Month Planner JANUARY

Sun 1 ..... Mon 16

Mon 2 ..... Tue 17

Tue 3 ..... Wed 18

Wed 4 ..... Thu 19

Thu 5 ..... Fri 20

Fri 6 ..... Java DataBase Connectivity (JDBC) ..... Sat 21

Sat 7 ..... Sun 22

Sun 8 ..... Mon 23

Mon 9 ..... Tue 24

Tue 10 ..... Wed 25

Wed 11 ..... Thu 26

Thu 12 ..... Fri 27

Fri 13 ..... Sat 28

Sat 14 ..... Sun 29

Sun 15 ..... Mon 30

..... Tue 31

Scanned by Scanner Go

January Sunday 1st 2006

JDBC (Java Database connectivity)

Jdbc is an open specification (API). By using Jdbc API we can connect to any database which is available in this world.

Some of the popular database servers are Oracle, Mysql, Sybase, DB2, ingrus, point, sql server etc.,

Every database vendor releases 2 types of slw. they are server software & client software. the server slw will be install on server computer and client slw will be install on the developers computer.

the database administrator is responsible to install database server slw on the server computer. This server computer will be connected in network. we uniquely identify the server computer by using 'Ip Address'.

Database administrator provides 'database service name', 'port number', at the time of installing the software.

If any one wants to communicate with database server we need the following details

(i) Ip Address (local host)
(ii) service name (xe) → express edition
(iii) port number (1521)

Engagements

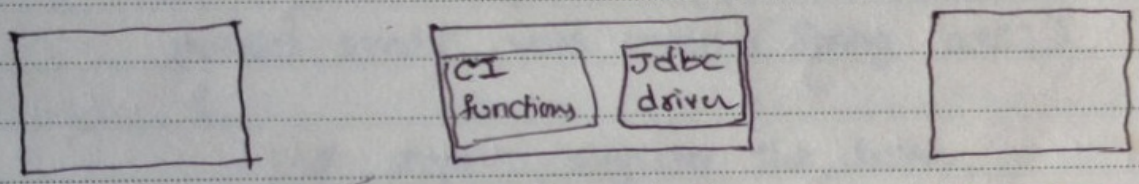
Scanned by Scanner Go

January Tuesday 3rd 2006

Initially Microsoft has come up with an API called as ODBC API (Open database Connectivity). By using ODBC API we can develop a C program to communicate with any database without any code change.

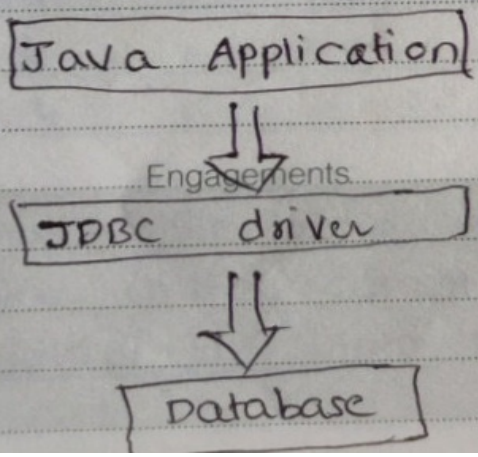
JDBC API :

This API is released by sunmicrosystem. This API is used to develop a java program to communicate with any database server without changing the java code.



sun microsystem has released jdbc API in 2 different packages: java.sql, javax.sql these packages are part of JSE API.

If any java program needs to establish a connection with database we require a jdbc driver



Scanned by Scanner Go

January Thursday 5th 2006

procedure to develop a jdbc Application

(1) Register the driver
(2) Get the connection
(3) create the statement
(4) execute the query
(5) close the connection

(1) Register the driver:

```
import java.sql.*;
public class DBConnect
{
    public static void main (String arr[])
    {
        // step1 : Register the driver
        try
        {
            Driver d = new oracle.jdbc.driver.
                                OracleDriver();
            DriverManager.registerDriver(d);
            System.out.println("Driver is registered");
        }
        catch (SQLException e)
        {
        }
}
```

Engagements

To compile the above program we need Oracle jdbc driver in the current working directory i.e., copy ojdbc14.jar into the current working directory.

Scanned by Scanner Go

set the classpath to ojdbc14.jar

c:\> set classpath = c:\oradexe...\ojdbc14.jar;.;

The meaning of registering the driver is loading all the required jdbc driver classes into the JVM's memory.

(2) get the connection:

syntax:

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

ex:

Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe", "scott", "tiger");

Here our java application requests Jdbc driver to establish a connection with database. Jdbc driver takes the 3 parameters above and tries to establish a connection with database.

If the driver succeeded in establishing a connection, it returns Connection object to the Jdbc program. If it fails in establishing a connection, then it throws SQLException.

Scanned by Scanner Go

(3) create the statement

To send any information to the database we need to create the statement object.

Statement stmt = con.createStatement();

There are 3 different types of statement objects They are (i) Statement
(ii) PreparedStatement
(iii) CallableStatement

we will use Statement / PreparedStatement to send sql queries to the database. Among these two preparedStatement improves the performance. we will use CallableStatement to call procedures.

(4) Execute the query :

In java the queries are classified into 2 types : select queries, Non Select queries.
To execute select queries, we use executeQuery() method.
To execute Non Select queries, we use executeUpdate() method.

Syntax : stmt.executeUpdate(sql query)

Engagements

Note:
Select queries are the queries which starts with 'select' keyword.

Scanned by Scanner Go

(5) close the connection:

As a programmer we are responsible to provide the code to close the connection. If we did it provide the code to close the connection, someday database server will not give a connection to any other program.

To close the connection we have to use a method close() on the connection object.

example:

```
public class CreateTable
{
    public static void main(String args[])
    {
Step1 →  DriverManager.registerDriver(new
                oracle.jdbc.driver.OracleDriver());

Step2 →  Connection con = DriverManager.gerConnection
         ("jdbc:oracle:thin:@localhost:1521:xe",
                "scott", "tiger");

Step3 →  Statement st = con.createStatement();
Step4 →  st.executeUpdate("Create table emp(
           empno number(10), name varchar2(10));

Step5 →  con.close();

    }
}
```

Engagements

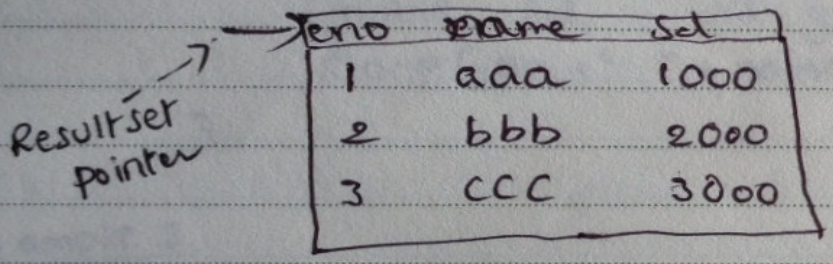
Scanned by Scanner Go

executeQuery() Method :

To execute the select query we will use executeQuery() method. This method takes select query as input and returns ResultSet object.

ResultSet executeQuery (String query)

When we ~~send~~ got the ResultSet object, as part of it, it contains ResultSet pointer. By default ResultSet pointer points to the before the 1st record.



ResultSet object

By using a java application's next() method we will move the pointer to 1st row.

Syntax of next()

Engagements

```
boolean next()
```

- if it returns true --> next record available
  false --> Not available

Scanned by Scanner Go

To get the data from a specific columns we use the appropriate getXXX() methods

examples: getInt()
getString()
getDouble()

example2

```
ResultSet rs = stmt.executeQuery("select * from emp");
while(rs.next())
{
    int no = rs.getInt("eno");
    String name = rs.getString("name");
    Double sal = rs.getDouble("salary");
    s.o.p(no+" "+name+" "+sal);
}
```

example 3

```
int no = rs.getInt(1);
```

Assignment

① Develop a jdbc appln that creates a table name 'employee' with following columns empno, ename, sal, DOJ

② Develop an application that retrieves all employee records who empno=1

Engagements

Scanned by Scanner Go

January Tuesday 17th 2006

Prepared Statement :

It is also used to send sql queries to the database. prepared statement improves the performance by using Bind variables of database.

Procedure to develop Jdbc Application which uses prepared statement:

step1: create the prepared statement object by supplying a query with ?
(Positional Parameter)

```
PreparedStatement pstmt = con.prepareStatement
("insert into emp values(?,?,?)");
```
pp1 pp2 pp3

step2: supply the values to the ? by using getXXX methods.

ex:
```
pstmt.setInt(1,20)
pstmt.setString(2,"aaa");
pstmt.setDouble(3,1000);
```

In prepared statement queries, index starts with '1'

Assignment

Engagements

Develop a Jdbc program which can retrieve a record from emp table whose empno = 3 (use prepared statement)

Scanned by Scanner Go

January Thursday 19th 2006

Callable Statements:

Sometimes we need to pass procedures to database server. This task is achieved through callable statements

```
SQL> create or replace procedure p1
     as
     begin
          insert into emp values (1, 'eone', 1000);
     end p1
     /

D:\> Edit ↵
```

```
CallableStatement cstmt = con.prepareCall("{call p1}");
cstmt.execute();

        (or)

CallableStatement cstmt = con.prepareCall
                ("{call p1(?,?,?)}");
cstmt.setInt(1, 10);
cstmt.setString(2, "aaa");
cstmt.setDouble(3, 1000);
cstmt.execute();
```

Engagements

Scanned by Scanner Go

System properties :

System properties are used to remove hard coded values.

In Jdbc program url, username, password are hard coded values.

```
import java.sql.*;
public class insert {
    public static void main (String arr[])
    {
        DriverManager.registerDriver(...);
        String url = System.getProperty("url");
        String uname=      "      ("uname");
        String pwd =       "      ("pwd");
    Connection con= DriverManager.getConnection
                    (url, uname, pwd);
    Statement stmt = con.createStatement();
    stmt.executeUpdate("insert into emp values
                        (1, 'aaa', 1000);
    }
}
```

To run the above program we have to supply 3 system variables as follows

Engagements

```
D:\> java -Durl = jdbc:oracle:thin:@localhost:1521:xe
          -Duname = System
          -Dpwd = admin      insert
```

Scanned by Scanner Go

Types of ResultSet: Forward only & Bidirectional

By default the behavior of ResultSet object is Forward only Result set.

To create the Bidirectional ResultSet we need to supply 2 arguments to the createStatement() object.

argument1

- ResultSet.TYPE_FORWARD_ONLY ← Default
- ResultSet.TYPE_SCROLL_INSENSITIVE
- ResultSet.TYPE_SCROLL_SENSITIVE

argument2

- ResultSet.CONCUR_READ_ONLY ← Default
- ResultSet.CONCOR_UPDATABLE

INSENSITIVE

Database updations will not effects the Resultset Data

SENSITIVE: Database updations will ~~not~~ effects the Resultset data.

Updatable:

Resultset updations will effects the database.

Scanned by Scanner Go

JDBC Drivers:

The following 4 types of drivers are available in jdbc.

1. Type 1 driver (JDBC-ODBC Bridge)
2. Type 2 driver (Java Native Driver)
3. Type 3 driver (Network protocol Driver)
4. Type 4 driver (Thin Driver)/(Pure Java Driver)

Type1 driver:

1. To use this Driver we need to install the following s/w
   - DB client s/w
   - ODBC drivers (other than windows OS)
   - appropriate Jdbc driver
2. It is a platform dependent.

Type2 driver:

1. In this we need to install the client software.
2. No Company will use this because this driver is going to change the behavior of java applications.
3. It is also platform dependent.

Type 4 driver:

Engagements

Here we use pure java code to communicate with database. and No need to perform any configurations.

Scanned by Scanner Go

January Friday 27th 2006

TYPE-3 Driver (Network protocol driver)

This driver is mainly used in the projects which are mainly depepadent on Security. This driver is placed behind the firewall and inside the server.

TYPE 3 driver internally uses TYPE-4 driver.

Engagements

Scanned by Scanner Go

Servlets

Scanned by Scanner Go

Sun Microsystems has released 3 different types of java applications.

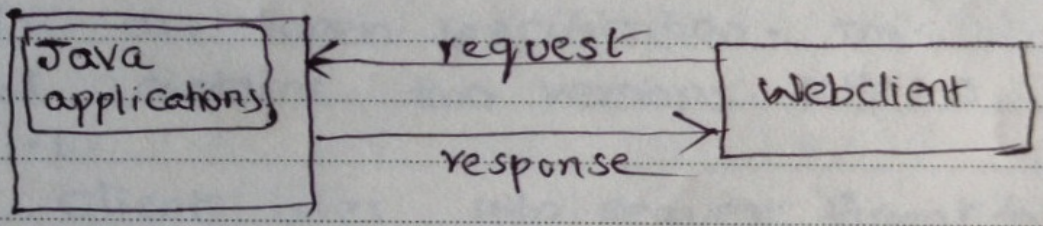
They are

(i) Standalone applications
(ii) Applet Applications
(iii) Web based Applications.

→ Standalone applications run on a specific computer. ex: Systems used in Bank, Railways.

→ Initially Sun Microsystems has released applets to work with web based applications. Applets run on client computers.

→ Web based applications are dependent on client server computing. We are going to develop the applications which runs inside the server.



Webserver is also a java program. A webserver program can take multiple requests at the sametime and process them.

There are somany webservers available in the market. examples are

- (Apache) Tomcat
- Weblogic (Bea)
- Websphere (IBM)
- JBoss (open source)
- Sun one server (Sun)

(Sun microsystems)

Engagements

Scanned by Scanner Go

Web clients :

There are so many Webclients are available in this market. They are Internet explorer, Mozilla, opera, Netscape Navigator etc.,

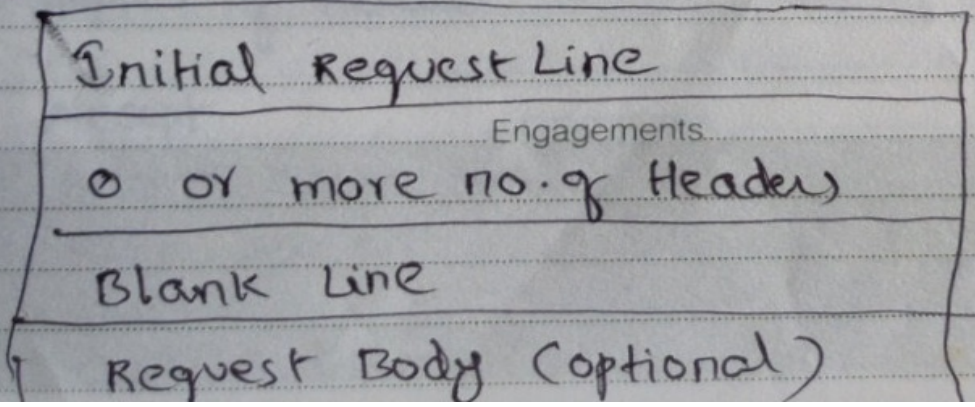
A web client sends request to the server. Webserver receives the request and process the request, and sends response to the client.

If client and server wants to communicate with each other they have to use same protocol. In Webbased applications we use Http protocol (Hypertext transfer protocol).

If we are having http server, any Http client can communicate with it. Http is an open specification. This protocol contains two versions. Http 1.0, Http 1.1.

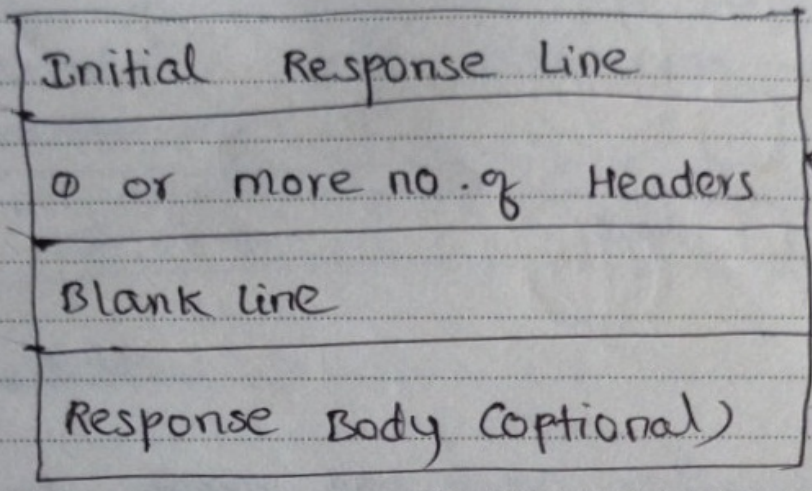
Client uses Http Request format to send requests and server uses Http Response format to ~~receive~~ send response.

Http Request Format :



Scanned by Scanner Go

Http Response Format:



Initial Request line:

Method Name	Request URI	protocol/version
GET	/one.html	HTTP/1.1

eg:

The method names can be any of the following methods.

(1) GET (2) PUT (3) POST (4) DELETE
(5) TRACE (6) LOCATE (7) HEAD

Header format: Header Name : Header Value

Examples of Headers

ACCEPT
ACCEPT_LANGUAGE
USER_AGENT
HOST
CONNECTION

Engagements

Scanned by Scanner Go

Initial Response Line

Protocol / version NO	status code	Status Message
ex: HTTP/1.0	200	OK

status codes

1xx → server has given information to client
2xx → the requested resource is available in server
3xx → " " " was redirected to another server
4xx → requested resource is not available in the server.
5xx → requested resource is being executed and if it encounters a problem then server sends 5xx error message.

Methods

GET: the client uses GET() if you want to process a resource in the server.
PUT: if you want to place a file inside a server. (No servers will support this - security)
DELETE: This is used to delete the resource from the server. (No server will support this)
LOCATE & TRACE: used to search an item inside the server.
(No servers will support this)
HEAD: used to get the Header information of the server to the client.

Engagements

Scanned by Scanner Go

By default When we submit the form it uses a method GET() as default. If we are using GET(), then browser captures the data from the Html form and append the data to Browser URL.

ex:
=>
http://localhost:9000/two.html?uname=vone&pwd=vpassword

TO overcome this problem, we can use a method "post". so that the data will be sent to the server by appending the data to the Request body.

we are going to develop the applications which runs inside the server. If you developed some application for 1 Server we can deploy the same application in any of the server.

Procedure to develop 1st Webbased application:

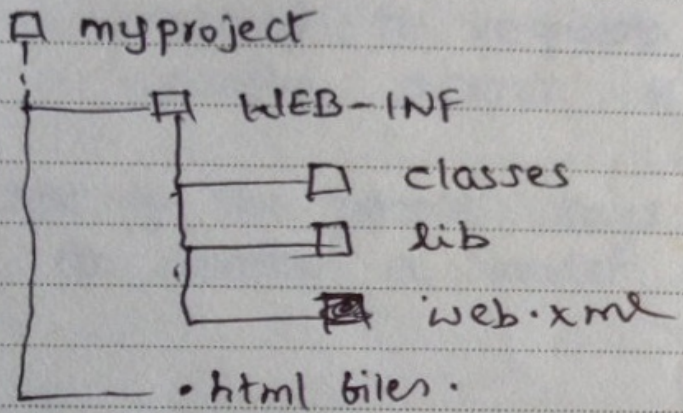
(1) create a folder with our project name
(2) Inside the above created folder create another folder with the name WEB-INF.
(3) Inside WEB-INF, create 2 folders with the names classes, lib.

Engagements

(4) Inside the WEB-INF folder, we need to create an xml file web.xml (deployment descriptor).

Scanned by Scanner Go

(5) Develop all the required html files and place them inside the project folder.



Procedure to deploy the project in Tomcat Server

(1) Install Tomcat server with specific port number
(2) copy the project inside webapps of Tomcat
(3) Run the tomcat server.

procedure to deploy in weblogic Server:

(1) create a domain in weblogic Server
(2) As part of weblogic folder copy the deployed project
The name of the project in weblogic is 'auto deploy'.
(3) Start the weblogic Server.

Scanned by Scanner Go

Servlet:

A servlet is a small java program that runs within a webserver. Servlets receive and respond to requests from web clients, usually across HTTP.

Most of the people says we have 3 options to develop a Servlet program.

option1:

implementing Servlet interface

```
import javax.servlet.*;
public class FirstServlet implements Servlet
{   // 5 methods: init, service, destroy, getServletConfig, getServletInfo
    // implementation to all 5 methods
}
```

Option 2:

By extending Generic servlet class which implements Servlet interface.

Generic Servlet is an abstract class. So we must implement service() method in this option.

```
import javax.servlet.*;
public class FirstServlet extends GenericServlet
{
  public void service(ServletRequest rq, ServletResponse rs)
  {
      // code
  }
}
```

Engagements

Scanned by Scanner Go

option 3:

By extending HttpServlet class. This class inherits the properties of GenericServlet. Httpservlet is an abstract class.

```
import javax.servlet.http.*;
public class FirstServlet extends HttpServlet
{
    public void service(HttpServletRequest rq,
                        HttpServletResponse rs)
    {
      // code
    }
}
```

option 4: By extending from an already implemented Servlet class

```
public class secondServlet extends FirstServlet
{   public void service(----)
    {
    }
}
```

→ Like this we can create the servlet classes in so many ways.

→ we can develop a Servlet program which can run on any Server.
→ unfortunately, only the servers which supports HttpProtocol has provided the implementation to Servlet API.

Engagements

Scanned by Scanner Go

The servlet Life cycle :

The following are known as life-cycle methods and are called in the following sequence

(1) init()
(2) service()
(3) destroy()

init : GenericServlet contains 2 versions of init.

(i) void init(ServletConfig sc) throws ServletExcepti

This method is called by the ServletContainer to indicate to a servlet that the servlet is being placed into service.

This is the (second) (First) version of init(). This version is used when the servlet needs to read server-specific settings before it can complete the initialization. like database settings, password files etc.,

(ii) Public void init() throws ServletException
{
// initialization code.
}

This is the (1st) (2nd) version of init(). This version is used when the servlet does not need to read any settings that can vary from server to server.

The init() is called when the servlet is first created.

Engagements

Scanned by Scanner Go

The service Method:

Each time the server receives a request for a servlet, the server spawns a new thread and calls service().

The service Method checks the HTTP Request type(GET, POST, PUT, DELETE etc.) and calls appropriate doXXX method.

The destroy Method:

Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.

```
Void destroy()
{
    // code
}
```

Engagements

Scanned by Scanner Go

Servlet program to get data from database 'product'

```
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;

public class DB extends HttpServlet {
    public void Service(HttpServletRequest rq,
                        HttpServletResponse rs)
{ try {
        PrintWriter out = response.getWriter();
DriverManager.registerDriver(new oracle.jdbc.driver.
                                    OracleDriver());
Connection con = DriverManager.getConnection("jdbc:oracle:thin:
        @localhost:1521:xe", "system", "admin");
  Statement st = con.createStatement();
ResultSet rs = st.executeQuery("select * from product");
 While(rs.next())
  {     out.println(rs.getString(1));
        out.println(rs.getString(2));
  }
 } catch(Exception e) { }
} }
```

→ At the time of compiling this program set class path to ojdbc14.jar, servlet-api.jar.

→ copy ojdbc14.jar file inside lib folder of your project Engagements

Scanned by Scanner Go

February Thursday 23rd 2006

Web.xml file (Deployment Descriptor)

```
<web-app>

  <servlet>
      <servlet-name> name of servlet </servlet-name>
      <servlet-class> servlet class name </servlet-class>
  </servlet>

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

</web-app>
```

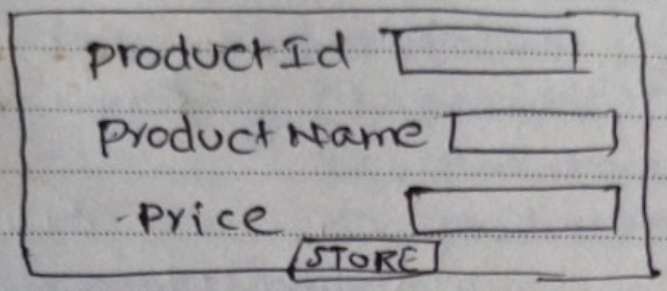
This web.xml file is also called as Deployment Descriptor which decides how to execute the program.

Engagements

Scanned by Scanner Go

February Saturday 25th 2006

creating the following prototype




```
<html>
<head> <title> This form is used to add new product </title>
</head>
<body>
<form action = " /webapp/sd " >
 product Id : <input type = "text" name = "pid" /><br>
 Product Name : <input type = "text" name = "pname" /><br>
 product price : <input type = "text" name = "price" /><br>
 <input type = "submit" value = "store" />
</form>
</body>
</html>
```

foldername → web.xml url pattern

store.java

```
public class store extends HttpServlet
{
   public void service ( ... ) throws IoException, ServletException
   {
      String productId = req.getParameter("pid");
             productName = req.getParameter("pname");
             productprice = req.getParameter("price");
      try {
          // register Driver code
```

Engagements

Scanned by Scanner Go

```
// get the connection code
PreparedStatement pstmt = con.prepareStatement ("
    insert into product values (?,?,?)");
    pstmt.setString (1, productid);
          "          (2, productnae);
          "          (3, productprice);
    pstmt.executeUpdate();
}
    catch( SQLException e)
    { }
    PrintWriter out = response.getWriter();
      out.print ("product Id is" + productid);
          :
}
}
```

Web.xml

```
          :
    <url-pattern> /sd </url-pattern>
          :
```

→ Web browser (client) is responsible to capture the data from the form and place it in HttpRequest format and sends it to the server

→ Server receives HttpRequest format and create the request object and places the data of form inside request object in(key, value) pairs

→ By using req.getParameter(key) → we can extract the value of that key. return type is string.

Engagements

Scanned by Scanner Go

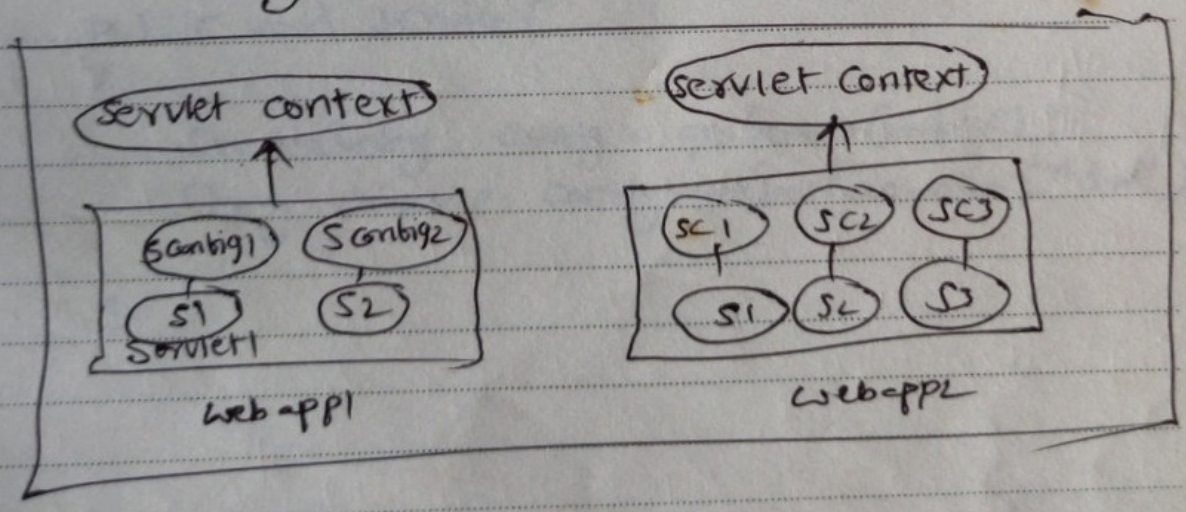
ServletConfig & ServletContext

⇨ server is responsible to create ServletConfig and ServletContext objects. (Application objects)

We will be having 1 ServletContext object per web application. Server creates the ServletContext object when we deploy the project in the server. server removes this object when we un deploy this Project.

ServletContext object is used to store the data. this data can be used by all the resources that are available ~~for~~ in this project.

⇨ Every servlet will have one servletConfig object. the server will create ServletConfig object when it has created the object for servlet. when it removes the servlet object then it removes servletConfig object also.



Scanned by Scanner Go

ServletContext & ServletConfig objects are used to remove HardCoded values.

Procedure To use servletConfig object we have to Configure the init parameters inside the web.xml.

web.xml

```
<servlet>
  <servlet-name> fs </servlet-name>
  <servlet-class> FirstServlet </servlet-class>
  <init-param>
      <param-name> drv </param-name>
      <param-value> oracle.jdbc.driver.OracleDriver
                                        </param-value>
  </init-param>
    :
</servlet>
```

ServletProgram

```
    :
public void service(...)
{
    ServletConfig config = getServletConfig();
    String driver = config.getInitParameter("drv");
      :
}
```

Scanned by Scanner Go

Procedure to use ServletContext object:

To store the data inside ServletContext object we specify the information inside web.xml

web.xml

```
<web-app>
<context-param>
        <param-name>drv</param-name>
        <param-value>oracle.jdbc.driver.OracleDriver
                                        </param-value>
</context-param>
        :
</web-app>
```

To get the ServletContext Object 1st we need to get Servlet Config object.

Servlet

```
public void service(---)
{
        ServletConfig config = getServletConfig();
ServletContext application= config.getServletContext();
String name= application.getInitParameter("drv");
```

Scanned by Scanner Go

Month Planner MARCH

Wed 1	Thu 16
Thu 2	Fri 17
Fri 3	Sat 18
Sat 4	Sun 19
Sun 5	Mon 20
Mon 6	Tue 21
Tue 7	Wed 22
Wed 8	Thu 23
Thu 9	Fri 24
Fri 10	Sat 25
Sat 11	Sun 26
Sun 12	Mon 27
Mon 13	Tue 28
Tue 14	Wed 29
Wed 15	Thu 30
	Fri 31

JSP
(Java Server Pages)

Scanned by Scanner Go

March Wednesday 1st 2006

JSP

As an alternative Technology to servlet, sun Micro systems has released JSP Technology. JSP's are used to develop server side applications. JSP takes less amount of time (development time) when compared with servlets.

Advantages of JSP:

Adv:1

If we use the JSP's in the project it will improve the productivity of developer. (efficiency)

Adv:2:

we can develop JSP's without learning Java code.

Adv:3:

By using JSP's we can seperate our Business logic & presentation Logic

Procedure to develop a JSP program:

1. Every jsp program ends with an extension .jsp

Engagements

2. Jsp program will be placed inside the project folder and outside to the WEB-INF folder.

Scanned by Scanner Go

March Friday 3rd 2006

Back ground procedure of 1st time requesting a JSP

one.jsp
↓
JSPC
↓
one_jsp.java
↓
javac
↓
one_jsp.class
↓
servlet object is created.

JSP Elements:

(1) Template Text
(2) Scriptlet
(3) JSP Declarations
(4) JSP Directives
(5) JSP Expressions
(6) JSP Action Tags
(7) JSP Custom Tags
(8) JSP EL Expression

Engagements

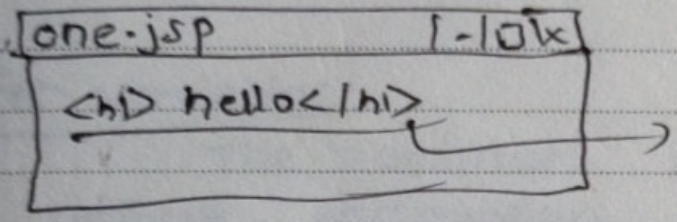
We can provide above JSP Elements as part of Every jsp program.

Scanned by Scanner Go

March Sunday 5th 2006

(i) Template Text :

If any text that is directly written inside jsp program, it is called Template Text.



Whenever the jsp compiler encounters a Template Text, the jsp compiler places it inside out.write() method; inside _JSP.Service().

(ii) Scriptlet :

It is used to write the Java code inside the JSP file.

Syntax:

start & end of scriptlet
<%
// Provide java code
%>

Engagements

ex
<%
int a=10;
System.out.println(a);
%>

Scanned by Scanner Go

Whatever the code we are provided inside the scriptlet, Jsp compiler will place the java code inside _JSPService() method of Generated Servlet.

implicit variables:

The variables which are able to use without declaration are called implicit variables.

```
<%  String s1 = "hello";
    out.print(s1);      // out is implicit var'.
%>
```

There are 9 implicit variables: request, response, pageContext, session, application, config, out, page, exception.

(iii) JSP Declarations

It is used to create instance variables (or) instance methods (or) static variables (or) static methods.

syntax of JSP Declarations:

```
<%!
    instance variables
    instance methods
    static variables
    static methods
%>
```

Engagements

Scanned by Scanner Go

March Thursday 9th 2006

ex:
```
<%!
    int a;
    static int b;
%>
<%!
    public void methodOne()
    {
    }
    public static void methodTwo()
    {
    }
%>
```

Whenever the JSP compiler encounters Jsp declaration it places the code inside the generated class.

Assignment: create a JSP that calls an instance method whenever user sends request to the server

```
<%! int a;
    public void methodone()
    { System.out.println("in methodOne()");
    }
%>
<%
    methodone();
%>
```

Engagements

Scanned by Scanner Go

March Saturday 11th 2006

JSP Life cycle Methods:

jspInit(), _jspService(), jspDestroy()

(iv) JSP Expressions:

JSP Expressions are alternative to Java Expressions.

Some Examples of Java Expressions

```
<%  int a=10,b=20;
    out.print(a);
    out.print(a+b);
    out.print("aitam".equals("AITAM"));
%>
```

syntax of jsp expressions:

<%= Java Expression %>

Ex:

<%= s1.equals("abc") %>

⇓

JSPC

⇓

out.print(s1.equals("xyz"));

Engagements

Scanned by Scanner Go

(v) JSP Directives:

It is an instruction which can give to JSP compiler.
The following are 3 JSP Directives.

(a) page Directive
(b) include Directive
(c) Taglib Directive

syntax: <%@ Directive-name optional parameters %>

example: <%@ page ... %>

(a) page Directive:

language page directive

one JSP

```
<%@ page language = "java script" %>
<%
    var a;
    a=10;
%>
```

above code returns an error because As of today No Server is supporting java scripting as part of scriptlets.

Engagements

Default one is "Java".

Scanned by Scanner Go

import page directive:

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

The import page directive instruction is used to add an import statement to the jsp program.

info page directive:

It tells about the purpose of jsp file. It is converted as getServletInfo() in the generated servlet by jspc.

<%@ page info = "This is used to store data in DB" %>

contentType page directive

If the JSP wants to send a specific o/p to client, we will use contentType page directive.

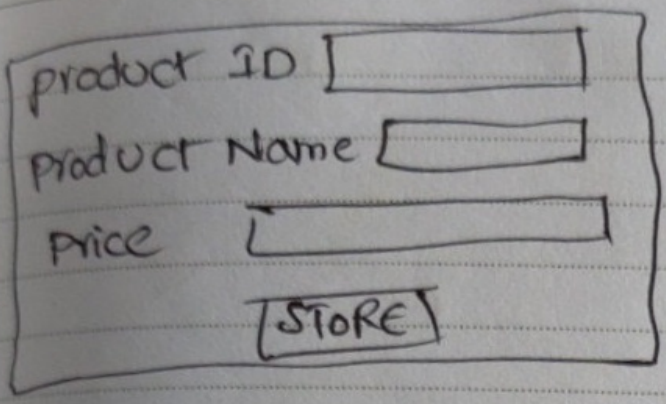
<%@ page contentType = "text/html" %>

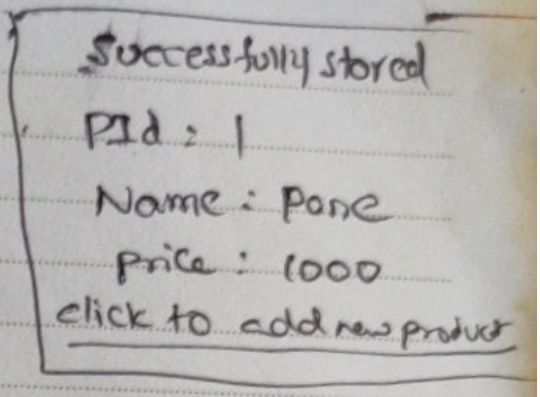
<h1> Hello </h1>

Whenever the jsp compiler encounters contentType page directive it will add response.setContentType

Engagements

Scanned by Scanner Go





store.jsp :

```
<%@ page import = "java.sql.*" %>
<% String pid = request.getParameter("pid");
   pname =                    ("pname");
   price =                    ("price");
   // code of jdbc including preparedstmt
   pstmt.executeUpdate();
   con.close();
%>
Successfully stored
pid: <%= pid %> <br>
Name: <%= pname %> <br>
price: <%= price %> <br>
<a href = "/webappl product.html"> click to add new product </a>
```

If user entered wrong datatype value in the form, application has displayed the exceptions to the user. we should not display exceptions to the user.

Instead of that we should display a proper error menage. This can be done with JSP error pages.

Engagements

Scanned by Scanner Go

March Sunday 19th 2006

procedure to develop an error page:

step1:
create a jsp file with the content to be displayed to the client. use isErrorPage page directive to indicate jsp is an error page.

ex:
```
<%@ page isErrorPage = "true" %>
A problem occured in your project.
contact system administrator.
```
error.jsp

step2: In target jsp, we need to include errorpage page directive.

ex: `<%@ page errorpage = "/error.jsp" %>`

Web.xml:

```
<web-app>
    <welcome-file-list>
        <welcome-file> /webapp/product.html
        </welcome-file>
        <welcome-file> /webapp/product.htm
        </welcome-file>
    </welcome-file-list>
</web-app>
```

Engagements

Scanned by Scanner Go

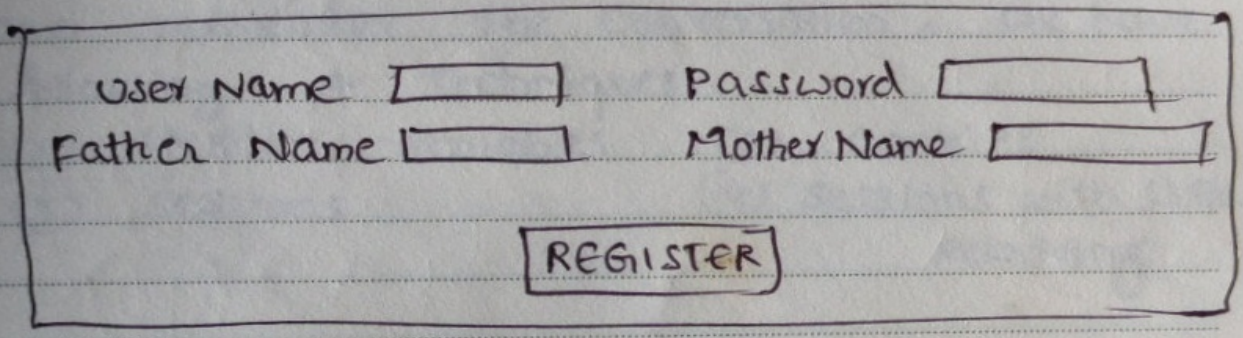
March Tuesday 21st 2006

Stateless protocol

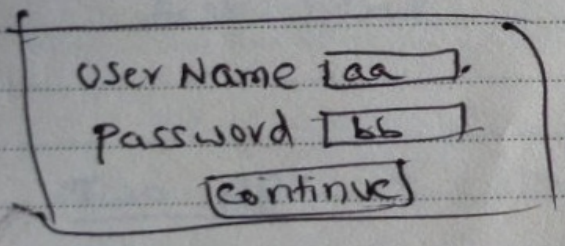
HTTP protocol is a stateless protocol. Http protocol cannot remember the conversation which has happened in the client. It can remember the very recent / last conversation only.

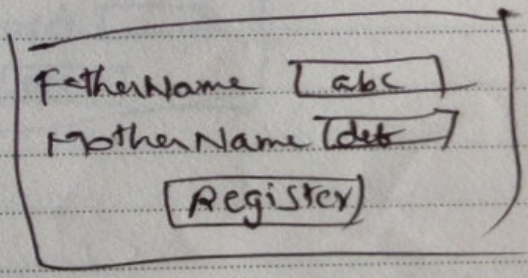
Req: Designing a Registration form.

Approach 1: Design a form that contains all the fields which will capture from the user.



Approach 2: capture the data from the user in multiple stages:





To develop above approach 2 scenario we have to provide 3 programs.

Engagements

one.html
two.html
store.jsp

Scanned by Scanner Go

store.jsp

```
<%= request.getParameter("uname") %> <br>
<%=          "          ("pwd")  %> <br>
<%=          "          ("fname") %> <br>
<%=          "          ("mname") %> <br>
```

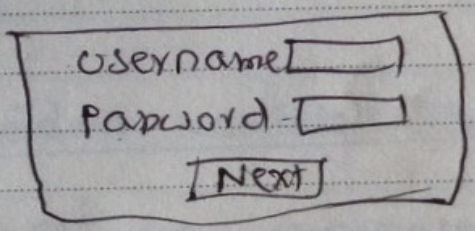
the above jsp returns null, null, abc, def. we got null values because the server is able to remember last conversation only.

TO develop an application which can remember the conversation, we have following 4 Techniques:

(1) Hidden variables (2) cookies
(3) sessions (4) sessions with URL Rewriting

Hidden variables :

first.html



two.jsp:

```
<% String uname = request.getParameter("uname")
           pwd                          ("pwd");
%>
```

Engagements

```
<html>
  <head>
    <title> - - </title> </head>
```

Scanned by Scanner Go

```
<body>
<form action = "/webapp/store.jsp">
<input type = "hidden" name = "uname"
                       value = <%= uname %>
<input type = "hidden" name = "pwd"
                       value = <%= pwd %>
Father Name : <input type = "text" name = "fname">
Mother Name : <                    "mname">
<input type = "submit" value = "Register"/>
</form>
</body>
</html>
```

store.jsp

```
<%= request.getParameter("uname") %>
<%                       ("pwd") %>
<%                       ("fname") %>
<%                       ("mname") %>
```

Disadvantages of hidden variables :

n/w traffic will be increased because the data will be transfered between client & server for multiple times.

Advantage :

Engagements

Hidden Variables does not occupy much memory space.

Scanned by Scanner Go

March Monday 27th 2006

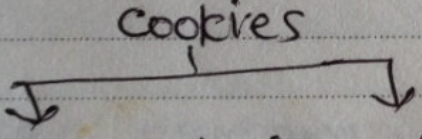
cookies:

These are also used to maintain the state of the application. cookies is a small piece of information set by the server on the client PC.

Every cookie contains 2 fields. They are name of the cookie and value of the cookie.

The server uses HttpResponse Format to set the cookies on the client program. The name of the Header cookies is SET-COOKIE. Life of the cookie is until we close the Brown

The cookies are stored inside the domain of Browser. Whenever client request for the domain next time, cookies are sent associated with the domain.



(default) Non-persistence persistence

Non-persistence cookies will be removed from the browser when we close the browser.

persistent cookies will not be removed from the browser until they expires.

Engagements

All the recent Browsers will accept max. of 15 cookies per domain.

Scanned by Scanner Go

March Wednesday 29th 2006

requirement: Develop a JSP that creates 2 cookies objects & send back to client.

```
<%@page import = "javax.servlet.http.*" %>
<%
    Cookie c1,c2;
    c1 = new Cookie ("username", "santosh");
    c2 = new Cookie ("password", "hello");
    response.addCookie(c1);
    response.addCookie(c2);
%>
```

Whenever we send a request to the server the browser checks for cookies to that particular domain, and they will be sent to the server as part of HTTP Request format. (as a part of Headers)

example:

Cookie: username = santosh
password = hello
JSessionID = J4h3x- - -

Whenever the server receives HttpRequest format the server checks for the cookies in header and it creates a Cookie Array. Now the server will gets the 1st Cookie details and create the Cookie object and added to Cookies array.

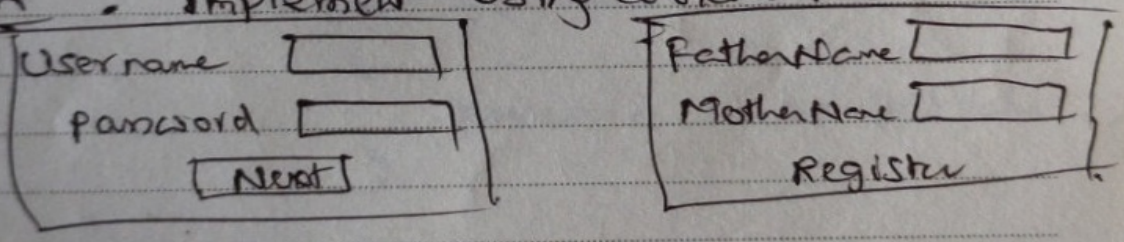
At last cookies array will be added to request object.

Engagements

Scanned by Scanner Go

```
<%
    Cookie c[] = request.getCookies();
    if (c != null)
    {
        for(int i=0; i<c.length; i++)
        {
            Cookie c1 = c[i];
            out.println( c1.getName());
                    "     (c1.getValue());
                    "     ("<br>");
        }
    else
        {
            out.print("cookies not available");
        }
%>
```

Requirement : implement using Cookies.



Need to develop 3 JSP programs :

Two JSP

```
<% String uname = request.getParameter("username");
                  ("pwd")
```

Engagements

```
Cookie c1 = new Cookie("uname", usrNm);
Cookie c2 = new Cookie("pwd", pwd);
response.addCookie(c1);
response.addCookie(c2);
```

Scanned by Scanner Go

three.jsp

```
<% String fname = request.getParameter("fname");
   mname =                              ("mname");
Cookie c[] = request.getCookies();
String uname = null;
Stri   pwd = null;

if (c != null)
{
   for(int i=0; i< c.length; i++)
   {
      Cookie ac = c[i];
      if (a[i] ac.getName().equals
                              ("uname"))
      {
         uname = ac.getValue();
      }
      if (ac.getName().equals ("pwd"))
      {
         password = ac.getValue();
      }
   }
   else
   {   uname = " ",  pwd = " "
   }
following details are entered by user
User Name   <%= uname %>
Password    <%= pwd %>
father Name <%= fname %>
Mother Name <%= mname %>
<a href="/webapp1/one.jsp"> New record </a>
```

Scanned by Scanner Go

Disadvantages of Cookies

(i) if the client does not accept Cookies the application will not behave as expected
(ii) Cookies are not used to store sensitive data of the project.

session:

1st time when a client send a request to the server on behalf of the client, the server will create one session object. and generate unique ID. and assign this ID to session object. Server will create 1 Cookie object JSessionID. and assigns the same unique ID. and sends this Cookie object to client.

To create session object:

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

To develop previous application second.jsp is written as

```
<% String un = request.getParameter("uid");
                                    ("pwd");
   session.setAttribute("uname", un);
                         ("pwd", pwd);
%>
```

code to display 2nd form

Pod Engagements

Scanned by Scanner Go

store.jsp
//

```
<%. String fn = request.getParameter("fname");
    mn =                          ("mname");
    Stri un = (String) session.getAttribute("un");
    pwd = (" ")                          ("pwd");
%>
```

URL Rewriting:

Disadv with Cookies & sessions

if Browser does not support cookies the application will not work as expected.

So By using URL Rewriting Technique we will rewrite all URLs with JsessionID. If we are using this technique even the client does n't accept the cookies, application will work.

In this, we will modify the URL and place inside response.encodeURL(-);

```
<form action = <%=response.encodeURL("/store.jsp")
                                              %>
</form>
```

Engagements

Scanned by Scanner Go

April Wednesday 5th 2006

JSP Action Tags:

Sun microsystems has released Jsp Action Tags to remove javacode from JSP's

Action Tags:

include, forward, useBean, setProperty, getProperty.

Syntax <jsp : include page="" />
- jsp : prefix
- include : Name of the Tag
- page : attribute.

include: (Dynamic)

we use this tag to include the calling jsp as well as called servlet content.

one.jsp

```
o/p from one.jsp <br>
< jsp:include page="two.jsp" />
```

forward: used to forward the request to any specific resource. Here only called jsp is included.

<jsp: forward page="two.jsp" />

include page directive (Static)

Engagements

used to include the contents of two.jsp in one.jsp:

<%@ include file="two.jsp" %>

Scanned by Scanner Go

April Friday 7th 2006

useBean :

This tag is used to create an object to any specific class.

<jsp : useBean id = "name" class = "java.lang.String" />

⇓

JSPC

⇓

String name = new String();

setProperty :

used to ~~set~~ call setter methods of java Bean.

<jsp : setProperty name = "p" property = "productId" value = "10" />

get property :

It is used to call getter methods of Java Beans

<jsp : getProperty name = "p" property = "productId" />

Engagements

Scanned by Scanner Go

April Sunday 9th 2006

Java Beans Introduction :

A program is called java Bean if it follows Java Beans Specification.

specification

(i) Every java Bean program must contain the default constructor.
(ii) It must be placed inside the package
(iii) It can contain properties / Events / Additional methods.

Ex

```
package ATAM.Addon;
public class Emp
{
    String name;
    public void setName(String name)
    { this.name = name; }
    public String getName()
    {
        return name;
    }
}
```

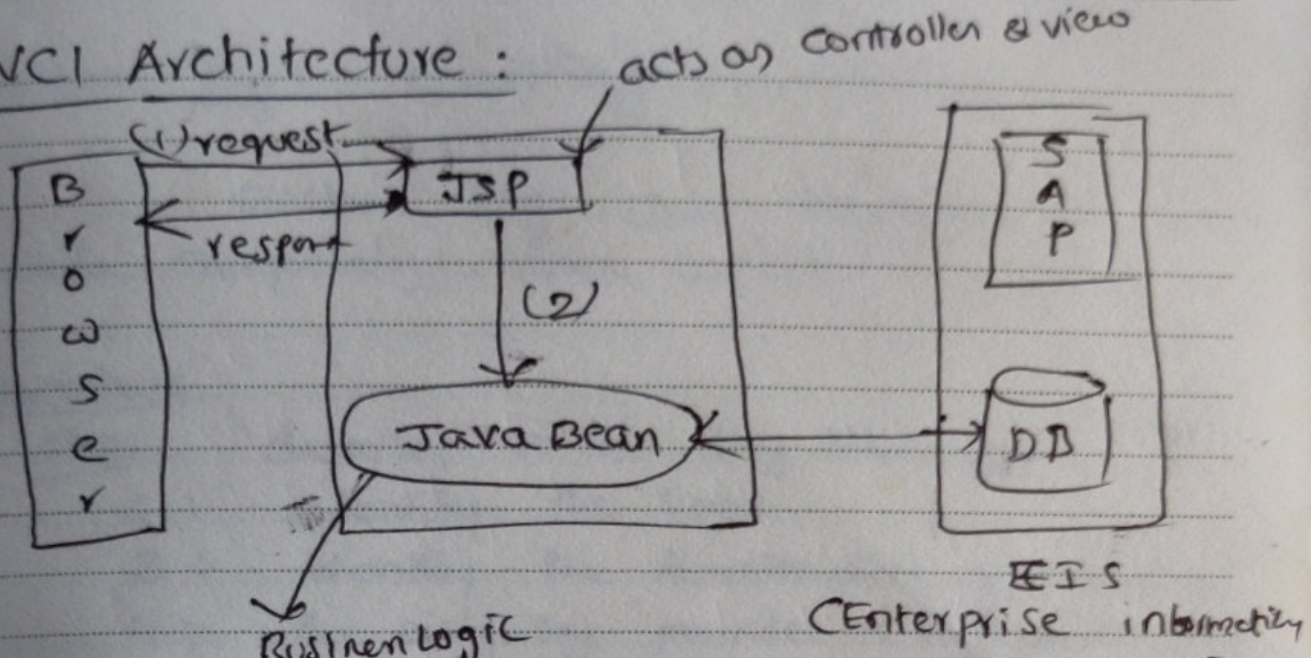
→ A property means either a setter / getter / both methods
→ An event can contain addListener(xxxListener) (or) removeListener(xxxListener)
→ Any other method is called 'additional methods.

Engagements

Scanned by Scanner Go

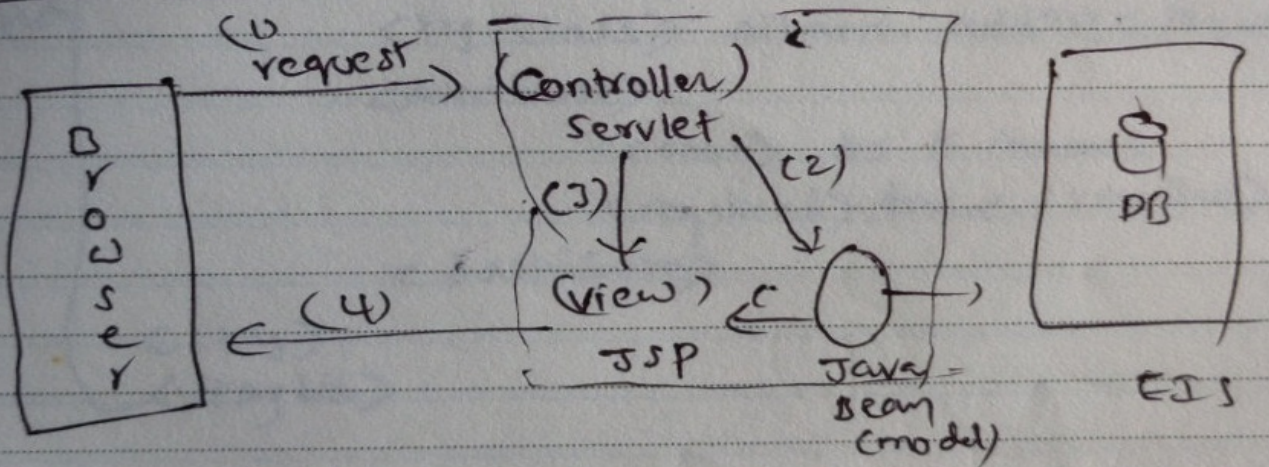
April Tuesday 11th 2006

MVC1 Architecture :



According to mvc1, JSP can act as view & controller.

MVC2 Architecture :



Engagements

Scanned by Scanner Go

April Thursday 13th 2006

custom Tags: (writing our own Tags)

Custom tags can remove the java code completely from JSP's.

procedure:

step1: develop it as tag library
2: Identify the tags
3: identify the functionality of every tag
4: for every tag provide a tagHandler class.
5. develop a TLD (Tag library descriptor)

```
<TagLib>
    <tag>
        <name> add </name>
        <tag-class> aitam.addTH </tag-class>
        <attribute>
            <name> AS </name>
            <required> true </required>
        </attribute>
    </tag>
</TagLib>
```

(or) th.tld

TagHandler class provides implementation of Tag interface directly (or) indirectly

javax.servlet.jsp.tagext.Tag

Engagements

↓

javax.servlet.jsp.tagext.TagSupport

Scanned by Scanner Go