Jsp custom tags,JSTL |
| 12 | ELExpressions |

**JDBC**

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The main purpose of databases is to store the data perminently.database server provides more security to huge amount data available in database.there are so many database servers are available in the market .some of them are Oracle,MySql,Sybase,IBM DB2,MsSql Server,Teradat,point database,ingrus etc…

**Beneifits of JDBC**

The JDBC API provides a set of implementation-independent generic database access methods for the above mentioned SQL-compliant databases. JDBC abstracts much of the vendor-specific details and generalizes the most common database access functions. Thus resulted a set of classes and interfaces of the java.sql package that can be used with any database providing JDBC connectivity through a vendor-specific JDBC driver in a consistent way. Thus if our application conforms to the most commonly available database features, we should be able to reuse an application with another database simply by switching to a new JDBC driver. In other words, JDBC enables us to write applications that access relational databases without any thought as to which particular database we are using.

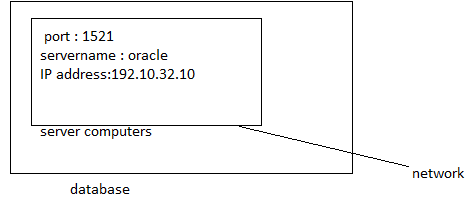
Also database connectivity is not just connecting to databases and executing statements. In an enterprise-level application environment, there are some important requirements to be met, such as optimizing network resources by employing connection pooling, and implementing distributed transactions. JDBC has all these features in accomplishing advanced database programming.

Every database server vendor provide two types of software they are

1 .server software

2 client software

The database server software will be installed on server computer.And database client software is installed on client computer. In the industry the database server is installed on server computer.network administrators connect database in a network.the database administrators are responsible to install and maintain database server in network.



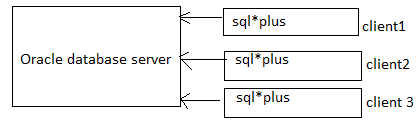
If any program wants to communicate with the database server we required 3 informations

1 IP address of database server

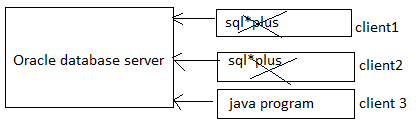
2 service name of database server

3 port no of database server.

In all the developer computers we install database client software. For example from your computer ,if you would like to access the oracle database server,in your computer you have to install database client software.



We can develop the java programs which interacts with the database servers.java programs are the alternative to the client program given by the database vendors.here java program are responsible to get the data entered by the users and send the data to database servers.



Every database sever vendor releases two different softwares.they are

1. Enterprise edition
2. Express edition

If we want to use enterprise edition, we have to buy the softwareby using enterprise edition we can configure our own service name,our own port number etc..

Express edition is free of cost .in this version we cannot change servicename,port number.in express edition the service name is xe and port number is 1521 for oracle.

Every database server uses files to store the data.these files are called s database files.the database server is responsible to store the data into database files,database server is responsible to retrieve the data from database files.every database server uses its own proprietery algorithms to store the data into database files and as well as for retrieve.



The JDBC library includes APIs for each of the tasks commonly associated with database usage:

* Making a connection to a database
* Creating SQL or MySQL statements
* Executing that SQL or MySQL queries in the database
* Viewing & Modifying the resulting records

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as:

* Java Applications
* Java Applets
* Java Servlets
* Java ServerPages (JSPs)
* Enterprise JavaBeans (EJBs)

All of these different executables are able to use a JDBC driver to access a database and take advantage of the stored data.

JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

**Creating JDBC Application.**

There are six steps involved in building a JDBC application.they are

1. **Register the driver/load the vendor specific driver :**

Sunmicrosystems has released JDBC API.so many companies has provided the implementation to JDBC API.(i,e they have released jdbc drivers).to interact with the database we require drivers

Asking the java program to use a specific driver is called as registering the jdbc driver.

Whenever we register the jdbc driver ,java program loads all the required classes into Jvm’m memory to interact with database server.

According to the documentation every driverclass should be able to create the object on its own and register the driver.

|  |
| --- |
| package oracle.jdbc.driver;  public class OracleDriver implements Driver{  static {  OracleDriver o=new OracleDriver();  DriverManager.registerDriver(o);  }  } |

OracleDriver.java

Similar to above code in all the drivers classes static blocks are provided.

Eg: DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver ());

The above statement will register the driver which we are passing.

1. **Getting the Connection** :

Establishing the physical connection between database server and java application is called as getting Connection.

Connection con=DriverManager.getConnection (url, username, password);

Where url is the connection string given by the database server vendor and username and password are the credentials of the database user.

Eg:

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

1. **Create the jdbc Statement object :**

A JDBC Statement object is used to send your SQL statements to the DBMS, and should not to be confused with an SQL statement. A JDBC Statement object is associated with an open connection, and not any single SQL Statement. You can think of a JDBC Statement object as a channel sitting on a connection, and passing one or more of your SQL statements (which you ask it to execute) to the DBMS.

Statement stmt=con.createStatement ();

1. **Executing queries /sql statements :**

Executing SQL statements in JDBC varies depending on the intention of the SQL statement. DDL (data definition language) statements such as table creation and table alteration statements, as well as statements to update the table contents, are all executed using the method executeUpdate. Notice that these commands change the state of the database, hence the name of the method contains Update.if we want to execute select queries like queries which return some data,we use execteQuery().

stmt.executeUpdate(String query) //for nonselect queries

stmt.executeQuery(String query) // for select queries.

eg :

stmt.executeUpdate(“create table student(sid number(3),name varchar2(30),address varchar2(30))”);

1. **Closing the connection.**

Here we release the resources from database interaction by closing the connections and statements etc…

con.close ();

or stmt.close()

* Program to create table in oracle database.

|  |
| --- |
| import java.sql.\*;  class CreateTable{  public static void main(String ar[]) throws SQLException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  Statement stmt=con.createStatement();  int no=stmt.executeUpdate("create table Employee(eid number(3),name varchar2(25),address varchar2(20))");  if(no!=0){  System.out.println(“table created sucessfully”);  }  }  } |

CreateTable.java

o/p:

table created successfully…

* Program to insert data into table Employee

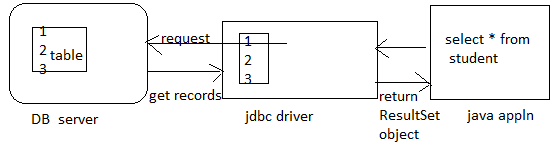
|  |
| --- |
| import java.sql.\*;  class InsertData{  public static void main(String ar[]) throws SQLException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  Statement stmt=con.createStatement();  int no=stmt.executeUpdate("insert into Employee values(101,'gopal','Hyderabad')");  if(no>0)  {  System.out.println("data inserted successfully");  }  }  } |

InsertData.java

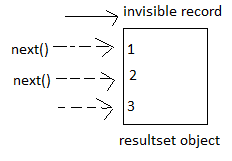
o/p:

data inserted successfully.

We can pass select and nonselect queries to the database server.nonselect queries return integer number which represents no of records affected with our query,where as select queries return ResultSet object which contains data given by the jdbc driver.



For select queries to get the data available in the ResultSet. we need to iterate the ResultSet object. ResultSet object has one pointercalled resultset pointer ,which points to a null record initially..to access the records in the resultset object , we sholud locate the resultset pointer to firstrecord by calling next().



Whenever we got the resultset object,every resultset object is associated with a resultset pointer.initially resultsetpointer pointing to invisible record before the firstrow as shown in above diag. to move the resultset pointer we use the method next() on ResultSet object.

boolean next();

here next() returns true if the next record is available else return false.

* To read the value of any column,first we need to find the datatype of the column in the table.now we need to identify the corresponding java datatype .now we need to call appropriate getter method.

|  |  |
| --- | --- |
| **Sql datatype** | **java datatype** |
| VARCHAR | java.lang.String |
| CHAR | java.lang.String |
| LONGVARCHAR | java.lang.String |
| BIT | boolean |
| NUMERIC | java.math.BigDecimal |
| TINYINT | byte |
| SMALLINT | short |
| INTEGER | int |
| BIGINT | long |
| REAL | float |
| FLOAT | float |
| DOUBLE | double |
| DATE | java.sql.Date |
| TIME | java.sql.Time |
| TIMESTAMP | java.sql.TimeStamp |
| CLOB | java.sql.Clob |
| BLOB | java.sql.Blob |

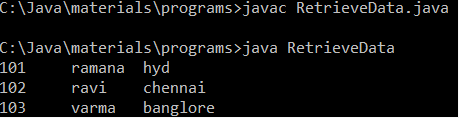
The following program retrieve the data from table Employee in oracle database

* Program to retrieve data from Employee table and display to the user.

|  |
| --- |
| import java.sql.\*;  class RetrieveData{  public static void main(String ar[]) throws SQLException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  Statement stmt=con.createStatement();  ResultSet rs=stmt.executeQuery("select \* from Employee");  while(rs.next()){  System.out.print(rs.getInt(“eno”)+"\t");  System.out.print(rs.getString(“name”)+"\t");  System.out.print(rs.getString(“address”)+"\n");  }  }  } |

RetrieveData.java

o/p:



Till now we have used getXXX(String columnname ) to retrieve specific cloumn values from table.we can retrieve the records based on the index of the columns also.

Eg:

getXXX(int columnindex);

observe below program in which,we are getting the records base on column index .

we can retrieve the records from resultset in two ways.

* 1. getXXX(String columnname)
  2. getXXX(int column index)

it is always recomonded to use getXXX(String columnname).

* Program to retrieve specific records from the database tables

|  |
| --- |
| import java.sql.\*;  public class RetrieveRecords  {  public static void main(String ar[]) throws SQLException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  Statement stmt=con.createStatement();  ResultSet rs=stmt.executeQuery("select \* from employee where eno=101");  while(rs.next()){  System.out.print(rs.getString(1)+"\t");  System.out.print(rs.getString(2)+"\t");  System.out.print(rs.getString(3)+"\n");  }  }  } |

RetrieveRecords.java

o/p: 

It is not recomonded to use getString() method for all the datatypes.it’s recomonded to use appropriate getXXX() to retrieve the data.

**PreparedStatement**

PreparedStatement is an object which is used to send the queries to the database server.

PreparedStatement improves the performance of java application when compared with the Statement

Procedure to work with PreparedStatement :

1. Register the jdbc driver
2. Get the connection from the database server
3. Create the PreparedStatement object.(supply the query with placeholders).
4. Supply the values to place holders
5. Call the appropriate method based on query(select/nonselect).
6. Close connection

* Java program to insert records into employee table using PreparedStatement.

|  |
| --- |
| import java.sql.\*;  public class RetrieveRecords  {  public static void main(String ar[]) throws SQLException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  PreparedStatement pstmt=con.prepareStatement("insert into employee values(?,?,?)");  pstmt.setInt(1,101);  pstmt.setString(2,"gopi");  pstmt.setString(3,"hyd");  int no=pstmt.executeUpdate();  if(no!=0){  System.out.println("data stored successfully");  }  pstmt.close();  con.close();  }  }  } |

RetrieveRecords.java

o/p:

data inserted successfully

* Java program which uses the PreparedStatement to retrieve the records from the database server

|  |
| --- |
| import java.sql.\*;  public class RetrieveRecords  {  public static void main(String ar[]) throws SQLException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  PreparedStatement pstmt=con.prepareStatement("select \* from employee where eno=?");  pstmt.setInt(1,101);  ResultSet rs=pstmt.executeQuery();  while(rs.next()){  System.out.print(rs.getString(1)+"\t");  System.out.print(rs.getString(2)+"\t");  System.out.print(rs.getString(3)+"\n");  }  }  } |

RetrieveRecords.java

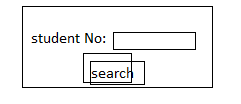
Output:



In the Preparedstatement if we have 3 place holders,we have to supply the values to all the 3 placeholders.if we didn’t supply the values to all the place holders ,the appliction fails.in the preparedtstatement,there is no rule saying compulsory the query must conatin placeholders.we can use the query without placeholders(?) also.  
the preparedstatement will improve the performance of application,if the query contains placeholders (?)’s.

**PreparedStatement improves performance of Java application.**

While we developing the projects,some times we need to send queries with different values.for eg: we developed the following screen.



If the use enter the stuent no value as 1 and click on search,our application ‘ll send query as select \* from student where sno=1.

If the user enters studentno value as 2 the query will be

Select \* from student where sno=2.

Database server treats these two queries as two different individual queries and execute them .

To resolve the problem of sending the same queries with different values in the database people are using bind variables.

To use bind variables instead of giving data directly in the query,they using like

select \* from student where sno=:no

if we use PeparedStatement in java,jdbc driver then converts the query to bind variables query and send it to server.PreparedStatement improves the performance of the application if query is like above.(i.e where sending same query with multiple values).

**CallableStatement**

CallableStatement is used to execute the procedure of database from java application.

A CallableStatement object provide away to call the storedprocedures in a standard way for all RDBMSs. A storedprocedure is stored in a database. the call to the storedprocedure is what a CallableStatement object means.

**Stored Procedures**

**Procedures without parameters:**

A simple stored procedure in oracle database :

|  |
| --- |
| create or replace procedure myproc  as  begin  insert into student values(10,’varma’,’hyderabad’);  end myproc;  / |

To execute the above procedure

SQL> exec myproc;

**Procedures which takes in parameters:**

|  |
| --- |
| create or replace procedure myproc(vsno in number,vsname in varchar,vaddress in varchar)  as  begin  insert into student values(vsno,vsname,vaddress);  end myproc  / |

To execute the above procedure

SQL> exec myproc(4,’gopi’,’hyderabad’);

**Procedures which takes in and out parameters**

|  |
| --- |
| create or replace procedure addition(vno1 in number,vno2 in number,result out number)  as  begin  result :=vno1+vno2;  end addition;  / |

To execute the above procedure we have to perform following steps:

1. Create a varibale in the database server.

SQL>variable result number;

this variable is used to store the output of the out variable of the procedure.

1. To call the procedure use the following command

SQL>exec addition(10,20, :result);

After the procedure is executed successfully the output is stored in variable result

1. To see the value avilable in variable result use command print

SQL>print result;

result

30

**Procedure to call the database procedures from java application**.

Step1 : register the driver

Step2 : get the connection

Step3 : create the callable statement

Step4: call the procedure

Step 5 : close the connection

* Java program to call a procedure with out any parameters

|  |
| --- |
| import java.sql.\*;  public class RetrieveRecords  {  public static void main(String ar[]) throws SQLException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  CallableStatement cstmt=con.prepareCall("{call myproc}");  boolean status=cstmt.execute();  if(status==t  }  } |

RetrieveRecords.java

* Java program to call the procedures with IN parameters

|  |
| --- |
| import java.sql.\*;  public class RetrieveRecords  {  public static void main(String ar[]) throws SQLException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  CallableStatement cstmt=con.prepareCall("{call myproc(?,?,?)}");  cstmt.setInt(1,5);  cstmt.setString(2,"ravi")  cstmt.setString(3,"hyd");  cstmt.execute();  }  } |

RetrieveRecords.java

* Java program to call the procedures with IN and OUT parameters

|  |
| --- |
| import java.sql.\*;  public class RetrieveRecords  {  public static void main(String ar[]) throws SQLException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","prasad");  CallableStatement cstmt=con.prepareCall("{call myproc(?,?,?)}");  cstmt.setInt(1,10);  cstmt.setInt(2,20);  cstmt.registerOutParameter(3,Types.INTEGER); //registering out variable using Types class  cstmt.execute();  int result=cstmt.getInt(3);  System.out.println(result);  }  } |

RetrieveRecords.java

o/p: 30

**ResultSet Types**

in jdbc we have two types of ResultSet. they are

1. ForwardOnly ResultSet
2. Bi-directional ResultSet

**ForwardOnly ResultSet**

The ResultSet which moves in forwardonly direction is called as ForwardOnly ResultSet.

**Bi-Directional ResultSet**

A ResultSet which can move forward as well as backward is called as bidirectional Resultset **.**by using forward only resultset we can use the methods like next(),getRow(),but we can not use methods like previous(),absolute(),isFirst(),isLast() and etc…

By default we get only the forwardonly resultset.if you want to create the bidirectional resultset ,we have to supply arguments to the methods prepareStatement() or createStatement().

Eg:

createStatement(resultset type,resultset concurrency)

prepareStatement(query,resultset type,resultset concurrency).

**ResultSet Type**

For the argument resultset type,we have to supply

ResultSet.TYPE\_FORWARD\_ONLY

ResultSet.TYPE\_SCROLL\_INSENSITIVE

ResultSet.TYPE\_SCROLL\_SENSITIVE

The ResultSet which we are getting by default is Forward Only ResultSet ,internally it uses

ResultSet.TYPE\_FORWARD\_ONLY.

**ResultSet Concurrency:**

ResultSet.CONCUR\_READ\_ONLY

ResultSet.CONCUR\_UPDATABLE

Here by default we have ReadOnly concurrency..

**Sensitive ResultSet:**

In java application after getting the ResultSet, if some one modifies the data in the database server,if it got reflected in the ResultSet ,then we call that ResultSet as Sensitive ResultSet.

**Insensitive ResultSet:**

In java application after getting the ResultSet, if some one modifies the data in the database server,if it not got reflected in the ResultSet ,then we call that ResultSet as Inensitive ResultSet.

**CONCURRENT\_UPDATABLE:**

Whenever we perform some modifications or insertions in the ResultSet, if it reflects in the database server,we call that concurrency as Concurrent Updatable.

**CONCURRENT\_READ\_ONLY:**

If we update/insert the data in the ResultSet object if it not updated in the database server,we call it as Concurrent ReadOnly.

* Following program demonstrates how to create bidirectional resultset.

|  |
| --- |
| import java.sql.\*;  public class ScrollableDemo{  public static void main(String ar[]) throws SQLException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  Statement stmt=con.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE,ResultSet.CONCUR\_READ\_ONLY);  ResultSet rs=stmt.executeQuery("select \* from student");  System.out.println(rs.last());  System.out.println(rs.next());  System.out.println(rs.next());  System.out.println(rs.first());  System.out.println(rs.previous());  System.out.println(rs.getRow());  }  } |

RetrieveRecords.java

o/p:

true

false

false

true

false

0

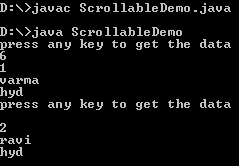
Note : some times when we try to retrieve the data from tables we may get an exception exhausted ResultSet, because we are trying to read the records which are not really available.

* Following program demonstrates use of sensitive resultset..

|  |
| --- |
| import java.sql.\*;  import java.io.\*;  public class ScrollableDemo{  public static void main(String ar[]) throws SQLException,IOException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  Statement stmt=con.createStatement(ResultSet.TYPE\_SCROLL\_SENSITIVE,ResultSet.CONCUR\_READ\_ONLY);  ResultSet rs=stmt.executeQuery("select sno,name,address from student");  while(rs.next()){  System.out.println("press any key to get the data");  System.in.read();  System.in.read();  rs.refreshRow();  System.out.println(rs.getString(1));  System.out.println(rs.getString(2));  System.out.println(rs.getString(3));  }  }  } |

RetrieveRecords.java

o/p:



Here in above program after displaying first record,if we go back to database and update any record that will be affected and we can see those changes in our result..so we call it as sensitive resultset.

In above program,we used a method refreshRow().this method will checks whether the data in the resultset and data in the database table are same or not.if it is not same it is the responsibility of jdbc driver to update the data in the resultset object. refreshRow() will work only for sensitive resultset.

* Java program to update the records in the databaseserver by using ResultSet object.(by using bidirectional resultset)

|  |
| --- |
| import java.sql.\*;  import java.io.\*;  public class ScrollableDemo{  public static void main(String ar[]) throws SQLException,IOException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  Statement stmt=con.createStatement(ResultSet.TYPE\_SCROLL\_SENSITIVE,ResultSet.CONCUR\_UPDATABLE);  ResultSet rs=stmt.executeQuery("select sno,name,address from student");  rs.absolute(2);  rs.updateString("name","name modified");  rs.updateRow();  }  } |

RetrieveRecords.java

When we run the above program,we are able to modify the data in the database server from the resultset.

When we try to use bidirectional resultset,there will be more load on jdbc drivers,,because of this reason most of the people doesn’t like to use bidirectional resultset..

We can also use the bidirectional resultset with PreparedStatement.

**BatchUpdates**

JDBC batch update is a collectively called when a group of SQL statements are executed simultaneously to a database as a single unit. The batch is sent to the database in a single request using connection object. Batchupdates improves the performance of java application.

The advantage of batch is to reduce the network calls between the front end application and its database rather than executing single SQL statement.

When do we use batchupdates::

If the java program woulf like to send the multiple queries to a database server,instead of sending the queries separately,we add the queries to batch object. And we send the batch to database server.

When we send the batch object to a database server,the db servercan execute all the queries in the batch.with this we can improve the performance of java application..

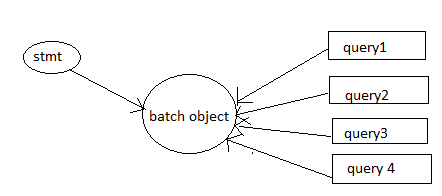
Procedure to use batchupdates :

* Register the jdbc driver
* Get the connection from db server
* Create the Statement object or PreparedStatement object(this will create the batch object)
* Add all the queries to batch object.
* Send the batch to db server
* Close the connection
* Program demonstrating use of batchupdates

|  |
| --- |
| import java.sql.\*;  import java.io.\*;  public class BatchDemo{  public static void main(String ar[]) throws SQLException,IOException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  Statement stmt=con.createStatement();  stmt.addBatch("insert into student values(10,'anil','hyd')");  stmt.addBatch("insert into student values(11,'gopi','hyd')");  stmt.addBatch("insert into student values(12,'ramana','hyd')");  stmt.addBatch("update student set name='maruthi' where sno=2");  stmt.executeBatch();  }  } |

BatchDemo.java

When the above java program is executed it ll create a statement object.now the internal code of jdbc driver will create the batch object and it ll be associated with a statement object.observe following diagram



]

The method executeBatch() returns integer array.

Int []a=executeBatch();

This array size is based on the number of queries send to databaseserver.this array contains the status of every query(i.e with the affect of query how many records are updated).

* Program

|  |
| --- |
| import java.sql.\*;  import java.io.\*;  public class ScrollableDemo{  public static void main(String ar[]) throws SQLException,IOException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  Statement stmt=con.createStatement();  stmt.addBatch("insert into student values(10,'anil','hyd')");  stmt.addBatch("insert into student values(11,'gopi','hyd')");  int []a=stmt.executeBatch();  for(int i=0;i<a.length;i++){  System.out.println(a[i]);  }  }  } |

ScrollableDemo.java

o/p: 1

1

**Transactions**

In a transaction,we can place a couple of steps in the form of a single unit.every transaction have two stages 1. Success and 2 failure.

We say a transaction is successful if all the steps are executed successfully,otherwise we say that transaction is failed..

Regarding database server by the time we starts communication with database server ,the transaction starts .we can end the transaction by using commit/rollback. Observer below program

|  |
| --- |
| import java.sql.\*;  import java.io.\*;  public class TransactionDemo{  public static void main(String ar[]) throws SQLException,IOException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  Statement stmt=con.createStatement();  stmt.executeUpdate("insert into student values(50,'varun','chennai')");  stmt.executeUpdate("insert into student values(51,'pavan','chennai')");  stmt.executeUpdate("insert into student values(52,'kiran','chennai')");  con.close();  }  } |

TransactionDemo.java

In the above program,whenever we establish the connection with db server,the jdbc driver starts the transaction.whenever the query is sent to the db server then jdbc driver ends transaction.by the time jdbc driver ends transaction ,it starts new transaction with the db server.

Above program inserted 3 records into db server in 3 different transactions..

**Userdefined Transactions**

When a connection is created using JDBC, by default it is in auto-commit mode. This means that each SQL statement is treated as a transaction and will be automatically committed immediately after it is executed. Sometimes, you want a group of statements to execute together or fail together. Transactions are used to group a set of statements so that they all execute successfully, or all fail. The way to allow two or more statements to be grouped into a transaction is to disable auto-commit mode.

The transactions which are controlled by the programmer/user is called userdefined transactions.in java application if we would like to strat userdefined transaction we use

con.setAutoCommit(false);

in java application by default jdbc driver autocommits the transaction.so the internal code of jdbc driver is con.setAutoCommit(true);

to end the userdefined transactions we use con.commit() or con.rollback();

* Program demonstrating userdefined transactions.

|  |
| --- |
| import java.sql.\*;  import java.io.\*;  public class UserDemo{  public static void main(String ar[]) throws SQLException,IOException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  con.setAutoCommit(false); //*starting userdefined transactions*  Statement stmt=con.createStatement();  stmt.executeUpdate("insert into student values(50,'varun','chennai')");  stmt.executeUpdate("insert into student values(51,'pavan','chennai')");  stmt.executeUpdate("insert into student values(52,'kiran','chennai')");  con.commit(); //*ending the userdefined transaction*  con.close();  }  } |

UserDemo.java

**MetaData**

Generally data about data is called as metadata.metadata gives mor information about actual data. In jdbc we have two metadata objects they are.

1. ResultSetMetaData
2. DatabaseMetaData

**ResultSetMetaData**

ResultSetMetaData is an object which gives more information about ResultSet object.it gives the information about number of columns we have retrieved from database table,names of columns which we have retrieved and datatypes of the columns..

Following example demonstrates how to get the metadata object and how do we find the number of columns and names of columns etc…

|  |
| --- |
| import java.sql.\*;  import java.io.\*;  public class ScrollableDemo{  public static void main(String ar[]) throws SQLException,IOException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  Statement stmt=con.createStatement();  ResultSet rs=stmt.executeQuery("select \* from student");  ResultSetMetaData rsmd=rs.getMetaData();  int count=rsmd.getColumnCount();  for(int i=1;i<=count;i++){  System.out.println(rsmd.getColumnName(i)+" "+rsmd.getColumnTypeName(i));  }  con.close();  }  } |

ScrollableDemo.java

o/p: SNO NUMBER

NAME VARVHAR2

ADDRESS VARCHAR2

**DatabaseMetaData**

DatabaseMetaData is used to find the information about underlying database server.to get the database metadata object,we use a method called as getMetaData().here we can use the methods like getDatabaseProductName(),getDatabaseMajorVersion() etc…

* Program

|  |
| --- |
| import java.sql.\*;  import java.io.\*;  public class MetaDataDemo{  public static void main(String ar[]) throws SQLException,IOException{  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  DatabaseMetaData dbmd=con.getMetaData();  System.out.println(dbmd.getDatabaseProductName());  System.out.println(dbmd.getDatabaseMajorVersion());  System.out.println(dbmd.getDatabaseMinorVersion());  System.out.println(dbmd.getDriverMajorVersion());  System.out.println(dbmd.getDriverMinorVersion());  System.out.println(dbmd.getDriverName());  con.close();  }  } |

MetaDataDemo.java

**Types of JDBC Drivers**

1. Type 1 driver (JDBC-ODBC bridge driver)
2. Type 2 driver (native API driver)
3. Type 3 driver (network or proxy driver)
4. Type 4 driver (thin or pure java driver)

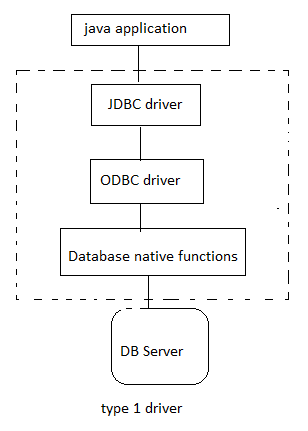
From sunmicrosystems we have 4 types of jdbc drivers.they are as mentioned above .of all those type 4 driver is the best driver.

**Type 1 driver** :

Below diagram represents the architecture of type 1 driver.here,JDBC driver,ODBC driver,Database native functions are called as type 1 driver.

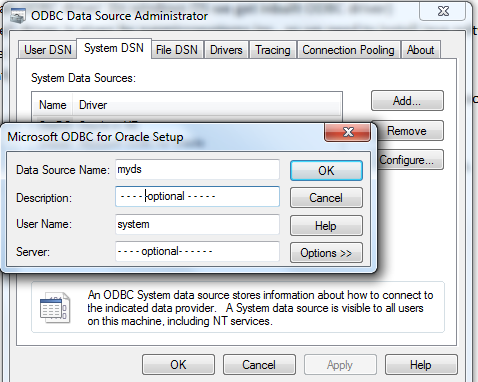
If we wan to develop java applications by using type 1 driver,we have to install client software.

In the client software we need to install odbc driver.and we make sure that jdbc driver is available in our computer.



**Working with type 1 driver**

1. Install client software
2. Install ODBC driver (in windows OS we get inbuilt ODBC driver)
3. Type1 driver is given by sunmicrosystems inc.. so we need to install java software to get type 1 driver.
4. Configuring type1 (JDBC-ODBC bridge driver):
5. open datasource (start----controlpanel----administrative tools---datasource)
6. from that window select tab system DSN
7. to add the new driver click on “add”
8. select the appropriate ODBC driver from the list of available drivers.
9. provide the following details in the driver configuration



While configuring datasource here I choosed driver “Microsoft ODBC for Oracle”,and supplied username in oracle.

Following is the java program which uses type 1 driver

|  |
| --- |
| import java.sql.\*;  public class StoreData  {  public static void main(String[] pppp)throws SQLException,ClassNotFoundException{  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  //geting connection  Connection con=DriverManager.getConnection("jdbc:odbc:myds","system","system");  String query="select \* from emp";  PreparedStatement pstmt=con.prepareStatement(query);  ResultSet rs=pstmt.executeQuery();  while(rs.next()){  System.out.print(rs.getString(1)+" "+rs.getString(2)+" "+rs.getString(3)+" ");  }  con.close();  }  } |

StoreData.java

Advantages:

By using type 1 driver ,we can communicate with any databse server.

Disadvantages:

Type 1 driver is platform dependent.

Type 1 driver will not give you the best performance.

**Type 2 driver :**

Before discussing about type 2 driver ,we required to know about JNI(Java Native Interface).

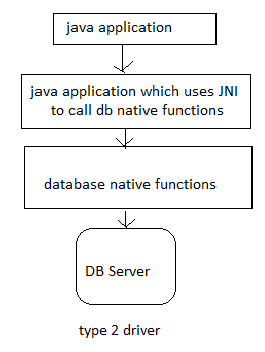
JNI (Java Native Interface) :

Why JNI ?

We use JNI to call non java functions(functions written in C/C++) from java application.

So type 2 drive ris used to call native functions from java application.

Following is the architecture of type 2 driver.

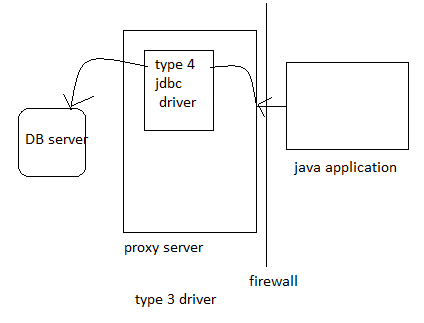


Disadvantages of type 2 driver :

We have to install client specific software.

This driver is platform dependent.

**Type 3 driver**

****

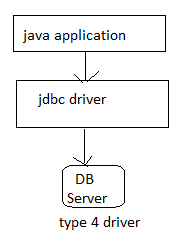
Advantages of type 3 driver:

By using type 3 driver we ll be able to get more security.

Type 3 driver is platform independent

Proxy servers are installed behind firewall..

**Type 4 driver**



Advantages of type 4 driver

Type 4 driver is developed in java . because of this reason type 4 driver is platform independent. In the client computer ,we no need to install db client softwares.most of the examples which we discusses in jdbc ,we used type4 driver.

**Servlets**

By using java we can develop 2 kinds of applications.

1. Standalone applications
2. Webbased applications

Standalone applications:

* These applications can be executed directly on to a computer.
* These applications are not dependent of any other softwares.
* Every standalone application contains main().

Disadvantages of standalone applications

* Standalone applications runs inside the client computer.we need to make sure that the software is installed before we used it.generally client will not aware of installation process
* Changes are so common in projects,whenever client /customer ask us for some modifications,we need to change it in all the client computers.
* The standalone applications utilizes the resources(RAM,processor,harddisk) of client computer

Webbased applications

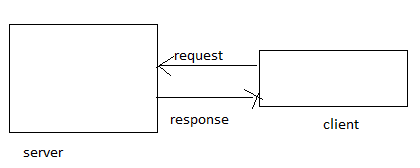
All internet and intranet applications are considered as webbased applications.

Advantages of webbased applications

* Webbased applications runs inside the server.
* The maintenance of the webbased applications is easy
* The webbsed applications uses resources of server computer.
* Webbased applications can be accessed from anywhere.

We develop webbased applications by using java,servlets,jsps …

In the webbased projects.The programs are place inside the server computer.if we want to execute the program,client has to send the request to server.then server takes the request and process the request and send the response back to client.



As a developer we will never develop a server program or the client program.we develop program which runs inside the server.There are so many server programs are already available in the market.some of them are

weblogic server

Websphere

Jboss

Tomcat server

Glashfish server etc…

There are somany client programs also available. Some of them are

Internet Explorer

Firefox

Opera

Chrome etc…

If the client and server would like to communicate with each other,they have to use dsame protocol. To develop webbased applications we use http protocol.

As server and client both uses same protocol http,whenever client wants to send request to server it uses “http request format”. Whenever server wants to send the response back to client it uses “http response format”.

Http protocol is mainly divided into two parts.

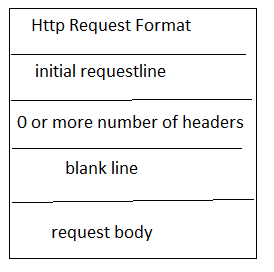
HttpRequestFormat.

HttpResponseFormat.

Client is responsible to send the user data to server by using HttpRequestFormat.in the following three scenarios client converts the user entered data into HttpRequest format.

1. Whenever user types the url and click on Enter it converts to HttpRequestFormat.
2. When the client clicks on a button in the browser,the request format will be sent to server.
3. When the user clicks on anchor tag HttpRequest format will be sent to server.

**HttpRequestFormat**



Initial Request Line is divided into 3 parts. They are

|  |  |  |
| --- | --- | --- |
| method | Requested resource URI | Protocol/version |

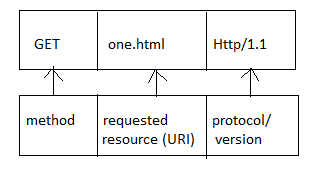
Here method can take multiple values like

GET,POST,PUT,TRACE,LOCATE,DELETE etc…

GET/POST: these methods indicates the server how to execute the resources in the server.every server supports these two methods.

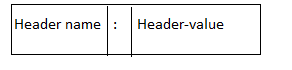
Rest of the methods are not used by the servers..

When the user sends request by entering URL ,the client always sends only get request..



Headers:

Every http request contains some headers and heders are used to send some extra information about the client to server. Following is the format of header

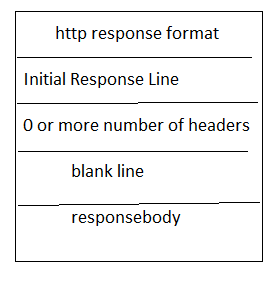


Some of the headers are

User\_agent : mozilla 14.0 accept\_language: telugu\_India

Blank line: to represent nextline we use \r\n. if we use this for 2 times we generate blankline.

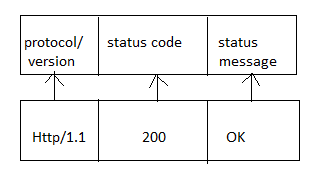
**HttpResponseFormat**



Initial response line:

|  |  |  |
| --- | --- | --- |
| Protocol/version | Status code | Status message |

In initial request line protocol/version indicates the version information of server.



Server uses different status code to represent whether the request is successful or not. The following are the status codes.

1XX - information

2XX -success message

3XX - redirect

4XX- requested resource is not found

5XX -failed to execute the requested the requested resource.

Headers:

Server also sends some headers to the client.these are used to supply more information to the client .one important header sent by the server to client is ContentType.

Eg: ContentType=text/html.

Based on the content type browser will display the data.

Differences between get and post methods?

These two methods are used to send the request to server for processing the resource.

If we use get method the data entered by client is appends to URL and send to server.

Eg:

<http://localhost:8000/stwo.html?uname=abc&password=xyz>

if we use post method data will be appended to request body and send to server.

Eg:

…….post|two.html|http/1.1

Headers

\_\_\_\_\_\_\_

uname=abc& password=xyz

webbased applications are portable ..once if we develop a webbased application,we can deploy them on any server..

**creating webbased applications**

* create a folder with the project name

projectname---myapp

* create a folder with the name WEB-INF inside the above created folder.here WEB-INF is a private folder.
* Inside WEB-INF create folders with the name classes & lib
* Create a file with the name web.xml

Eg:

|  |
| --- |
| <web-app>  </web-app> |

web.xml

here web.xml is called as deployment descriptor.usually a webapplication is a set of resources.

Resources: .html,.jsp,images,javascript,vbscript etc… files are called as resources.

Assume in the above created project we have placed two html files named one.html and two.html.

We can download tomcat server from apache.org url. Incase of tomcat server deploying project means we have to place the project in a folder named **webapps** in the server.after deploying the project we need to perform unit testing.

Following is the URL used for unit testing.

[*http://localhost:8000/myapp/one.html*](http://localhost:8000/myapp/one.html)

if we place any resources inside WEB-INF only server can access it. We cannot access the files inside the WEB-INF from browser/client. Here we can develop two kinds of webapplications they are

* static webapplications
* dynamic webapplications

html files are used to develop static webapplications.

Servlets,jsps are used to develop dynamic webapplications.

**Servlet :** Servlet is an API used to develop webbased applications.the predefined classes and interfaces of Servlet API are place in two different packages

javax.servlet & javax.servlet.http

following are the most important interfaces of javax.servlet package.

Servlet

ServletRequest

ServletResponse

ServletConfig

ServletContext

What is a Servlet?

Servlet is aprogram which provides implementation of Servlet interface directly or indirectly..Servlet interface has the following 5 abstract methods

|  |
| --- |
| javax.servlet.Servlet |
| public void init(ServletConfig config)  public void service(ServletRequest,ServletResponse)  public void destroy()  public ServletConfig getServletConfig()  public String getServletInfo() |

Procedure to develop a Servlet:

1. develop a java program which provides the implementation of Servlet interface

|  |
| --- |
| import javax.servlet.\*;  public class FirstServlet implements Servlet{  ServletConfig config;  public void init(ServletConfig config){  this.config=config;  System.out.println("init() called");  }  public void service(ServletRequest request,ServletResponse response){  System.out.println("service() called");  }  public void destroy(){  System.out.println("destroy() called");  }  public ServletConfig getServletConfig(){  return config;  }  public String getServletInfo(){  return " ";  }  } |

FirstServlet.java

1. to compile the above program we have to set the CLASSPATH to a jar file which provides the implementation of Servlet API.(in tomcat server it is servlet-api.jar).
2. copy the .class into classes folder of our project.
3. Configure the servlet in deployment descriptor (web.xml)

|  |
| --- |
| <web-app>  <servlet>  <servlet-name>firstservlet</servlet-name>  <servlet-class>FirstServlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>firstservlet</servlet-name>  <url-pattern>/fs</url-pattern>  </servlet-mapping>  </web-app> |

1. Deploy the project and perform unit testing.

whenever we deploy the above project and send the request to server.server has created the servlet object.

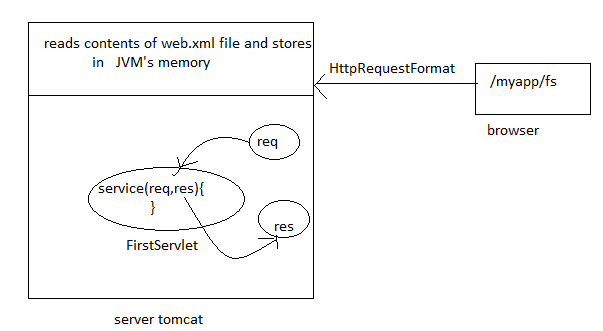
When the servlet object is created servlet calls init(). After init() is executed server calls the service().(this will happen when the client send the request for the first time). When the client send the request for the second time onwards every time server executes service() only..when we undeploy the project server executes destroy() method.

Whenever we deploy the project server reads the contents from web.xml file and stores the information in JVM’s memory.to read the contents of xml file server uses parser programs.there are two different kinds of parsers are available SAX parser and DOM parser.the parser can read the contents of xml file if it is a valid xml file, otherwise parser displays error message.

What will happen when the client sends the request to server :

* When the client enter the URL and click on enter the browser converts the request into HttpRequest format and send it to server.
* When the server receives the request,server will create request object and respons eobject. Now server reads the contents from requestformat and place inside the request object.
* Now server opens request object and gets the URL and check whether it is configured in web.xml file or not. If it is not configured in web.xml file server sends error message to the client.if it configured in web.xml server get the servlet class name & if required server creates servlet object.
* When server creates servlet object it executes init() method.every time client sends request,it keep on executes service() method.
* After service() executed successfully server send response back to client,it removes request object and response object.

Multiple clients can send the request to the server simultaneously..it is the responsibility of the server to handle multiple requests at the same time. So every server internally contains thread pool. Whenever client send the request to a server, server picks a thread from threadpool and execute the servlet program.while developing servlet class we need to make sure that servlet class must be public.if we doesn’t create a servelt with a public keyword it displays javax.servlet.ServletException(cannot instantiate class).



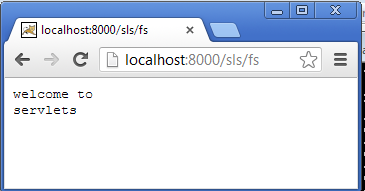
It is always recomonded to develop a servlet program which contains name as servlet.always we have to provide the business logic in service().if the server wants to send the output to client the servelt has to write the content to response object.it is the responsibility of server to get the data from response object and send it to client.

Develop a program which runs inside the server and send output to client.

|  |
| --- |
| import javax.servlet.\*;  public class FirstServlet implements Servlet{  ServletConfig config;  public void init(ServletConfig config){  this.config=config;  System.out.println("init() called");  }  public void service(ServletRequest request,ServletResponse response){  System.out.println("service() called");  PrintWriter out=response.getWriter();  out.println("welcome to ");  out.println("servlets");  }  public void destroy(){  System.out.println("destroy() called");  }  public ServletConfig getServletConfig(){  return config;  }  public String getServletInfo(){  return "";  }  } |

FirstServlet.java

o/p:



Below class is belongs to javax.servlet package.

javax.servlet.GenericServlet

following are the most important classes &interfaces of javax.servlet .http package.

Interfaces:

HttpServletRequest

HttpServletResponse

Classes:

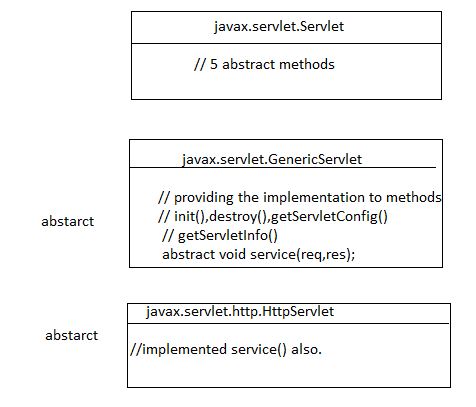
HttpServlet

Following are the UML diagrams describing these two packages.

By looking at below diagram we can say that there are three ways are there to develop servelts.

1. By implementing Servlet interface
2. By using GenericServlet
3. HttpServlet

Not only these 3 in no of ways we can develop servlts..



* Program which display welcome message using **GenericServlet** class

|  |
| --- |
| import javax.servlet.\*;  import java.io.\*;  public class FirstServlet extends GenericServlet{  public void service(ServletRequest request,ServletResponse response) throws IOException{  System.out.println("service() called");  PrintWriter obj=response.getWriter();  obj.println("welcome");  obj.println("o/p from FirstServlet");  }  } |

FirstServlet.java

* Program which display welcome message using **HttpServlet** class.

|  |
| --- |
| import javax.servlet.http.\*;  import java.io.\*;  public class FirstServlet extends HttpServlet {  public void service(HttpServletRequest request,HttpServletResponse response) throws IOException{  System.out.println("service() called");  PrintWriter obj=response.getWriter();  obj.println("welcome");  obj.println("o/p from FirstServlet");  }  } |

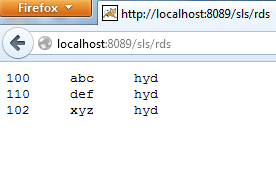
FirstServlet.java

* Write a servlet program which can retrieve records from employee table of Database server and display to client.

|  |
| --- |
| import javax.servlet.http.\*;  import java.sql.\*;  import java.io.IOException;  import java.io.PrintWriter;  public class RetrieveDataServlet extends HttpServlet{  public void service(HttpServletRequest request,HttpServletResponse response) throws IOException{  PrintWriter out=response.getWriter();  try{  Class.forName("oracle.jdbc.driver.OracleDriver");  Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");  Statement stmt=con.createStatement();  ResultSet rs=stmt.executeQuery("select \* from employee");  while(rs.next()){  out.print(rs.getInt(1)+"\t");  out.print(rs.getString(2)+"\t");  out.print(rs.getString(3)+"\n");  }  }  catch(SQLException se){  se.printStackTrace();  }  catch(ClassNotFoundException cne){  cne.printStackTrace();  }  }  } |

RetrieveDataServlet.java

o/p:

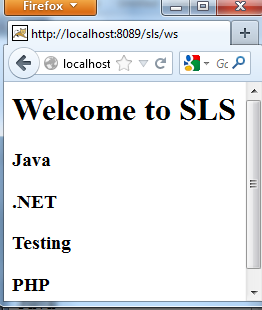


Develop a servlet program which sends html document to the client

|  |
| --- |
| import javax.servlet.http.\*;  import java.io.\*;  import java.util.\*;  public class WelcomeServlet extends HttpServlet{  public void service(HttpServletRequest request,HttpServletResponse response) throws IOException{  PrintWriter out=response.getWriter();  out.println("<html>");  out.println("<body>");  out.println("<h1 align=\"left\">Welcome to SLS</h1>");  out.println("<h3 align=\"left\">Java</h3>");  out.println("<h3 align=\"left\">.NET</h3>");  out.println("<h3 align=\"left\">Testing</h3>");  out.println("<h3 align=\"left\">PHP</h3>");  out.println("</body>");  }  } |

WelcomeServlet.java

o/p:



The above WelcomeServlet is sending static output to the client .by using servlets we can develop static as well as dynamic webbased applications. It is not recomonded to develop a servlet which display static contents to the clients.

The disadvantages of servlets which send static output to the clients.

* The maintenance of the project becomed difficult.
* It consumes lot of resources of the server

It is always recomonded that servlets must be used to send only dynamic contents.

Develop a dynamic servlet which sends current data and time of the server to client.

|  |
| --- |
| import javax.servlet.http.\*;  import java.io.\*;  import java.util.\*;  public class WelcomeServlet extends HttpServlet{  public void service(HttpServletRequest request,HttpServletResponse response) throws IOException{  PrintWriter out=response.getWriter();  Date d=new Date();  out.println(d);  }  } |

o/p:

displays current date of the system.

Whenever server tries to create the object to any class,it has to load the class into JVM’s memory.incase of webbased applications ,server tries to check for .class file first in classes folder,if it is not available it checks in project lib folder(inside jar files).if it is not available in project lib folder,it checks in server lib folder.if the class is available in server lib folder,it uses it,otherwise ,it send error message to the client saying NoClassDefFoundError…

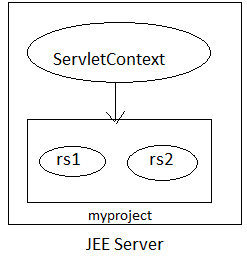
In the above program which retrieves the data from dbserver,we hardcoded the driverclass name,url,username,password..it is not recomonded to hardcode the values.to remove hardcoding in webbased applications we can use

1. ServletContext
2. ServletConfig

**ServletContext:**

ServletContext is an object created by server ,when we try to deploy the project.i.e by the time we deploy the project ,server creates ServletContext object..

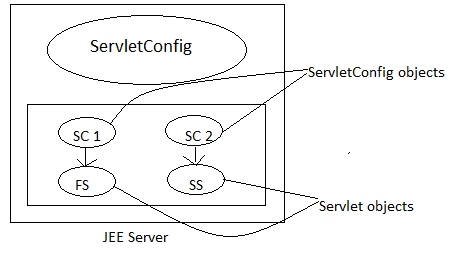
Every webapplication contains one ServletContext object.



* When we deploy a webbased application.server creates ServletContext object.
* When we undeploy the project,server removes ServletContext object.
* We can store the data into ServletContext object,once if we store the data into ServletContext object,all the resources of the project can access the data.
* Here data available in the ServletContext can be accessed by only the resources of that project,the other projects cannot access the data available in the ServletContext object…

**ServletConfig**

* Server creates the ServletConfig object,at the time of Servlet object is created.
* ServletConfig object are associated with servlet objects.
* If we store the data into ServletConfig object,only that particular servlet can access the data in the servletconfig object,other servlets cannot access ..



Form the above diagram we can say that for every servlet we have one ServletConfig object..

If we want to store the data in the Servletconfig object we need to configure them in web.xml file as shown below…

|  |
| --- |
| <web-app>  <servlet>  <servlet-name>fs</servlet-name>  <servlet-class>FirstServlet</servlet-class>  <init-param>  <param-name>username</param-name>  <param-value>abc</param-value>  </init-param>  <init-param>  <param-name>password</param-name>  <param-value>xyz</param-value>  </init-param>  </servlet>  </web-app> |

web.xml

Program for getting data available in the ServletConfig object..

|  |
| --- |
| import javax.servlet.http.\*;  import javax.io.PrintWriter;  public class SecondServlet extends HttpServlet throws IOException{  ServletConfig config=getServletConfig();  String uname=config.getParameter("uname");  String pwd=request.getParameter("pwd");  Printwriter out=response.getWriter();  out.println("username "+uname);  out.println("password "+pwd);  } |

SecondServlet.java

In the above program,when we call getServletConfig(),this method will be executed from GenericServlet.

once if we get ServletConfig object,to read the contents of ServletConfig object,we use a method getInitParameter().

String uname=config.getInitParameter(“uname”);

When the above java code is executed,it tries to get data from config object,whose key name is uname.if the data is available in the config object,it will return the data otherwise it ll return null value.the method getInitParameter() returns data in the form of String..

To store the data into ServletContext object,we use a tag <context-param> in web.xml.

Here ServletContext is accessible throughout the project.so to store the data in the ServletContext object,we need to configure the tag outside the <servlet> in web.xml.

|  |
| --- |
| <web-app>  <servlet>  <servlet-name>fs</servlet-name>  <servlet-class>FirstServlet</servlet-class>  </servlet>  <context-param>  <param-name>driver</param-name>  <param-value>oracle.jdbc.driver.OracleDriver</param-value>  </context-param>  <context-param>  <param-name>url</param-name>  <param-value>jdbc:oracle:thin:@localhost:1521:xe</param-value>  </context-param>  </web-app> |

web.xml

To read the values from Servletcontext object,we need to get the ServletContext object.(which is created at the time of deploying the project)

To get the SerlvetContext object,we have two ways.

* By using ServletConfig
* By calling getServletContext() of GenericServlet class

Once if we store the data into ServletContext object,we can read the data from the ServletContext by using getInitParameter().

|  |
| --- |
| import javax.servlet.http.\*;  import javax.io.PrintWriter;  public class SecondServlet extends HttpServlet throws IOException{  ServletConfig config=getServletConfig();  ServletContext aplication=config.getServletContext();  String driver=application.getParameter("uname");  String url=application.getParameter("pwd");  Printwriter out=response.getWriter();  out.println("driver class "+driver);  out.println("url "+url);  } |

SecondServlet.java

Whenever we would like to send output to the client,it is recomonded to provide the contentType . based on the contentType header,the client will render the output..to set the contentType we use a method setContentType();

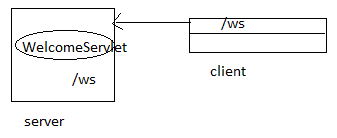
response.setContentType(“text/html”);

or

response.setContentType(“application/img”);

to call the servlet available in the server (Which we deployed in the server),we have several options..

* In the browser address bar type the URL and click on enter. Now the request goes to server and server executes servlet.



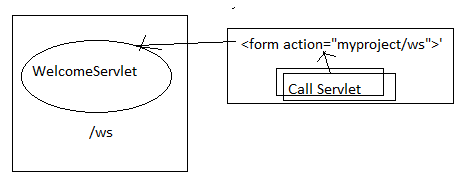
* By using the html form submit buttons,we can execute the servlet available in server.for this take an html form..

<form action="/myproject/ws">

<input type="submit" value="Call Servlet"/>

</form>

When the user clicks on Call Servlet in the browser,browser checks what is the value of action attribute ,it gets the action attribute value and store in HttpRequestformat and send it to server..



* By using anchor tag,we can call the resources available in the server..

<a href=”/myproject/ws”> Call Servlet</a>

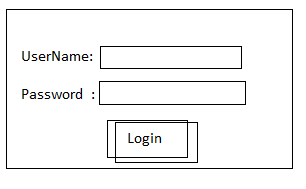
* We can call the resources available in the server by using user defined buttons.

**Developing formbased applications:**

To develop web project with form based applications,before starte developing the actual project,we need to deliver the prototype of the project to customer.

Prototype: prototype is a dummy project with set of html files.

Consider the following formbased application:



To develop the above form based application,we have two approaches,they are

Approach 1:

developing two programs.

* + Html file(to display the form)
  + Servlet (to interact with DBServer).

Approach 2:

developing two programs

* + Servlet (to display the form)
  + Servlet(to interact with DB server)

Approach 1:

Step1:develop the html file to display the form

|  |
| --- |
| <html>  <body>  <form action="/myproject/ls">  Username:<input type="text" name="uname"/><br>  Password:<input type="password" name="pwd"/><br>  <input type="submit" value="Login"/>  </form>  </body>  </html> |

login.html

Step2: develop a servlet which captures the data from the client.

|  |
| --- |
| import javax.servlet.http.\*;  import java.io.\*;  public class LoginServlet extends HttpServlet{  public void service(HttpServletRequest request,HttpServletResponse response)throws IOException{  String uname=request.getParameter("uname");  String pwd=request.getParameter("pwd");  PrintWriter out=response.getWriter();  if((uname.equals("abc")||uname=="abc")&&(pwd.equals("xyz")||pwd=="xyz")){  out.print("welcome "+uname);  }  else{  out.println("you are not valid user");  }  }  } |

LoginServlet.java

Step3:configure the LoginServlet in web.xml and deploye the project then perform the unit testing.

Approach 2:

In this approach we need to develop two servlets,the following is the servlet which generates html content and send it to the client.

|  |
| --- |
| import javax.servlet.http.\*;  import java.io.\*;  import java.util.\*;  import java.text.\*;  public class SearchServlet extends HttpServlet{  public void service(HttpServletRequest request,HttpServletResponse response) throws IOException{  response.setContentType("text/html");  PrintWriter out=response.getWriter();  out.println("<html>");  out.println("<body>");  out.println("<form action=\"/sls/ls\">");  out.println("Username:<input type=\"text\" name=\"uname\"/><br>");  out.println("Password:<input type=\"password\" name=\"pwd\"/><br>");  out.println("<input type=\"submit\" value=\"Login\"/>");  out.println("</form>");  out.println("</body>");  out.println("</html>");  }  } |

SearchServlet.java

Now develop another servlet to handle data given by this servlet..

Here approach 2 is not at all recomonded to use .because it is not recomonded to use servlets to generate static output.

Usually we use the URL like



By default,above URL looks for index.html but we would like to use our own html files or

Jsps as the default files. To use our own files as default, we have to configure them in a deployment descriptor as shown below.

|  |
| --- |
| <web-app>  <welcome-file-list>  <welcome-file>login.html</welcome-file>  <welcome-file>one.html</welcome-file>  <welcome-file>two.html</welcome-file>  <welcome-file>search.html</welcome-file>  </welcome-file-list>  </web-app> |

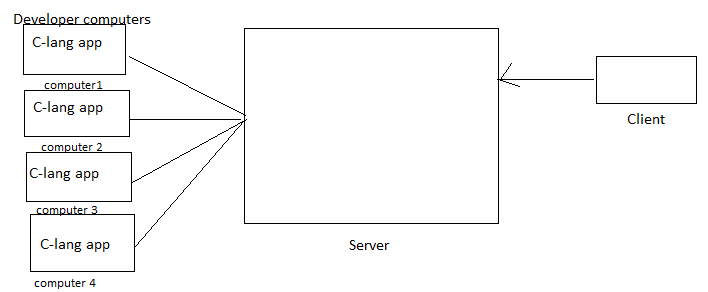
web.xml

After developing a project or project files, we need to test the applications whether they working properly or not. There are different servers are available. They are

* Development server: this server is used by developers to perform unit testing.
* Integration server: this server is used by developers to integrate different modules and test it.
* QA server: this server is used by testers to perform system testing
* Production server: this server is used by the clients/customers to use the project

**CGI (Common Gateway Interface):**

CGI is used to develop webbased applications prior to servelts and jsps.



* The CGI applications are developed in C-language and deployed it on normal developer computers.
* These developer computers are connected with server computer.
* When the client send the request to the server server takes the request and process the request.(here server will handover the request to developer computers .the developer computers will execute the program and send the response back to the server. It is the responsibility of server to send the output to the client).
* There are so many disadvantages of CGI applications.
* CGI applications are platform dependent
* CGI applications are very slow in application processing.
* In CGI applications ,they start a separate process for every request.
* Sunmicro systems has released Servlet API to resolve all the problems of CGI technology.

# Advantages of Servlets over CGI

      Servlets are server side components that provides a powerful mechanism for developing server web applications for server side. Earlier CGI was developed to provide server side capabilities to the web applications. Although CGI played a major role in the explosion of the Internet, its performance, scalability and reusability issues make it less than optimal solutions. Java Servlets changes all that. Built from ground up using Sun's write once run anywhere technology java servlets provide excellent framework for server side processing.

Using servlets web developers can create fast and efficient server side applications and can run it on any servlet enabled web server. Servlet runs entirely inside the Java Virtual Machine. Since the servlet runs on server side so it does not depend on browser compatibility.

Servlets have a number of advantages over CGI and other API's. They are:

1. **PlatformIndependence**  
   Servlets are written entirely in java so these are platform independent. Servlets can run on any Servlet enabled web server. For example if you develop an web application in windows machine running Java web server, you can easily run the same on apache web server (if Apache Serve is installed) without modification or compilation of code. Platform independency of servlets provide a great advantages over alternatives of servlets.
2. **Performance**  
   Due to interpreted nature of java, programs written in java are slow. But the java servlets runs very fast. These are due to the way servlets run on web server. For any program initialization takes significant amount of time. But in case of servlets initialization takes place first time it receives a request and remains in memory till times out or server shut downs. After servlet is loaded, to handle a new request it simply creates a new thread and runs service method of servlet. In comparison to traditional CGI scripts which creates a new process to serve the request.
3. **Extensibility**  
   Java Servlets are developed in java which is robust, well-designed and object oriented language which can be extended or polymorphed into new objects. So the java servlets take all these advantages and can be extended from existing class to provide the ideal solutions.
4. **Safety**  
   Java provides very good safety features like memory management, exception handling etc. Servlets inherits all these features and emerged as a very powerful web server extension.
5. **Secure**  
   Servlets are server side components, so it inherits the security provided by the web server. Servlets are also benefited with Java Security Manager.

There are so many servers are available in the market.they are

* HTTP Servers
* FTP servers
* SMTP Servers
* TCP/IP Servers
* Tomcat,Weblogic.WebSphere,Jboss are HttpServers
* Http Servers: the servers which are built based on http protocol are called as HttpServers.
* When the sunmicro systems released Servlet API,they said servlets can be run on any servers.(FTP servers,SMTP servers,HTTP servers etc..). but practically servlet will run on only HttpServers.
* Protocols are classified into two ways.
  1. Stateful protocol
  2. Stateless protocol
* Statefull protocols remembers all the conversation happening to the client

Eg: Tcp/IP,SMTP,FTP protocols

* Stateless protocols remembers only the last conversation which is happening to the client.

Eg: http protocol

* Statefull protocols requires huge amount of memory for remembering the conversation.
* As tomcat,weblogic are built based on http protocols,we call these servers as stateless servers.

**doXXX() methods:**

if we develop an application as method specific i.e application supports either get request or post request.then we use doXXX() methods.

The following servlet shows use of doGet() and doPost() .

|  |
| --- |
| import javax.servlet.http.\*;  import java.io.\*;  public class SearchServlet extends HttpServlet{  public void doGet(HttpServletRequest request,HttpServletResponse response) throws IOException{  PrintWriter out=response.getWriter();  out.println("output from doGet()");  }  public void doPost(HttpServletRequest request,HttpServletResponse response) throws IOException{  PrintWriter out=response.getWriter();  out.println("output from doPost()");  }  } |

SearchServlet.java

**Take the input form as**

Form 1:

|  |
| --- |
| <html>  <body>  <form action="/sls/sss" method="post">  Username:<input type="text" name="uname"/><br>  Password:<input type="password" name="pwd"/><br>  <input type="submit" value="Login"/>  </form>  </body>  </html> |

form1.html

If we send the request from Form1. The output is

*output from doPost()*

Form2:

|  |
| --- |
| <html>  <body>  <form action="/sls/sss" method="get">  Username:<input type="text" name="uname"/><br>  Password:<input type="password" name="pwd"/><br>  <input type="submit" value="Login"/>  </form>  </body>  </html> |

form2.html

If we send the request from Form2 . the output is

*output from doGet().*

The following servlet shows using doGet(),doPost() methods to work with all types of methods.

|  |
| --- |
| import javax.servlet.http.\*;  import java.io.\*;  public class SearchServlet extends HttpServlet{  public void doGet(HttpServletRequest request,HttpServletResponse response) throws IOException{  doWork(request,response);  }  public void doPost(HttpServletRequest request,HttpServletResponse response) throws IOException{  doWork(request,response);  }  public void doWork(HttpServletRequest request,HttpServletResponse response) throws IOException{  PrintWriter out=response.getWriter();  out.println("output from doPost()");  }  } |

SearchServlet.java

**Preinitialization**

In general when the servlet object for the servlet which is configured in the web.xml file can be created when we send the request to the servlet for the first time.

If we want the server to create objects for servlets when you deploy the project itself,use a tag

<load-on -startup> as part of servlet tag.(for which servlet you want to create the object at the time of deployment).

Eg:

|  |
| --- |
| <servlet>  <servlet-name>abc</servlet-name>  <servlet-class>LoginServlet</servlet-class>  <load-on-startup>4</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>abc</servlet-name>  <url-pattern>/ls</url-pattern>  </servlet-mapping> |

web.xml

**Listeners**

In case of awt&swings applications when the user performs any aoperation an event will be raised by the application. When the event is raised ,the application checks for appropriate event handler.

* If the event handlers are available application executes.
* If event handlers are not available application will not do any thing.

Incase of webbased applications server raises events when the objects created or modified.

Majorly there are three listeners are available in servlets.. they are

1. ServletContextListener
2. ServletRequestListener
3. HttpSessionListener

The following 3 listeners are used whenever any operation performed on objecti. When we call setAttribute(),getAttribute(),removeAttribute().

* ServletContextAttributeListener
* ServletRequestAttributeListener
* HttpSessionAttributeListener

**ServletContextListener**

If you want to perform any operations when ServletContext object is created or destroyed,we need to implement this interface.ServletContextListener has two methods.observe below example.

public void contextInitialized(ServletContextEvent sce);

public void contextDestroyed(ServletContextEvent sce);

|  |
| --- |
| import javax.servlet.http.\*;  import javax.servlet.\*;  import java.io.\*;  public class ContextListener extends HttpServlet implements ServletContextListener{  public void contextInitialized(ServletContextEvent sce){  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* contextinitialized\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  }  public void contextDestroyed(ServletContextEvent sce){  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*contextdestroyed\*\*\*\*\*\*\*\*\*\*\*\*\*");  }  } |

ContextListener.java

* To configure listener in web.xml we are using atag listener-class

|  |
| --- |
| <web-app>  <listener>  <listener-class>ContextListener</listener-class>  </listener>  </web-app> |

web.xml

**ServletRequestListener**

If you want to perform any operations when ServletRequest object is created or destroyed,we need to implement this interface.ServletRequestListener has two methods.observe below example.

public void requestInitialized(ServletRequestEvent sre);

public void requestDestroyed(ServletRequestEvent sre);

|  |
| --- |
| import javax.servlet.http.\*;  import javax.servlet.\*;  import java.io.\*;  public class RequestListener extends HttpServlet implements ServletRequestListener{  public void requestInitialized(ServletRequestEvent sce){  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* requestinitialized\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  }  public void requestDestroyed(ServletRequestEvent sce){  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*requestdestroyed\*\*\*\*\*\*\*\*\*\*\*\*\*");  }  } |

RequestListener.java

**HttpSessionListener**

If you want to perform any operations when HttpSession object is created or destroyed,we need to implement this interface.HttpSessionListener has two methods.observe below example.

public void sessionCreated(HttpSessionEvent hse);

public void sessionDestroyed(HttpSessionEvent hse);

the following code demonstrates how to find number of users who has visited the website.

|  |
| --- |
| package info.fls.sls;  import javax.servlet.http.\*;  import javax.servlet.\*;  import java.io.\*;  public class SessionListener extends HttpServlet implements HttpSessionListener{  private static int count;  public void sessionCreated(HttpSessionEvent hse){  count++;  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* session created\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  }  public void sessionDestroyed(HttpSessionEvent sce){  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*sessiondestroyed\*\*\*\*\*\*\*\*\*\*\*\*\*");  }  public static int getCount(){  return count;  }  } |

SessionListener.java

Configure the SessionListener in web.xml

|  |
| --- |
| <web-app>  <listener>  <listener-class>info.fls.sls.SessionListener</listener-class>  </listener>  </web-app> |

web.xml

the following is the jsp which display number of users.

|  |
| --- |
| <%@ page import="info.fls.sls.\*"%>  visitors <%=SessionListener.getCount()%> |

one.jsp

**Filters**

Filter is a java program,which provides implementation of Filter interface directly or indirectly.

The Filter interface contains the following lifecycle methods.

* public void init(FilterConfig config)
* public void doFilter(ServletRequest request,ServletResponse response,FilterChain chain)
* destroy()

FilterConfig is an object which will be created at the time of FilterObject is created.by using FilterConfig object,we can remove hardcoding and we can also get Servletcontext object.

Following is the sample Filter

|  |
| --- |
| import javax.servlet.\*;  public class FirstFilter implements Filter{  FilterConfig config;  public void init(FilterConfig config){  this.config=config;  System.out.println("init() called");  }  public void doFilter(ServletRequest request,ServletResponse response,FilterChain chain){  System.out.println("foFilter() called");  }  public void destroy(){  System.out.println("destroy() called");  }  } |

FirstFilter.java

* whenever we deploy the project it is the responsibility of server to create FilterObject and call init().
* For every client request server keep on executing doFilter().
* When we undeploy the project and stop the server,then server removes Filter object.
* To resolve the problems of servlet as controller sunmicrosystems has introduced Filter .

So we can use Filter as controller.

* As the Filter acts as controller the url pattern of Filter must be /\*.
* We can configure filters in web.xml file as shown below.

|  |
| --- |
| <web-app>  <filter>  <filter-name>aa</filter-name>  <filter-class>FirstFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>aa</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  </web-app> |

web.xml

**FilterChain**

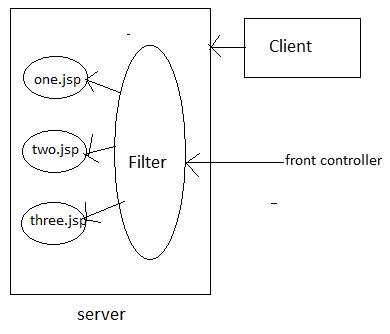
Filterchain is an object which is used to call another filter.FilterChain interface contains a method doFilter().

*void doFilter(ServletRequest request,ServletResponse response);*

**what does chain.doFilter do?**

When we call chain.doFilter() it will checkare there any Filters are configured in web.xml .if they are available the request will be handovered to another Filter,if next Filter is not available,it will check to which resource the client has sent request…if that resource is available it will handover the request to that resource.We can change the order of Filter execution by configuring the Filter,the order in which you want to execute the filters.

Any code which you want to execute by all the requests ,we can provide that code as part of filter.when the client send request for any resource in the server ,the request then go through filter.



The following is the LoginFilter,which checks whether user has entered valid username and password.

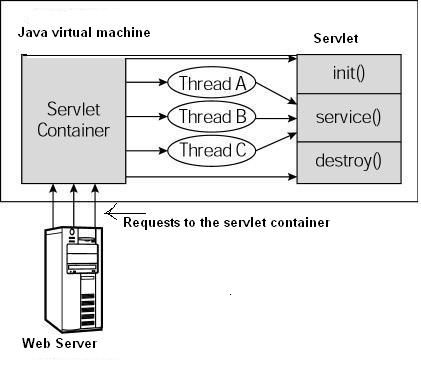
|  |
| --- |
| import javax.servlet.\*;  import javax.servlet.http.\*;  import javax.servlet.jsp.\*;  import java.io.\*;  public class LoginFilter implements Filter{  FilterConfig config;  static int count=0;  public void init(FilterConfig config){  this.config=config;  }  public void doFilter(ServletRequest req,ServletResponse res,FilterChain chain) throws IOException,ServletException{  System.out.println("in doFilter()");  HttpServletRequest request=(HttpServletRequest)req;  HttpServletResponse response=(HttpServletResponse)res;  HttpSession session=request.getSession(false);  if(session!=null){  chain.doFilter(req,res);  }  else{  String uname=request.getParameter("uname");  String pwd=request.getParameter("pwd");  if((uname==null || uname.equals(""))||( pwd==null) ||pwd.equals("")){  displayForm(request,response);  }  else{  if(uname.equals("xyz")&&pwd.equals("abc")){  session=request.getSession(true);  chain.doFilter(req,res);  }  else{  request.setAttribute("failed","username &password are invalid");  displayForm(request,response);  }  }  }  }  public void displayForm(HttpServletRequest request,HttpServletResponse response) throws IOException,ServletException{  String fail=(String)request.getAttribute("failed");  PrintWriter out=response.getWriter();  System.out.println("in displayForm()");  response.setContentType("text/html");  if(fail!=null){  out.println("error!! usernme and password invalid");  RequestDispatcher rd=request.getRequestDispatcher("/input.html");  rd.include(request,response);  }  else{  System.out.println("in displayForm() else");  if(count==0){  count++;  RequestDispatcher rd=request.getRequestDispatcher("/input.html");  rd.include(request,response);  }  else{  out.println("username and password required");  RequestDispatcher rd=request.getRequestDispatcher("/input.html");  rd.include(request,response);  }  }  }  public void destroy(){  }  } |

LoginFilter.java

**LIFE CYCLE OF SERVLET**

The life cycle of a servlet can be categorized into four parts:

1. **Loading and Instantiation:**The servlet container loads the servlet during startup or when the first request is made. The loading of the servlet depends on the attribute <load-on-startup> of web.xml file. If the attribute <load-on-startup> has a positive value then the servlet is load with loading of the container otherwise it load when the first request comes for service. After loading of the servlet, the container creates the instances of the servlet.
2. **Initialization:**After creating the instances, the servlet container calls the init() method and passes the servlet initialization parameters to the init() method. The init() must be called by the servlet container before the servlet can service any request. The initialization parameters persist untill the servlet is destroyed. The init() method is called only once throughout the life cycle of the servlet.  
     
   The servlet will be available for service if it is loaded successfully otherwise the servlet container unloads the servlet.
3. **Servicing the Request:** After successfully completing the initialization process, the servlet will be available for service. Servlet creates seperate threads for each request. The sevlet container calls the service() method for servicing any request. The service() method determines the kind of request and calls the appropriate method (doGet() or doPost()) for handling the request and sends response to the client using the methods of the response object.
4. **Destroying the Servlet:**If the servlet is no longer needed for servicing any request, the servlet container calls the destroy() method . Like the init() method this method is also called only once throughout the life cycle of the servlet. Calling the destroy() method indicates to the servlet container not to sent the any request for service and the servlet  releases all the resources associated with it. Java Virtual Machine claims for the memory associated with the resources for garbage collection.



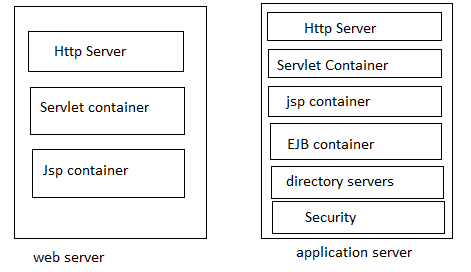
**JSP(Java Server Pages)**

* Jsp stands for java server pages.
* Jsps are also used to develop webbased applications
* Sunmicro systems has given jsps as alternative to servlets.
* Jsps’s improves the productivity of the project(we can deliver the projects quickly to the customers).
* We can develop the jsps without java code.
* If we carefully design the project,by using jsps,we can seperateout business logic and presentation logic.
* Every jsp program must end eith extension .jsp
* We have to plca the jsp files inside the project folder and outside the WEB-INF folder.

Generally as part of our webservers they have provided the implementation to Servlet API and Jsp API . they are called as containers. The containers are placed inside the server.

Generally the servers are classified into two categories.

* Web servers
* Application servers



Application servers contain more features like EJB container,directory servers,various security concerns etc..

# [JSP ARCHITECTURE](http://www.roseindia.net/jsp/jsparchitecture.shtml)

**J**SP pages are high level extension of servlet and it enable the developers to embed java code in html pages. JSP files are finally compiled into a servlet by the JSP engine. Compiled servlet is used by the engine to serve the requests.

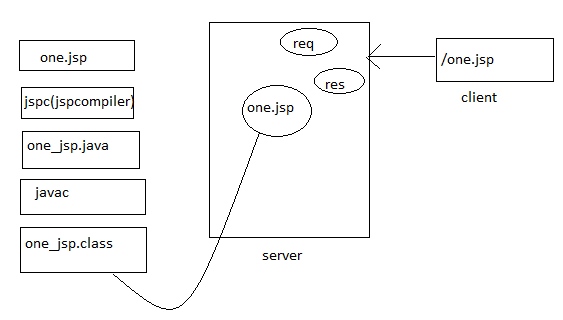
**javax.servlet.jsp**package defines two interfaces:

* *JSPPage*
* *HttpJspPage*

These interfaces defines the three methods for the compiled JSP page. These methods are:

* *jspInit()*
* *jspDestroy()*
* *\_jspService(HttpServletRequest request,HttpServletResponse response)*
* The **javax.servlet.jsp.JspPage** interface contains two methods:
* 1. **public void jspInit()** - This method is invoked when the JSP is initialized and the page authors are free to provide initialization of the JSP by implementing this method in their JSPs.
* 2**. public void jspDestroy() -** This method is invoked when the JSP is about to be destroyed by the container. Similar to above, page authors can provide their own implementation.
* The **javax.servlet.jsp.HttpJspPage** interface contains one method:
* public void \_jspService(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

* as part of every jsp container,we have a program called jsp compiler.
* It is the responsibility of jsp compiler to convert the jsp into corresponding servlet,when the client send the requests for the first time.
* The following is the procedure that executes,when the client send the request to a jsp for the first time.



The jsps which we developed ,when we deploy in a server,they are converted into corresponding servlet by the jsp compiler. These servlets are called as automatically generated servlets.

Manually developed servlets gives best performance when compared with the automatically generated servlets.

Initially sunmicrosyastems has released servlet API to develop webbased applications.to develop servlet ,we must be aware of java.without knowledge of java we cannot able to develop servlet applications.

* For testers and html developers it is very difficult to leran the java programming.
* Meanwhile Microsoft has released atechnology ASP(Active server pages) this is also used to develop webbased applications.
* To develop asp applications we no need to aware of any technology.if we know how to develop scripting languages like html,javascript,vb script etc.. ,we can develop ASP applications
* Here It is the responsibility of IIS server to convert the scripting language code intop corresponding server understandable language and execute the applications.
* As so many customers are moving toward ASP applications,sunmicrosystems has released a technology JSP(Java Server Pages).
* According to sunmicro systems, to develop jsps we can use any scripting language.

As we said whenever we place jsp in a project and deployed it, it is the responsibility of jsp compiler to convert .jsp into corresponding .java and produce a servlet.

Whenever jsp compiler converts .jsp code into corresponding servlet code,,it follows some thumb rules.

**JSP Elements**

Every jsp program is a set of jsp elements.

|  |
| --- |
| Set of jsp elements |

One.jsp

The following are the elements of jsp.

1. Template text
2. Scriptlet
3. Jsp expressions
4. Jsp declarations
5. Jsp directives
6. Jsp actiontags
7. Jsp customtags
8. ELExpressions(Expression language).

**1 ) Template text:**

If you would like to send any output to the client,we write the text in the form of template text.

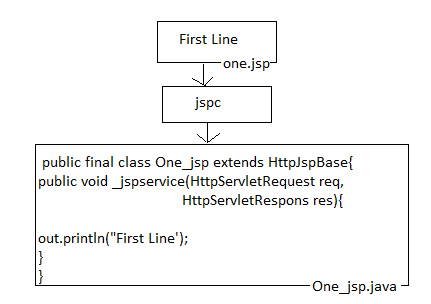
|  |
| --- |
| FirstLine  Secondline  Third line |

Two.jsp

All the above three lines of code is called as template text.

Whenever the jsp compiler encounted template text,it just place it inside the *out.print();*

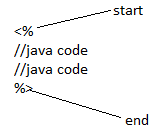
i.e out.write(“First line”).



**2 Scriptlet :**

We use scriptlets to write the java code in jsps.

Syntax:



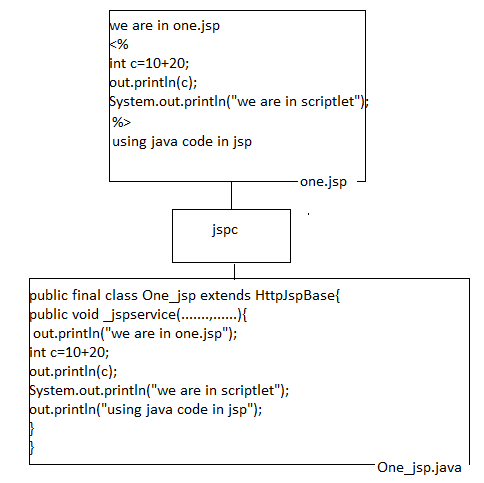
**Eg:**

|  |
| --- |
| we are in one.jsp  <%  int c=10+20;  out.println("c");  System.out.println("we are in scriptlet");  %>  using java code in jsp |

One.jsp

Whenever the jsp compiler encounters the scriptlet tag,it directly copies the code of scriptlet into \_jspservice().

Eg:



Whenever we declare local variables in scriptlet,by default ,these variables become local variables.

* when we provide a scriptlet code it is place inside \_jspservice()(we know if a variable is placed inside a method it becomes localvariable).
* In a jsp we can provide multiple scriptlets.

Eg:

|  |
| --- |
| <%  int a;  %>  <%  a=12;  %>  <%  System.out.println(a);  %>  One.jsp |

* While developing the scriptlets,if we doesn’t follow rules of java,jsp compiler converts .jsp program to .java program,but java compiler will fail to convert .java program into .class file.
* If java compiler fails all the error messages will be sent to the browser.

We can configure a jsp into web.xml file for security reasons. The following is the configuration of a jsp in web.xml file.

|  |
| --- |
| <web-app>  <servlet>  <servlet-name>aa</servlet-name>  <jsp-file>/one.jsp</jsp-file>  </servlet>  <servlet-mapping>  <servlet-name>aa</servlet-name>  <url-pattern>/o</url-pattern>  </servlet-mapping>  </web-app>  web.xml |

Now we can access the jsp through the url pattern specified in the web.xml file.

To improve the security of jsps we can also place the jsp file inside the WEB-INF folder.in this scenario we need to specify the location of jsp in web.xml file.

Eg:

<jsp-file>/WEB-INF/one.jsp</jsp-file>

**implicit objects or variables of jsp**

the variables which can be used in the jsp without declaration are called as implicit variables.

In jsp there are 9 implicit variables are available.

1. request
2. response
3. pageContext
4. session
5. config
6. application
7. out
8. page
9. exception

the following are the local variables and datatypes generated by jsp compiler.

|  |
| --- |
| public final class one\_jsp extends HttpJspBase  public void \_jspService(HttpServletRequest request, HttpServletResponse response)  throws java.io.IOException, ServletException {  PageContext pageContext ;  HttpSession session;  ServletContext application;  ServletConfig config;  JspWriter out;  Object page;  Exception exception;  - - - - - - - - - - - - - - -  - - - -- - - - - - - - - - - - - -  }  }  one\_jsp.java |

All these local variables are declared by jsp compiler as part of \_jspService(.,.) in the form of local variables.

We can directly use the implicit variables in scriptlet.

**request**

we can use this implicit variable directly in the jsp.



Eg:

|  |
| --- |
| <%  out.println(request.getMethod());  out.println(request.getRequestURI());  out.println(request.getHeader(“user-agent“));  out.println(request.getHeader(“accept-language”));  %> |

**response**

response implicit variable can be used directly in jsp.



By using response implicit variable ,we can send the contentType,as well as error message to the client.

Eg:

<%

response.setContentType(“text/html”);

%>

<%

response.sendError(555,“some problem with program”);

%>

<%

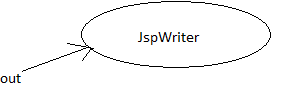
response.setContentType(“application/image”);

%>

**out**

out is a implicit variable which can be directly used in jsp, to send the output to the client.

In jsp out is pointing to JspWriter.



Eg:

<%

int i=20;

int j=30;

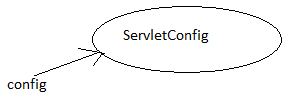
out.println(“I value”+i);

out.println(“j value”+j);

%>

**config**

config is a implicit variable,which we can directly use in jsp. We use config to remove hardcoding as well as to find the name of the servlet,which is configure in web.xml.



Eg:

<%

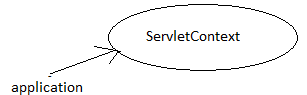
out.println(config.getInitParameter(“uname”));

out.println(config.getServletName());

%>

**application**

this implicit variable holds the object of ServletContext.



By using application,we can remove hardcoding.

And also we can find the majorVersion and minorVersions of ServletContainer.

ServletContainer:

Servletcontainer is an implementation of servlet API.

Eg:

<%

out.println(application.getMajorVersion()+”<br>”);

out.println(application.getMinorVersion());

%>

**page**

page is a implicit variable in jsp,that holds currently executing servlet object.

Eg:

<%

out.println(page.getClass());

%>

**exception**

this implicit variable will be created whenever the jsp is declared as errorpage.if you have not declared jsp as errorpage,it ll not create this exception.

Eg:

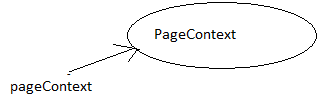
<%@ page isErrorpage=”true”%>

<%

out.println(“exception”);

%>

**pageContext**

****

whenever server starts executing \_jspService(),it creates the PageContext object.server removes the PageContext object,once if it has executed the \_jspService().

* PageContext is a class which is available in javax.servlet.jsp package.
* By using this implicit variable,we can get any of other implicit objects of jsp.

Eg:

We can use methods like

getRequest()

getResponse()

getSession()

getOut() etc……

**3 Jsp Declarations**

By using scriptlets,we cannot create instance variables,instance methods.

to create instance variables,instance methods,static variables,static methods we use jsp declarations.

Syntax:

<%!

//insatnace variables

//instance methods

//static variables

//static methods

%>

Eg:

<%!

int a;

public void methodOne(){

System.out.println(“output from methodOne()”);

}

Public void methodTwo(){

System.out.println(“output from methodTwo()”);

}

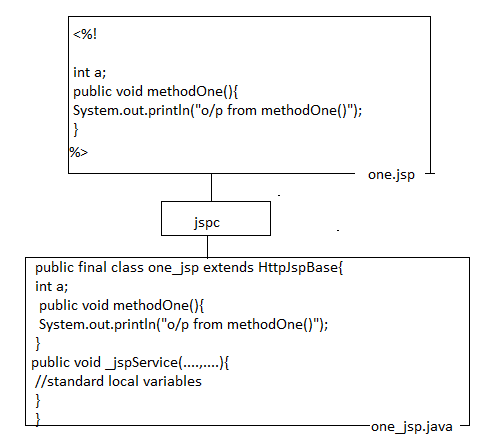
%>

<%

methodOne();

%>

Whenever we write jsp declarations,the code is directly copied into generated class.



In jsp declarations,we cannot use implicit variables, why because implicit variables are local to \_jspService().

**3 Jsp Expressions**

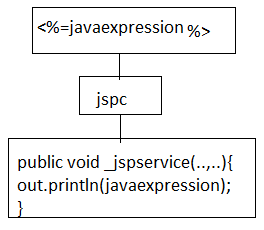
Syntax:

<%=java expression%>

Eg:

|  |
| --- |
| <%  int a=10;  %>  <%=a%> |

Whenever jsp compiler encounter jsp expression,it will convert into corresponding servlet as shown below.



Eg:

<%

int i=20;

int j=30;

String name="software";

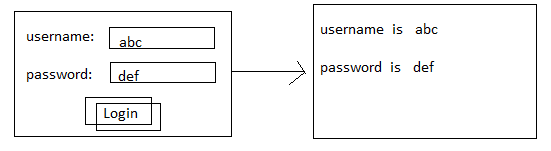
%>

<%=i+j%>

<%=i<j%>

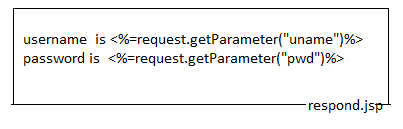
<%=name.equals("software")%>

Eg:



To develop the above application,first we need to take a form with above prototype

Now develop a jsp to for displaying output as



**3 Jsp Directives**

Jsp directive is an instruction given to the jsp compiler.jsp compiler uses this instruction to convert jsp file into corresponding servlet.

There are three directives available ,they are

* + - 1. page directive
      2. include directive
      3. taglib directive

syntax:



Eg:

<%@ page attribute %>

<%@ include attribute %>

<%@ taglib attribute %>

**page directive**

we can supply the following attributes as part of page directive.

* language
* import
* info
* contentType
* buffer
* autoFlush
* isErrorPage
* errorpage
* session
* isElIgnored
* extends

**language**

by default jsp compiler thinks that java code is provided as part of scriptlet.if we would like to use different scripting languages as part of scriptlet, we specifi by using **language** page attribute.

we can provide javascript also as part of jsp.when we provide javascript we have to give an instruction to jsp compiler saying that scriptlet contains javascript.

<%@page language="javascript"%>

<%

var a= 10;

alert("hi welcome");

%>

According to jsp specification released by sunmicrosystems,in scriptlet,we provide only scripting language.it is nthe responsibility of jsp compiler to translate scripting language into corresponding java code.

Today there is no server available in the market,who is supporting any scripting language.i.e the implementation softwares of jsp specification are supporting only java language in jsp.

**import**

import attribute of page directive is used to import the packages in generated servlet.

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

<%

ArrayList al=new ArrayList();

al.add(10);

al.add(20);

al.add(30);

%>

<%=al%>

* in the jsp page we can use page directive anywhere.
* To import the multiple packages ,we use import directive with a “,” separator as shown below.

<%@ page import=”java.util.\*,java.sql.\*”%>

* It is not recomonded to use a single import for multiple packages.it is recomonded to use individual imports for importing the multiple packages.

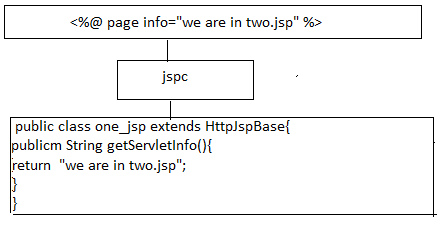
**info**

Eg:

<%@ page info=”we are in two.jsp”%>

In jsps we use info attribute to specify the purpose of jsp. Whenever jsp compiler has encounted info page directive it will convert that into corresponding servlet getServletInfo() in the generated servlet.

Eg:



**contentType**

by default jsp compiler place the contentType as

response.setContentType("text/html");

if we want to specify a different contenttype we use contentType page attribute.

<%@ page contentType="text/xml"%>

<student>

<no>12</no>

<name>ramesh</name>

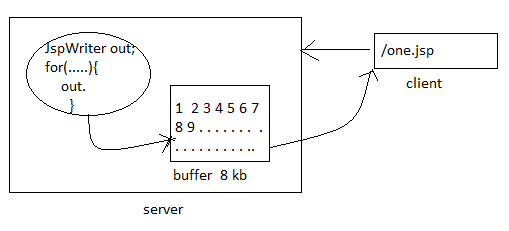
</student>

**buffer**

in the jsp,whenever out implicit variable is created,it will create an internal buffer of 8kb.whenever jsp would like to send the o/p to the client,instead of sending it directlyit ll write the content to the buffer.

It is the responsibility of server to send the buffer contents to the clients in the following scenarios

* Whenever the buffer got filled,the output will be sent to the client.
* When the execution of jsp is finished ,it ll send the output to the client.



If we would like to change the size of the buffer,we specify the buffer page directive as

|  |
| --- |
| <%@ page buffer="10kb"%>  <%  for(int i=0;i<10000;i++){  out.println("hi"+i);  out.println("\n");  }  %>  One.jsp |

When we execute the above jsp,it ll create out variable with a size of 10kb.if we would like to controle the autoFlush value we can specify a directive autoFlush .

**autoFlush**

the following is an example of autoFlush page directive.

|  |
| --- |
| <%@ page buffer="10kb"%>  <%@ page autoFlush="false"%>  <%  for(int i=0;i<10000;i++){  out.println("hi"+i);  }  %> |

When we specify the autoFlush="false",the server will not flush the content to the client,until the jsp execution is finished.

The problem with autoFlush="false" is,when we’d like to send more data to client and if the buffer is filled,execution of jsp failed an display an error message

*java.io.IOException: Error: JSP Buffer overflow*

becasuse of this reason,it is not recomonded to controle the buffer autoFlush.in ,most of the cases we ll never use buffer or autoFlush page directives,we use only default setting of jsp

8kb of buffer size and autoFlush="true".

**Errorpages:**

Jsp error pages are used to display appropriate error messages to the client.generally when the exception is occurred server sends error messages with http status codes.these error messages will not be understandable by the clients.we use error pages to display appropriate error messages.

Procedure to erropages:

Step1:

create an error page with appropriate error messages.for this jsp,we have to specify this page should act as errorpage,so we use isErrorPage page attribute.

|  |
| --- |
| <%@ page isErrorPage="true"%>  some problem is occured  <br>  <%=exception%> |

error.jsp

step2:

to use the above errorpage inour program,we must specify a directive errorPage in our jsp.

|  |
| --- |
| <%@ page errorPage="/error.jsp"%>  <%  out.println(10/0);  %> |

one.jsp

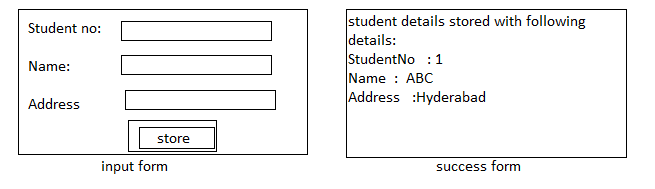
**session tracking techniques**

generally there are two types of formbased applications.

1.Single formbased applications

2. multiple formbased applications

If we ‘d like to develop a single formbased application by using a single form.this kind of applications are called as single formbased applications.



To implement above requirement,we have to develop two programs. They are

* Input form
* Store.jsp

Input form:

|  |
| --- |
| <form action="/myproject/store.jsp">  SNO:<input type="text" name="sno"/><br>  Name:<input type="text" name="sname"/><br>  Address:<input type="text" name="address"/><br>  <input type="submit" value="store"/>  </form> |

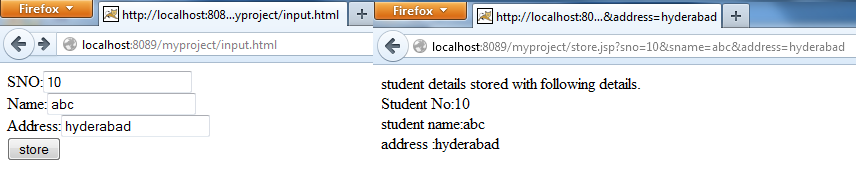
input.jsp

Store.jsp:

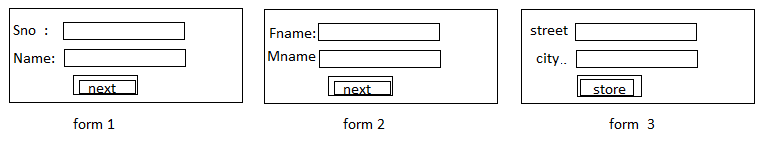
|  |
| --- |
| <%  String sno=request.getParameter("sno");  String sname=request.getParameter("sname");  String address=request.getParameter("address");  //code to store the details in database.  %>  student details stored with following details.<br>  Student No:<%=sno%><br>  student name:<%=sname%><br>  address :<%=address%><br> |

store.jsp

when we deploy above application the output will be.



We follow above procedure to develop single formbased applications. To implement multiple formbased applications. We take multiple input forms.as



When we design the forms like above and deployed the project,the application is not works as expected.our application is able to capture the data from the last form only,it ll not able to capture the data from other forms.

* Any server which is built based on http protocol becomes stateless server.all the JEE servers are stateless servers only.
* To resolve the problem of multiple formbased applications or to remember the conversation between client and server sunmicro systems has suggested 4 solutions. They are called as session tracking techniques.
* They are

hidden variables

cookies

sessions

sessions with URL rewriting

**Hidden variables**

By using this technique ,when the client send the data to the server.server get the data and sendback the data to client in the form of hidden form fields.whenever the client click on next button it is the responsibility of client to send all the hidden form fields data also to server.

In html we use <input> tag with type="hidden",if you don’t want to display the fields to the user.but this field will be stored in the browser only. When we click on next,client will send all the hidden form fields data to server.

Eg:

<input type="hidden" name="sno" value="10"/>

Consider th above requirement to develop multiple form based application using hidden formfields.

Step1: develop form1 .html

|  |
| --- |
| <form action="/myproject/form2.jsp">  Sno:<input type="text" name="sno"/><br>  Name:<input type="text" name="name"/><br>  <input type="submit" value="Next"/>  </form> |

Step2:

Develop form2.jsp ,this jsp is responsible to capture the data from the user and send the data back to the client in the form of hidden variables.

|  |
| --- |
| <%  String sno=request.getParameter("sno");  String name=request.getParameter("name");  %>  <form action="/myproject/form3.jsp">  Fname<input type="text" name="fname"/><br>  Mname<input type="text" name="mname"/><br>  <input type="hidden" name="sno" value="<%=sno%>"/>  <input type="hidden" name="name" value="<%=name%>"/>  <input type="submit" value="Next"/>  </form> |

Step3:

Develop form3.jsp to capture the data from form2.jsp and send it back to the client with two visible fields street and city

|  |
| --- |
| <%  String sno=request.getParameter("sno");  String name=request.getParameter("name");  String fname=request.getParameter("fname");  String mname=request.getParameter("mname");  %>  <form action="/myproject/store.jsp">  Street<input type="text" name="street"/><br>  City<input type="text" name="city"/><br>  <input type="hidden" name="sno" value="<%=sno%>"/>  <input type="hidden" name="name" value="<%=name%>"/>  <input type="hidden" name="fname" value="<%=fname%>"/>  <input type="hidden" name="mname" value="<%=mname%>"/>  <input type="submit" value="Store"/>  </form> |

Step4:

Develop store.jsp to capture the data from hidden formfields as well as visible form fields .store the data into DB and send the acknowledgment back to client.

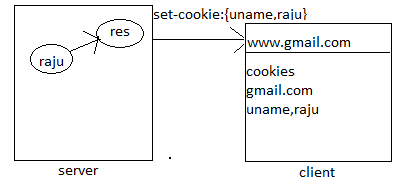
|  |
| --- |
| <%  String sno=request.getParameter("sno");  String name=request.getParameter("name");  String fname=request.getParameter("fname");  String mname=request.getParameter("mname");  String street=request.getParameter("street");  String city=request.getParameter("city");  // code to store all this data into DB server  %>  the following details received  Sno :<%=sno%><br>  Name:<%=name%><br>  fname:<%=fname%><br>  mname:<%=mname%><br>  Street:<%=street%><br>  city:<%=city%><br> |

* the disadvantage of hidden form fields is ,we have to transfer huge amount of data between client and server for multiple times.

**Cookies:**

Cookie: cookie is a small piece of information sent by theserver to client.

* Cookies get stored in the browsers memory.
* it is the responsibility of browser/client to send all the cookies back to the server while he is sending request.
* If a server would like to send a cookie to the client server has to create the cookie object & add it to response object.once if you add cookie object to response object,server converts cookie in the form of header and send it to client.
* Server send the cookies to the client usinga header “set-cookie”.



* When we send the cookies to the client,browser stores cookies in the form of domain names.
* When the client send the request to server it is the responsibility of browser to send all the cookies to server which are associated with that domain.client converts all the cookies in the form of a header and send it to server.
* A client can store cookies form different websites.its the responsibility of client to send all the cookies to server which are associated with appropriate domain.

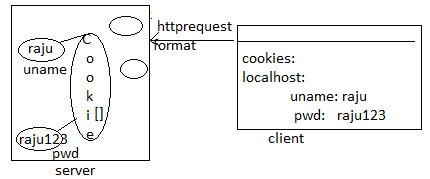
Let us look at a simple servlet,which creates Cookie object and send it to client.

|  |
| --- |
| import javax.servlet.http.\*;  import java.io.\*;  public class SendCookieServlet extends HttpServlet{  public void service(HttpServletRequest request,HttpServletResponse response)throws IOException{  Cookie c1=new Cookie("uname","raju");  Cookie c2=new Cookie("pwd","raju123");  response.addCookie(c1);  response.addCookie(c2);  PrintWriter out=response.getWriter();  out.println("cookie added successfully");  }  } |

SendCookieServlet.java

In the above servlet,we have created two cookie objects and sent it to client,we have added cookie objects to response object by using a method addCookie();

* When we try to send cookies to the client using jsp program,jsp is sending an extra cookie to client whose name is “JSESSION ID”.
* Whenever the client send the request to server,server creates two objects request and response .now server opens HttpRequest format and checks how many cookies sent by client to server.based on number of cookies server creates cookie array. The size of the cookie array is depends upon the no of cookies sent by the client. Now the server add cookie array to request object.

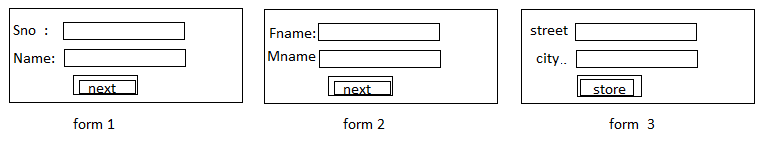


Let us develop a servlet program,which reads the cookies sent by the client.

|  |
| --- |
| import javax.servlet.http.\*;  import java.io.\*;  public class ReceiveCookieServlet extends HttpServlet{  public void service(HttpServletRequest request,HttpServletResponse response)throws IOException{  PrintWriter out=response.getWriter();  Cookie c[]=request.getCookies();  for(int i=0;i<c.length;i++){  if(c[i].getName().equals("uname")){  out.println(c[i].getName()+" "+c[i].getValue());  }  if(c[i].getName().equals("pwd")){  out.println(c[i].getName()+" "+c[i].getValue());  }  }  out.println("cookies readed successfully");  }  } |

ReceiveCookieServlet.java

Consider the multiple form based application which is give below



Let us implement this by using cookies.here we need to get the data from form1 and add it to cookie then send to client ,then client send back the data and cookies stored in its memory.

Step1:develop form1.html

|  |
| --- |
| <form action="/myproject/form2.jsp">  Sno:<input type="text" name="sno"/><br>  Name:<input type="text" name="name"/><br>  <input type="submit" value="Next"/>  </form> |

form1.html

Step2:

Develop form2.jsp . it is responsible to get the data from the form1.html and store the data into cookies and add cookies to response to send it to client.as well as html code to display second form client.

|  |
| --- |
| <%  String sno=request.getParameter("sno");  String name=request.getParameter("name");  Cookie c1=new Cookie("sno",sno);  Cookie c2=new Cookie("name",name);  response.addCookie(c1);  response.addCookie(C2);  %>  <form action="/sls/form3.jsp">  Fname<input type="text" name="fname"/><br>  Mname<input type="text" name="mname"/><br>  <input type="submit" value="Next"/>  </form> |

form2.jsp

step3:

develop form3.jsp to capture cookies as well as form data sent by the client. And again add all the data to client along with form controls.

|  |
| --- |
| <%  String sno=null;  String name=null;  String fname=request.getParameter("fname");  String mname=request.getParameter("mname");  Cookie c[]=request.getCookies();  if(c!=null){  for(int i=0;i<c.length;i++){  if(c[i].getName().equals("sno")){  sno=c[i].getValue();  }  if(c[i].getName().equals("name")){  name=c[i].getValue();  }  }  response.addCookie("sno",sno);  response.addCookie("name",name);  response.addCookie("fname",fname);  response.addCookie("mname",mname);  }  else{  out.println("there are no cookies");  }  %>  <form action="/sls/store.jsp">  Street<input type="text" name="street"/><br>  City<input type="text" name="city"/><br>  <input type="submit" value="Store"/>  </form> |

form3.jsp

step4: develop store.jsp to read all the data sent by the client through form3.jsp.and to display the same data.

|  |
| --- |
| <%  String sno=null;  String name=null;  String fname=null;  String mname=null;  String street=request.getParameter("street");  String city=request.getParameter("city");  Cookie c[]=request.getCookies();  if(c!=null){  for(int i=0;i<c.length;i++){  if(c[i].getName().equals("sno")){  sno=c[i].getValue();  }  for(int i=0;i<c.length;i++){  if(c[i].getName().equals("name")){  name=c[i].getValue();  }  for(int i=0;i<c.length;i++){  if(c[i].getName().equals("fname")){  fname=c[i].getValue();  }  for(int i=0;i<c.length;i++){  if(c[i].getName().equals("mname")){  mname=c[i].getValue();  }  }  else{  out.println("there are no cookies");  }  %>  the following details are received:<br>  Sno :<%=sno%><br>  Name :<%=name%><br>  Fname :<%=fname%><br>  Mname :<%=mname%><br>  street:<%=street%><br>  city :<%=city%><br> |

Disadvantages of cookies:

* The major disadvantage of above application is if the browser/client doesn’t accept the cookies ,the application fails.
* Here also the data is getting transferred between the client and server for multiple times.
* As cookies stored in browsers memory anybody can see the data of the cookie which is stored in browsers memory.this lead to majoe security concerns.
* It is not recomonded to use cookies in projects which uses/store sensitive data.
* We can use cookies for personalizing websites.it is recomonded to use cookies if the data in insensitive.
* There are two types of cookies are available.

Persistent cookies

Non persistent cookies

**Persistent cookies:**

A cookie which will be get stored in a browsers memory even if you close the browser.

**Non –persistent cookies:**

The cookies which will be removed from the browser when we close the browser.

By default we get only non persistent cookies. Non persistent cookies will be removed when we close the browser ,they stored in browsers memory where as persistent cookies get stored in file system of the operating system.the pesrsitsent cookies have the expiry date .browser removes this cookies based on the expiry time. Following is an example of non-persistent cookie.

<%

Cookie c1=new Cookie(“uname”,”raju”);

c1.setMaxAge(100000);

response.addCookie(c1);

%>

**Sessions**

* It is the responsibility of server to create session object on behalf of client.
* The server creates sesion object,when client sends the request fro the first time only.
* Whenever server has created the session object server executes an unique algorithm to generate the SESSION ID cookie and send it to the client.
* Now it is the responsibility of client to send the JSESSIONID cookie to the server,when the client send the request for the second time.
* Session object contains two methods like *getId()* and *isNew()*
* The getId() returns the unique id which is associated with session object.
* isNew() checks ,whether the session object is created newly or is it using old session object only.if it is using new session object,it returns true.otherwise it returns false.
* To create session object ,we use an interface HttpSession which is available in javax.servlet.http package.
* We use a method getSession() to get the session object created by the server.

Syntax:

HttpSession session=request.getSession(boolean value)

* In servlets,to get the session object,we use method like

request.getSession(true) and request.getSession(false)

* Let us write a simple program on sessions deals with its methods

|  |
| --- |
| import javax.servlet.http.\*;  import java.io.\*;  public class SessionServlet extends HttpServlet{  public void service(HttpServletRequest request,HttpServletResponse response) throws IOException{  PrintWriter out=response.getWriter();  HttpSession session=request.getSession(true);  out.println("session object is"+session);  out.println("session id is "+session.getId());  out.println("old or new session "+session.isNew());  }  }  o/p:  session object isorg.apache.catalina.session.StandardSessionFacade@104c575  session id is A89FEB2FF86F63199E12D96B2768AA7A  old or new session true |

SessionServlet.java

Difference between request.getSession(true) and request.getSession(false):

* If request.getSession(true) is executed,if the session object is not available on behalf of client.server creates new session object ,if the session object is already available,server returns available session object
* If request.getSession(false) is executed,if session object is not available on behalf of client,server will not create new session object.it returns a null value.if the session object is already available it returns existed session object.
* In jsp session is implicit variable.here in jsp jspcompiler creates session object.jspcompiler always create the session object by using request.getSession(true).
* In jsps,if we don’t want the jsp compiler to create the session object,we can use session page directive.

Eg:

<%@ page session="false">

* When the above line of code is executed jsp compiler will not create session implicit variable itself. If jsp compiler is not creating it, we can provide the code to create the session object according to outr convenient.

<%@ page session="false"%>

<%

HttpSession session=request.getSession(false);

out.println(session);

%>

* We can use session object to storwe the data of a specific client/browser.
* We use the following methods to store and retrieve the data from session object.

void setAttribute(key,object);

Object getAttribute(key);

void removeAttribute();

* Observe the following program which demonstrates the use of session object.

|  |
| --- |
| <%@ page import="java.util.\*"%>  <%  ArrayList al=new ArrayList();  al.add("one");  al.add("two");  session.setAttribute("list",al);  Object o=session.getAttribute("list");  ArrayList l=(ArrayList)o;  out.println(l);  %> |

* We can implement any multiple formbased application based on sessions.
* Procedure to implement multiple formbased application by using sessions

Step1:

Develop form1.html

|  |
| --- |
| <form action="/myproject/form2.jsp">  Sno:<input type="text" name="sno"/><br>  Name:<input type="text" name="name"/><br>  <input type="submit" value="Next"/>  </form> |

form1.html

step2:

develop form2.jsp in this we ‘d like to store the data into session object,which we received from form1.html and send another form to client.

|  |
| --- |
| <%  String sno=request.getParameter("sno");  String name=request.getParameter("name");  session.setAttribute("sno",sno);  session.setAttribute("name",name);  %>  <form action="/myproject/form3.jsp">  Fname<input type="text" name="fname"/><br>  Mname<input type="text" name="mname"/><br>  <input type="submit" value="Next"/>  </form> |

form2.jsp

step3:

develop form3.jsp to capture data from form 2.jsp and store that data in session object.and send another form to client.

|  |
| --- |
| <%  String sno=request.getParameter("sno");  String name=request.getParameter("name");  session.setAttribute("sno",sno);  session.setAttribute("name",name);  %>  <form action="/myproject/form3.jsp">  Fname<input type="text" name="fname"/><br>  Mname<input type="text" name="mname"/><br>  <input type="submit" value="Next"/>  </form> |

form3.jsp

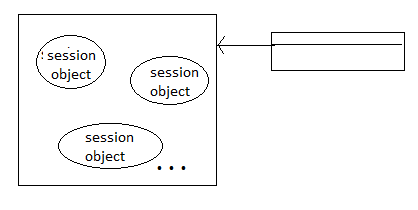
step4:

develop store.jsp to capture all the data from session object and also data from form3.jsp.

|  |
| --- |
| *<%*  String street=request.getParameter("street");  String city=request.getParameter("city");  String sno=(String)session.getAttribute("sno");  String name=(String)session.getAttribute("name");  String fname=(String)session.getAttribute("fname");  String mname=(String)session.getAttribute("mname");  %>  the following details are received:<br>  Sno :<%=sno%><br>  Name :<%=name%><br>  Fname :<%=fname%><br>  Mname :<%=mname%><br>  street:<%=street%><br>  city :<%=city%><br> |

store.jsp

* The drawback of above application is if the client doesnt accept the cookies, the application fails.
* Here when a client send the request to server ,server creates the session object on behalf of the cleint.
* When the client close the browser and open it again and send request server creates another session object. But server hasn’t removed old sesion object.
* The problem with this approach is so many unused objects are available in server.
* Because of this problem in server ,server kep on occupying lot of space .it causes memory management issues in the server.
* It is always recomonded to remove unused object from server.



**Scenario1**:

* It is the responsibility of server to find unused session object as part of server& remove all those session objects.
* Server uses amethod getLastAccessedTime() to find when the session object accessed at last. Most of the servers will remove/invalidate the session objects after every 30 minuites.

**Scenario 2:**

* To invalidate the session object,we call a method session.invalidate()

<%

session.invalidate();

%>

* When the above code is executed server removes currently associated session object.

**Scenario 3:**

* If we don’t want server to remove session object after every 30 minutes,we can configure the session timeout specific to our project.we have to configure it in web.xml.
* The folowing is the configuration.

|  |
| --- |
| <web-app>  <session-config>  <session-timeout>1</session-timeout>  </session-config>  </web-app> |

web.xml

**Scenario 4:**

* In the jsp we can specify the session timeout by using

setMaxInactiveInterval(int seconds);

eg:

<%

session.setMaxInactiveInterval(120)

%>

* In every project,we try to create the session object only once ,when we are trying to enter into project by providing valid nusername and password.
* In all the remaining pages we get the session object which is already available.if the session object is not available it display error message.

**Sessions with URL rewriting:**

We use this technique,if the client doesn’t accept cookies.here we need to use encodeURL().

In the jsp wherever the URL’s are available all of them has to be placed inside response.encodeURL().

Eg:

<form action="<%=response.encodeURL("form2.jsp")%>">

</form>

* When the above line is executed it checks whether,the client is accepting cookies or not.if the client is not accepting cookies it ll append the JSESSIONID cookie to the URL as shown below

<http://localhost:9000/myproject/form3.jsp;jsessionid=8110D6FDDBCCBE42E1B2BC9EFD5E8F17>

**RequestDispatchers**

RequestDispatcher is an object which can be used to dispatch the request from one resource to another resource. By using RequestDispatcher you can dispatch the request to any resource(Servlet,jsp,html files).

RequestDispatcher object contains two methods they are

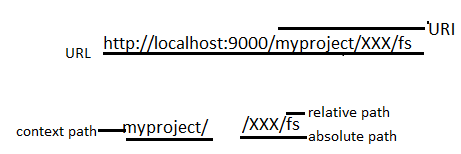
* Forward
* Include

Syntax:

void forward(ServletRequest request,ServletResponse response)

void include(ServletRequest request,ServletResponse response)

here we need to understand different terms like URL,URI,contextpath,relativepath,absolute path. Look at below URL



If we want to dispatch the request to manother resource ,we must get the RequestDispatcher object.there are two ways available to get RequestDispatcher object. They are

1*.by using ServletContext object:*

RequestDispatcher rd=application.getRequestDispatcher(String path);

2*By using HttpServletRequest object :*

RequestDispatcher rd=request.getRequestDispatcher(String path);

Observe the below example explaining RequestDispatcher

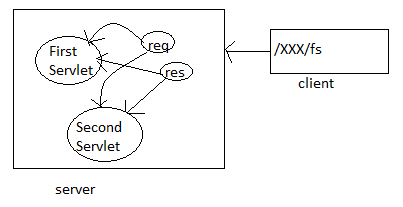
|  |
| --- |
| import javax.servlet.http.\*;  import javax.servlet.\*;  import java.io.\*;  public class FirstServlet extends HttpServlet{  public void service(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException{  response.setContentType("text/html");  PrintWriter out=response.getWriter();  out.println("o/p from first servlet");  ServletContext application=getServletContext();  RequestDispatcher rd=application.getRequestDispatcher("/XXX/ss");  rd.include(request,response);  }  } |

FirstServlet.java

o/p: output from firstservlet output from second servlet

When the client send the request to server to execute FirstServlet Server creates request and response objects. Now the server will handover the request to FirstServlet service().whenever the server has encountered the RequestDispatcher objectit gets the path of the resource and call the appropriate method include() or forward().

* When we call the include(), it is the responsibility of server to dispatch the request to the appropriate resource ,while dispatching the request,server uses same request and responses to other resource also (whether servelt or jsp).



* Incase of include() server will include the output of callingservlet and called servlet and send the output to client.
* ServletChaining:

In a single if more than one servlet is got executed we call it as servlet chaining.

**forward():**

incase of forward the RequestDispatcher will forward ther request to called servlet. Here server will discard the output of calling servlet and display only the output of called servlet(SecondServlet). When we use include it will include the output of multiple servlets and send it to client.

Eg:

|  |
| --- |
| Import javax.servlet.http.\*;  import javax.servlet.\*;  import java.io.\*;  public class FirstServlet extends HttpServlet{  public void service(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException{  response.setContentType("text/html");  PrintWriter out=response.getWriter();  out.println("o/p from first servlet");  ServletContext application=getServletContext();  RequestDispatcher rd=application.getRequestDispatcher("/XXX/ss");  rd.forward(request,response);  }  } |

FirstServlet.java

o/p: output from second servlet.

In jsps we write RequestDispatcher code as part of scriptlet only.

|  |
| --- |
| o/p from one.jsp  <%  RequestDispatcher rd=application.getRequestDispatcher("two.jsp")  rd.include(request,response);  %> |

one.jsp

output:: o/p from one.jsp

o/p from two.jsp

**Scopped variables**

In jsp we have 4 scoped variables. They are

* page scope
* request scope
* session scope
* application scope

**page scope**

storing data into pageContext object is known as page scope.

Eg:

|  |
| --- |
| <%  String uname="abc";  String pwd="xyz";  pageContext.setAttribute("uname",uname);  pageContext.setAttribute("pwd",pwd);  String a=(String)pageContext.getAttribute("uname");  String b=(String)pageContext.getAttribute("pwd");  %>  <%=a%><br>  <%=b%><br>  <%  pageContext.removeAttribute("pwd");  b=(String)pageContext.getAttribute("pwd");  %>  <%=b%> |

display.jsp

**request scope**

storing data into request object is known as request scope.

Eg:

|  |
| --- |
| <%  String uname="aaa";  request.setAttribute("uname",uname);  %>  <%=request.getAttribute("uname")%> |

display.jsp

**application scope**

storing data into ServletContext object is known as application scope.we use methods like setAttribute(),getAttribute(),removeAttribute() to perform operations on scoped variables.

|  |
| --- |
| <%@ page import="java.util.ArrayList"%>  <%  ArrayList al=new ArrayList();  al.add("one");  al.add("two");  application.setAttribute("list",al);  %>  <%=application.getAttribute("list")%> |

display.jsp

**session scope**

similar to above scopes we can store data into session scope and we can getback data from session scope.

|  |
| --- |
| <%  session.setAttribute("uname","raju");  %>  <%=session.getAttribute("uname")%> |

session scope

*when to use page scope:* if we want to store data until jsp execution is completed,we use page scope.

*When to use request scope*: if we want to store the data until the request is completely processed,we use request scope.

*When to use session scope:*if we want to share the data between multiple requests of the client we use session scope.

*When to use application scope*:the data which is not specific to any client or request we store that data in application scope.if we store the data in application scope ,all the clients can access it.

According to Sunmicrosystems jsp specification,it is not recomonded to eliminate java code form jsps.to eliminate java code form jsps we should not write scriptlet code as part of jsp.

Toremove the java code from jsps,sunmicrosystems has released jsp Action tags.

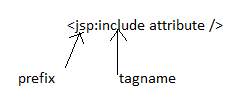
If we make use of jsp action tags in jsp,we can remove the java code from jsps. Following are the jsp action tags.

**Jsp action tags :**

* <jsp:include>
* <jsp:forward>
* <jsp:useBean>
* <jsp:setProperty>
* <jsp:getProperty>

How to use jsp action tags in jsp:

Syntax:



* <jsp:include>

include tag is used to include the output of multiple jsp pages and send to client

eg:

|  |
| --- |
| o/p from one.jsp  <jsp:include page="two.jsp"></jsp:include> |

one.jsp

|  |
| --- |
| o/p from two.jsp |

two.jsp

o/p: o/p from one.jsp o/p from two.jsp

when we use above code as part of one.jsp,it is the responsibility of jsp compiler to evaluate jsp action tag(i.e converting tag data into corresponding java code). Jsp include is called as dynamic include this is because server is creating two servlet objects for two jsps,when the client send the request to server.

* <jsp:forward>

|  |
| --- |
| o/p from one.jsp  <jsp:forward page="two.jsp"></jsp:forward> |

one.jsp

|  |
| --- |
| o/p from two.jsp |

two.jsp

o/p: o/p from two.jsp

when we execute the above one.jsp.server executes both one.jsp and two.jsp and it will discard the output of one.jsp and send only the output of two.jsp to the client.

<jsp:include> and <jsp:forward> are the alternatives to the RequestDispatcher s java code.

**include directive**

to include the output of multiple jsps we can use include directive also.

Eg:

|  |
| --- |
| output from one.jsp  <%@ include file="/two.jsp"%> |

one.jsp

|  |
| --- |
| Output from two.jsp |

two.jsp

here when client send the request to client it has displayed the output of one.jsp as well as two.jsp also .here include directive includes the output of one.jsp and two.jsp and send it to client.we call include directive as static include, why because here jspc include the o/p of these two jsps and form single .java file and give it to server.so the o/p id coming from single file.

**response.sendRedirect():**

|  |
| --- |
| output from one.jsp  <%  response.sendRedirect("two.jsp")  %> |

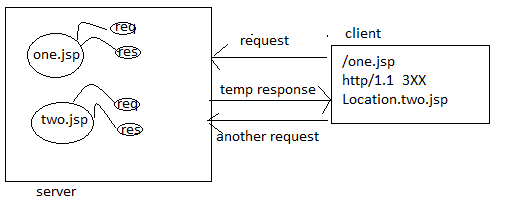
one.jsp

|  |
| --- |
| Output from two.jsp |

two.jsp

incase of response.sendRedirect() when the client send the request to one.jsp, server creates request object and response object and server handover the request to one.jsp.

when the server has encounted response.sendRedirect() server stops the execution of one.jsp and send a temporey response to client.as part of temporary response server will add 3XX status code and location header and send it to client.when the browser receives status code browser send another request to server to execute two.jsp.



Diff between forward and response.sendRedirect() :

* Forward executes multiple resources in single request.
* sendRedirect() executes multiple resources in multiple requests.
* By using response.sendRedirect() we can forward the request to the resources available in same server as well as resources available in other projects of same server as wella s resources available in some other servers.

Eg:

<%

response.sendRedirect("http://localhost:7001/project2/two.jsp")

%>

**Java Beans**

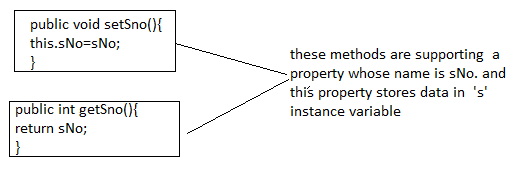
Java beans are very simple java programs.java bean specification is given by sunmicrosystems.

Any java progra if developed based on java bean specification,we call that java program as JavaBean. As part of JavaBean specification sunmicrosystems has released set of rules to develop java beans.if our java program follows all the rules of javabean specification,then only we call that as a javabean.

According to sun java bean specification,

* The java bean program must contain no argument constructor.
* Ever java bean must be placed inside apackage.
* A java bean can contains its elements as
  + Properties
  + Events
  + Additional methods
* Property in javabean :

A setter method or getter method is treated as property in javabean.



Sample java bean:

|  |
| --- |
| package info.sls.fls;  public class Student{  private int sNo;  private String sName;  private boolean gender;  public void setSno(int sNo){  this.sNo=sNo;  }  public int getSno(){  return sNo;  }  public void setSname(String sName){  this.sName=sName;  }  public String getSname(){  return sName;  }  public void setGender(boolean gender){  this.gender=gender;  }  public boolean getGender(){  return gender;  }  } |

Student.java

According to sun jsp specification a property can have a setter method or gettermethod or both.

* A java bean can contain set of events also.

Eg:

public void addXXXListener(XXXListener)

public void removeXXXListener(XXXListener)

public void addActionListener(ActionEvent ae)

public void removeActionListener(ActionEvent ae)

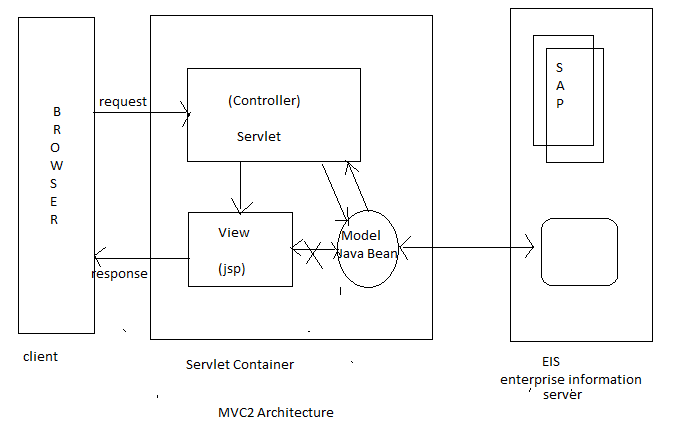
* A java bean can contain additional methods also.an additional method is not a property or not an event. It is just a method intended to provide some logics etc..

**Design patterns**

* Design patterns are the best solutions for repeatedly commonly occuring problems across multiple projects.
* Small talk company has invented a design pattern MVC to develop projects
* If we ‘d like to develop any project based on MVC design pattern, the project must contain minimum of three components, they are model,view,controller.
* Sunmicrosystems has identified 2 design patterns to resolve the problems of servlets and jsps they are

MVC model 1 design pattern

MVC model 2 design pattern

Generally architects are the one ,who gives architecture of the project.the architecture tells us how one component is communicating with another component.architect will take care of any modifications or changes in future even. If we construct the project based on architecture ,we can do any changes easily in future.

According to MVC architecture given by sunmicrosystems every project must contain minimum of three components.they are

Model

Controller

View

According to MVC2 architecture controller must be servlet.in struts ActionServlet acts as controller. In spring DispatcherServlet is acts as Controller.in MVC architecture model component must be java beans or EJB’s.these model components can interact with the EIS.

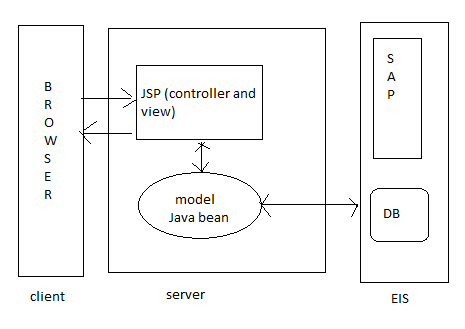
View component must be jsp in MVC2 architecture.

MVC1 architecture:

According to MVC1 architecture any of two components can be clubbed into one component.in MVC1 architecture controller and view component clubbed into a single component.

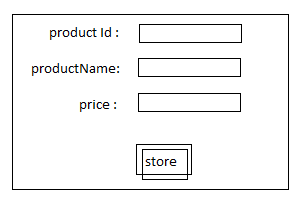
In MVC1 architecturejsp is acts as model and view.java beans are acts as model components in MVC1 .

For any project it is recomonded to use MVC2 architecture.



Let us develop one application based on MVC 1& MVC 2 architectures. Develop following application by separating out business logic ,presentation logic, controller logic.

**MVC 2:**



step1: develop product.html

|  |
| --- |
| <form>  productId:<input type="text" name="pid"/><br>  productName:<input type="text" name="pname"/><br>  price:<input type="text" name="price"/><br>  <input type="submit" value="Login"/>  </form> |

product.html

step2: develop a javabean to get the data from the controller and store the data in dbserver.

|  |
| --- |
| public class StoreJB{  int productId;  String productName;  double price;  public void setProductId(int productId){  this.productId=productId;  }  public int getProductId(){  return productId;  }  public void setProductName(String productName){  this.productName=productName;  }  public String getProductName(){  return productName;  }  public void setPrice(double price){  this.price=price;  }  public double getPrice(){  return price;  }  public void storeData(){  try{  System.out.println("data stored in db");  //code to store data into DB server  }  catch(Exception e){  }  }  } |

StoreJB.java

Step3: develpp a servlet which acts as a controller

|  |
| --- |
| import javax.servlet.http.\*;  import javax.servlet.\*;  import java.io.\*;  public class StoreDataServlet extends HttpServlet{  public void service(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException{  int productId=Integer.parseInt(request.getParameter("pid"));  String productName=request.getParameter("pname");  double price=Double.parseDouble(request.getParameter("price"));  StoreJB s=new StoreJB();  s.setProductId(productId);  s.setProductName(productName);  s.setPrice(price);  s.storeData();  request.setAttribute("pid",productId);  request.setAttribute("pname",productName);  request.setAttribute("price",price);  RequestDispatcher rd=request.getRequestDispatcher("display.jsp");  rd.forward(request,response);  }  } |

StoreDataServlet.java

Step4: develop a jsp which display the stored data infmn to the client

|  |
| --- |
| the following details stored in db server:<br>  <%=request.getAttribute("pid")%><br>  <%=request.getAttribute("pname")%><br>  <%=request.getAttribute("price")%> |

display.jsp

**MVC1:**

Develop above product form based based on MVC 1 architecture.

Step1: develop product.html

|  |
| --- |
| <form>  productId:<input type="text" name="pid"/><br>  productName:<input type="text" name="pname"/><br>  price:<input type="text" name="price"/><br>  <input type="submit" value="Login"/>  </form> |

product.html

|  |
| --- |
| step2: develop a java bean which capture the data from controller and stores the data into DB server.  public class StoreJB{  int productId;  String productName;  double price;  public void setProductId(int productId){  this.productId=productId;  }  public int getProductId(){  return productId;  }  public void setProductName(String productName){  this.productName=productName;  }  public String getProductName(){  return productName;  }  public void setPrice(double price){  this.price=price;  }  public double getPrice(){  return price;  }  public void storeData(){  try{  System.out.println("data stored in db");  //code to store data into DB server  }  catch(Exception e){  }  }  } |

StoreJB.java

Step3:develop store.jsp as a controller.

|  |
| --- |
| <%@ page import="info.fls.sls.\*"%>  <%  int productId=Integer.parseInt(request.getParameter("pid"));  String productName=request.getParameter("pname");  double price=Double.parseDouble(request.getParameter("price"));  StoreJB s=new StoreJB();  s.setProductId(productId);  s.setProductName(productName);  s.setPrice(price);  s.storeData();  %>  the following details stored in db server:<br>  <%=productId%>  <%=productName%>  <%=price%> |

store.jsp

* In the above store.jsp,we are writing we are writing huge amount of java code.according to sunmicrosystems it is not recomonded to write java code as part of jsps. Here we should completely eliminate the java code from jsps.
* Here we need to make use of jsp action tags.

**<jsp:useBean> :**

sunmicro systems has given <jsp:useBean> tag to remove java code from jsp. We use <jsp:useBean> tag as

<jsp:useBean id="s" class="info.fls.sls.StoreJB"/>

Whenever jsp compiler encounted a tag,he ll evaluate(coverts to corresponding java code) the tag.when the above tag is evaluated it will create ‘s’ reference variable as shown below.

StoreJB s=new StoreJB();

* When the above <jsp:useBean> tag is evaluated by the jsp compiler,it will create a reference variable ‘s’ and holds StoreJB object.
* The jsp compiler will add ‘s’ reference into request scope by using a key ‘s’.
* Here <jsp:useBean> will not create the object to java bean every time.it checks whether the object is available in the give scope with that key.if it is not available it will create the object to java bean,if it is available it returns the same objectwhich is already available.

|  |
| --- |
| <%@ page import="info.fls.sls.StoreJB"%>  <%  StoreJB p=new StoreJB();  pageContext.setAttribute("s",p);  %>  <jsp:useBean id="s" class="StoreJB"/> |

display.jsp

* In the above code as already StoreJB object is available with the name ‘s’ it will not create the object to StoreJB again,it just return the available StoreJB object.

Following is the piece of code used internally incase of <jsp:useBean>.

|  |
| --- |
| Object o=pageContect.getAttribute("s");  StoreJB s=null;  if(o==null){  s=new StoreJB();  pageContect.setAttribute("s",s);  }  else{  s=(StoreJB)o;  } |

We can call the getter methods on the java bean object using <jsp:getProperty>

**<jsp:getProperty>**

|  |
| --- |
| <jsp:useBean id="s" class="StoreJB"/>  <jsp:getProperty name="s" property="productId"/><br>  <jsp:getProperty name="s" property="productName"/><br>  <jsp:getProperty name="s" property="price"/><br> |

display.jsp

whenever we call <jsp:getProperty> tag it get the value of name attribute and checks ,with that key any data is available in the pagescope.if it is available it call the appropriate getter method on the object associated with that key based on the propertyname.

When the above tag code is executed it will call all the getter method on the appropriate properties and send the data to client.

We can remove the java code to call setter methods by using a jsp action tag <jsp:setProperty>

**<jsp:setProperty>**

Eg: <jsp:setProperty name="s" property="productId" value="100"/>

Here for <jsp:setProperty> value attribute is mandatory. So we can completely remove the java code from jsp’s using jsp action tags.observer the below example.

|  |
| --- |
| <%@ page import="info.fls.sls.StoreJB"%>  <jsp:useBean id="s" class="StoreJB"/>  <jsp:setProperty name="s" property="productId" value="100"/>  <jsp:setProperty name="s" property="productName" value="pone"/>  <jsp:setProperty name="s" property="price" value="1000.00"/>  the following details ares stored in db<br>  <jsp:getProperty name="s" property="productId"/><br>  <jsp:getProperty name="s" property="productName"/><br>  <jsp:getProperty name="s" property="price"/><br>  <a href="product.html">Click here to add another</a> |

display.jsp

if we carefully design the project,we can completely remove the java code from jsps.

If the html form field names and java bean property names are same,the java bean <jsp:setProperty> tag will capture the data and store into instance variables automatically.

Eg:

write product.html form with the property names same as java bean properties.

|  |
| --- |
| <form action="/myproject/display.jsp">  productId:<input type="text" name="productId"/><br>  productName:<input type="text" name="productName"/><br>  price:<input type="text" name="price"/><br>  <input type="submit" value="Login"/>  </form> |

product.html

now write display .jsp to display values from javabean.

|  |
| --- |
| <%@ page import="info.fls.sls.StoreJB"%>  <jsp:useBean id="s" class="StoreJB"/>  <jsp:setProperty name="s" property="\*"/>  the following details ares stored in db<br>  <jsp:getProperty name="s" property="productId"/><br>  <jsp:getProperty name="s" property="productName"/><br>  <jsp:getProperty name="s" property="price"/><br>  <a href="product.html">Click here to add another</a> |

display.jsp

.

**Jsp custom tags:**

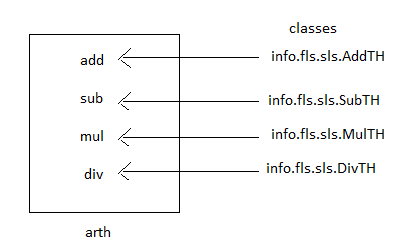
A custom tag library contains group of user defined tags.here we can develop multiple tag libraries.to develop our own tag library we have to follow couple of steps.

1: Identify all the tags which are to be part of our tag library.in my tag library I would like to add add,sub,mul and div tag library.

2: identify functionality of every tag.

3: identify the name to the tag library in my eg we are using the tag library nakme as “arth”.

4: for every functionality,we have to develop taghandlerclass or tag class.



5: we have to develop tld (tag library descriptor) file.

This file contains the mapping information between the tag and tag handler class.

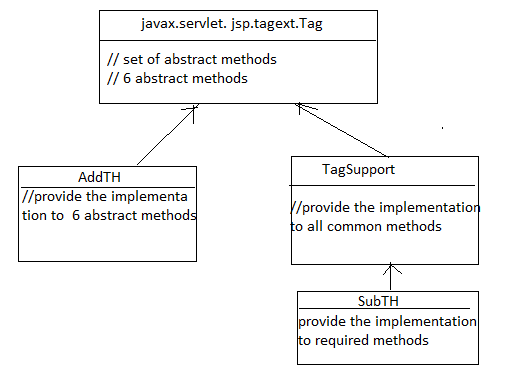
|  |
| --- |
| <taglib>  <tlib-version>1.0</tlib-version>  <jsp-version>1.2</jsp-version>  <short-name>a</short-name>  <uri>http://www.info.fls.sls/tag</uri>  <description>Sample tag library</description>  <tag>  <name>add</name>  <tag-class>info.fls.sls.AddTH</tag-class>  <attribute>  <name>pone</name>  <required>true</required>  </attribute>  <attribute>  <name>ptwo</name>  <required>false</required>  </attribute>  </tag>  </taglib>  arth.tld |

6: we need to develop tag classes or **tag handler classes.**

**Tagclass/tag handler class.**

Sunmicrosystems has released an interface Tag as part of javax.servlet.jsp.tagext.Tag.

Any class that provide the implementation of Tag interface directly or indirectly is called as Tag class.

****

From the above diagram we can understood there are multiple ways are there to develop Tag classes. Most of the people uses TagSupport class to provide the implementation of Tag interface. Observer below taghandler class.

|  |
| --- |
| package info.fls.sls;  import javax.servlet.jsp.\*;  import javax.servlet.jsp.tagext.\*;  import java.io.\*;  public class AddTH extends TagSupport{  int pone;  int ptwo;  public void setPone(int pone){  this.pone=pone;  }  public void setPtwo(int ptwo){  this.ptwo=ptwo;  }  public int doEndTag(){  try{  int result=pone+ptwo;  JspWriter out=pageContext.getOut();  out.println(result);  }  catch(Exception e){  }  return EVAL\_PAGE;  }  }  AddTH.java |

To compile the above tag class we have to specify jsp-api.jar in classpath. Incase of weblogic it is javax.jsp.jar. similarly write tag classes for other operations also.compile all the taglibraries and create a jar file.anybody who releases the taglibraries they must release two files .they are

* a jar file which contains set of class files
* a tld file

procedure to use taglibrary:

step1: copy taglibrary jar files into project lib folder.

Step2: copy a tld file into project WEB-INF folder

Step3:configure tld file in web.xml

|  |
| --- |
| <web-app>  <taglib>  <taglib-uri>http://www.info.fls.sls/tag</taglib-uri>  <taglib-location>/WEB-INF/arth.tld</taglib-location>  </taglib>  </web-app> |

web.xml

step4: import the tag library and use the tags in jsp.

|  |
| --- |
| <%@ taglib uri="http://www.info.fls.sls/tag" prefix="a"%>  <a:add pone="10" ptwo="20"/> |

one.jsp

the above tag results in addition of pone and ptwo data. And o/p will be 30. All the opearational logic behind the execution of tag calling taghandler class will be done by jsp compiler internally. Finally we are getting o/p upon execution of tag.

* Nowadays people who are developing custom tag libraries are not releasing the tld files and jsr files separately,they are placing tld files also into jar files and releasing into market.the advantage here is we can some steps in configuring tld files.

**Procedure to use custom tag library,if the tld files are available in jar file.**

Step1: copy the jar file into project lib folder.

Step2: import the taglibrary in the jsp and use it.(the uri of tld file and uri of jsp’s must be same)

Note: while placing the tld file in a jar file ,we need to place the tld file in META-INF. The frameworks like struts,spring uses taglibraries. Most of the cases we ll not develop tag libraries on our own. This is because alreadt there are so many people has developed their own taglibraries.so we go fro using tag libraries given by som others.

After sunmicrosystems released custom tag library,so many people has started developing their own tag libraries.because of this reason,so many companies repeatedly developing taglibraries for same tags for multiple times.because of this companies are loosing money .to resolve this problem sunmicrosystems has released JSTL specification(Java Standard Tag Library).

Sunmicrosystems has released 4 tag libraries as part of JSTL, they are

* core taglibrary
* xml taglibrary
* sql taglibrary
* FMT taglibrary(I18N)
* **Core taglibrary** contains the most important tags which will be used across multiple projects. Some of the core taglibrary tags are

set,remove, out,if,forEach,catch,choose,when,otherwise,url,redirect etc…

* **Xml taglibrary** contains all the tags which are used to deal with xml files,reads contents from xml files .some of the tags are set,remove,if etc….
* **SQL taglibrary** contains set of tags which are used to interact with database server some of the tags are query,datasource,update etc…..
* **Formatting taglibrary(FMT)** contains set of tags which are used to develop I18N applications some of the tags are message,locate etc……

Apache guys has provided the implementation of sunmicrosystems JSTL.if we want to use JSTL in our project we have to download it from apache.org

*Procedure to use JSTL in out project:*

Copy the jar files into project lib folder then import the taglibrary and use the tags.

**ELExpressions**

In most of the projects we store data into scopped variables.sunmicrosystems has introduced ELExpressions to read the data from scopped variables and display o/p to client.

Sunmicrosystems realized the importance of ELExpressiosn and added ELExpressions as part of jsp 2.0 specification.

Syntax:



The following is an example of using ELExpressions.

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%  request.setAttribute("vone",100);  %>  <c:out value="${vone}"/> |

one.jsp

whenever jsp compiler has encounted ELExpression,it checks in pagescope,if the key is not available in pagescope,it checks in request scope,otherwise in session and followed by application. If the data (key)is not available in any scope, it will display a blank value.(internally it returns null value,onbehalf of null value we are getting displayed blank value).

* We can directly use ELExpressions in jsp page.

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%  request.setAttribute("vone",100);  %>  ${vone} |

one.jsp

* ELExpression can deal with only scopped variables, they cant deal with java variables.

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%  String name="software";  %>  ${name} |

one.jsp

when we execute above jsp it wll display a blank value,why because when the expression is evaluated it checks for the key name in pagescope,requestscope,sessionscope,application scope respectively.as data is not available with that key it displayed a blank value.so it tries to read data from only scopes.

On the ELExpressions we can perform arithemetic,relational,logical opearions.

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%  request.setAttribute("vone",100);  request.setAttribute("vtwo",200);  %>  ${vone+vtwo} |

one.jsp

* we can read the data of any object or key which is available in scopes.

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ page import="java.util.ArrayList"%>  <%  ArrayList al=new ArrayList();  al.add("one");  al.add("two");  al.add("three");  request.setAttribute("list",al);  %>  ${list} |

one.jsp

* ELExpressions are also deals with javabeans.we can use ELExpressions on javabeans if they stored in scopped variables.

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ page import="info.fls.sls.\*"%>  <%  Address a=new Address();  a.setStreet("apet");  a.setCity("hyd");  a.setState("AP");  Employee e=new Employee();  e.setEmpNo(120);  e.setName("ramu");  e.setAddress(a);  request.setAttribute("empdata",e);  %>  ${empdata.empNo}<br>  ${empdata.name}<br>  ${empdata.address.street}<br>  ${empdata.address.city}<br>  ${empdata.address.state} |

one.jsp

in ELExpressions there are 6 implicit variables they are

* pageScope
* requestScope
* sessionScope
* applicationScope
* param
* header

eg:

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%  pageContext.setAttribute("vone",100);  request.setAttribute("vone",200);  session.setAttribute("vone",300);  application.setAttribute("vone",400);  %>  ${requestScope.vone} |

one.jsp

when the above jsp is executed ,it check for the key vone in requestScope if it is not available it display a blank value.

**param**

this implicit variable is used to capture the data from form and display to client.



|  |
| --- |
| <%@ page isELIgnored="false"%>  ${param.uname}<br>  ${param.pwd} |

one.jsp

**header**

this implicit variable is used to read the values of http headers and display to the client.

|  |
| --- |
| <%@ page isELIgnored="false"%>  ${header['user-agent']}<br>  ${header['accept-language']} |

one.jsp

**core tag library**

**out:**

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  <%  request.setAttribute("name","raju");  %>  <c:out value="${requestScope.name}"/> |

one.jsp

o/p:raju

**set:**

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  <c:set var="name" value="gopi"/>  <c:out value="${pageScope.name}"/> |

one.jsp

o/p:gopi

when the above tag is evaluated it stores data into scopped variable as we havnot specified any scope it store in default page scope.it uses the key name to store the data into scopped variables.if we want to store the data into specific scope we use attribute ‘scope’.

**remove:**

this tag is used to remove data from specific scope.

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  <%  request.setAttribute("name","raju");  %>  ${name}  <c:remove var="name"/>  ${name} |

one.jsp

o/p: raju

if we want to remove data from a specific scope ,we need to use an attribute scope.generally every tag has tag body.if the condition is successfully evaluated,it will execute tagbody,otherwise it will not execute tagbody.

**if:**

this tag is used to check the conditions or evaluate expressions.

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  <c:if test="${10<30}">  we are in if block  </c:if> |

one.jsp

o/p: we are in if block.

the mandatory attribute for ‘if’ is test.for this we can provide an ELExpression results Boolean value.here test expression should be ELExpression otherwise it will not evaluate the tag.

**forEach:**

this tag is used to iterate the elements through a specific begin idex to end index.

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  <c:forEach var="a" begin="1" end="10" step="2">  <c:out value="${a}"/><br>  </c:forEach> |

one.jsp

o/p:

1

3

5

7

9

We can use forEach tag to iterate elements through collection objects.

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  <%  java.util.ArrayList al=new java.util.ArrayList();  al.add("one");  al.add("two");  al.add("three");  al.add("four");  request.setAttribute("list",al);  %>  <c:forEach var="a" items="${list}">  <c:out value="${a}"/><br>  </c:forEach> |

o/p:

one  
two  
three  
four

in EL expressions,we can use **empty** variable to check whether the give list is empty or not.

|  |
| --- |
| <c:if test="${empty lis}">  welcome  </c:if> |

one.jsp

o/p: welcome

**forTokens:**

eg:

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  <%  String name="jdbc:oracle:thin:@localhost";  request.setAttribute("name",name);  %>  <c:forTokens var="a" items="${name}" delims=":">  <c:out value="${a}"/><br>  </c:forTokens> |

one.jsp

**choose,when,otherwise:**

we have to use all these tags combinely.by using these tags we can achieve if ,else,switch conditions.

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  <%  int i=10;  request.setAttribute("i",i);  %>  <c:choose>  <c:when test="${i==1}">  i value is 1  </c:when>  <c:when test="${i==5}">  i value is 5  </c:when>  <c:when test="${i==10}">  i value is 10  </c:when>  <c:otherwise>  value not matching  </c:otherwise> |

one.jsp

o/p: I value is 10

**catch:**

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  <c:catch var="i">  <%  int i=10/0;  %>  </c:catch>  <c:out value="${i}"/> |

one.jsp

o/p: java.lang.ArithmeticException: / by zero

**import**

this is used to include the o/p of two jsps and sen the o/p to client.

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  we are in one.jsp<br>  <c:import url="two.jsp"/> |

one.jsp

o/p: we are in one.jsp output from two.jsp

**redirect**

this is used to redirect the request from one resource to another resource. This is an alternative to response.senRedirect().

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  <c:redirect url="http://www.google.co.in"/> |

one.jsp

**sql** **tag library**

this tag library is used to write jdbc code as part of jsp.

**setDataSource**

using this we can create a datasource with the specified credentials for all the properties required for database connectivity.

Eg:

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>  <sql:setDataSource var="ds" driver="oracle.jdbc.driver.OracleDriver"  url="jdbc:oracle:thin:@localhost:1521:xe"  user="system"  password="prasad"/>    <sql:update dataSource="${ds}">  insert into emp values(101,'raju',10000)  </sql:update> |

one.jsp

**update**

this tag is used to send update queries(insert,delete,update) to database server.

**query**

this tag is used to send select queries to database server.

Eg:

|  |
| --- |
| <%@ page isELIgnored="false"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  <sql:setDataSource var="ds" driver="oracle.jdbc.driver.OracleDriver"  url="jdbc:oracle:thin:@localhost:1521:xe"  user="system"  password="prasad"/>    <sql:query dataSource="${ds}" var="rs" >  select \* from emp  </sql:query>  <c:forEach var="i" items="${rs.rows}">  ${i.eno} ${i.name} ${i.salary}<br>  </c:forEach> |

one.jsp

**param**

this tag is used to set the place holder values in queries.

Apart from these tags ,there are some other tags also available,the usage of those tags is based on on the requirement.