Scanned by CamScanner

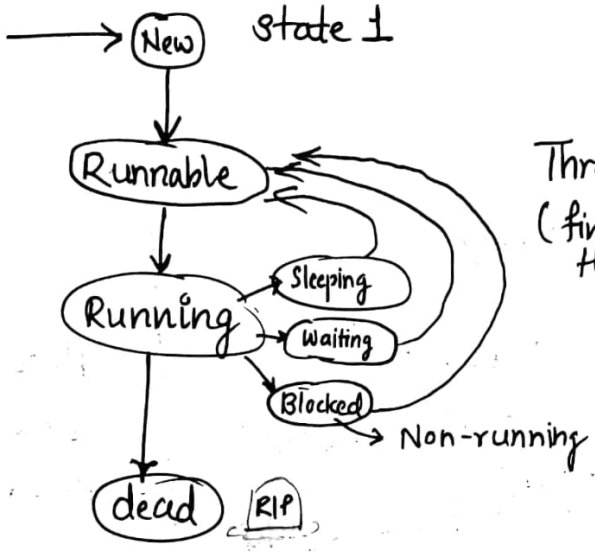
Thread

14-02-2018

Smallest part of program that gets executed independently.

By default the whole program is single threaded. It is called <u>main thread</u>.

* Life Cycle of thread



Thread.sleep(200)
↑
milliseconds.

Runnable interface : contains only one single method i.e. run() method.
We need to override the run() method.

We are 4 types of constructor in thread class.

ex: Thread t = new Thread(this, "xyz");

Scanned by CamScanner

15-02-18

```
class A implements Runnable
{
  public void run()
  {
    for(int i=0; i<10; i++)
    {
      Thread.sleep(30);
      sout(i);
    }
  }

class Test
{
  psvm(String args[])
  {
    A a1 = new A();
    Thread t1 = new Thread(a1);
    t1.start();
    Thread t2 = new Thread(new Runnable()
    {
      public void run()
      {
        for(int i=0; i<10; i++)
        {
        }
      }
    });
```

Thread();
Thread(String)
Thread(Runnable obj)

Other meth
Runnabl
Thread

Other method:

```
Runnable r1 = new Runnable() {
        public void run() {
            .....
        }
    };
Thread t2 = new Thread(r1);
```

OR

```
new Thread(r1).start();
```

OR

```
new Thread( new Runnable()
  {
    public void run()
    {
      .....
    }
  }).start();
```

Thread();
Thread(String)
Thread(Runnable obj)

Scanned by CamScanner

```
class A implements Runnable
{
    public void run()
    {
        for (int i=0; i<10; i++)
        {
            Thread.sleep(300);
            sout(i);
        }
    }
    void show()
    {
        . . .
    }

class Test
{
    psvm (string args[])
    { A a1 = new A()
      { public void run()
        {
            for(i=0; i<10; i++)
            { sout("Hi");
            }
        }
      };
    }
}
```

H.W : Create a Java Program : ~~containing~~ ~~two~~ th
i)- Unsynchronized, to perform writing & reading

where
file. wh
from t
ii) R
synchro

Scanned by CamScanner

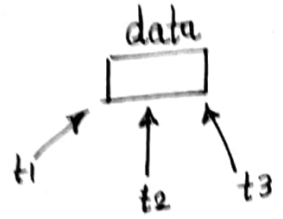
where first is trying to write a new data into the file. whereas second is trying to read, a data from the file

ii) Repeat the above program by providing synchronized behaviour.

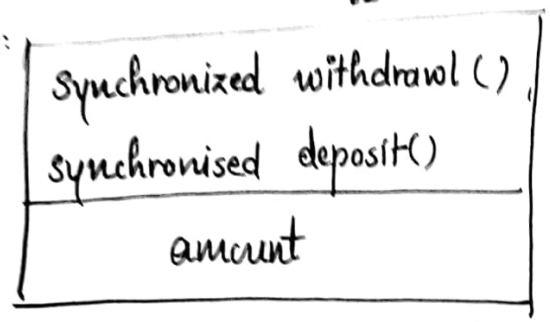
1)

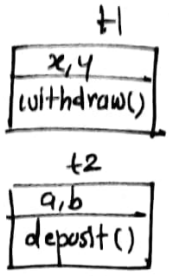
Scanned by CamScanner

19-02-2018



ex:

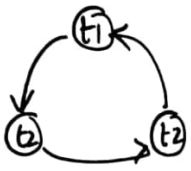




In some condition, t1 tries to access resource of t2 & t2 tries to access resource of t1. This is known as "Rounded wait" or "Circular wait".

Deadlock: No process can be executed.
(Thread based deadlock)

[Analogous to Process based deadlock in OS]



Priority

Thread Constructors:

```
Thread();
Thread(string)
Thread(Runnable, string)
```

}3

To get the name of the thread, use getName() method & to set use setName().

* By default the program is single threaded.

Scanned by CamScanner

Priority

1 ..... 5 10

(MIN_PRIORITY) (NORM_PRIORITY) (MAX_PRIORITY)

By default, every thread is having priority 5.

To set Priority:

use. thread_name.setPriority(1 to 10)

ex: t1.setPriority(6);
t2.setPriority(3);

t3.MIN_PRIORITY
(final & static)

To get Priority no:

use int x = thread_name.getPriority();

* If many threads are having same priority, thread scheduler will come into picture.

o join() method.

```
class A
{
  int sum;

  run()
  {
    //calculate sum;
  }
```

```
A al;

t1
t2   } threads
main
{
  t1.start();
  sout(al.sum);
}
```

← prints current value of t1 (even before t1 is finished) coz to thread scheduler.

to avoid this we can use the join method.

```
t1.start();
t1.join();
```

join()
join(millisecond)

Scanned by CamScanner

Database Connectivity

query → Language → RDBMS → SQL

Relational structure

ex: Student
- name
- Roll No.
- Age

Roll	Name	Age	Per
1			
2			
3			

Oracle → Oracle (DBMS software)
MySQL
MongoDB
DB2
Post Gress

MySQL

execute()
executeUpdate()
executeQuery()
} — statement class (static methods)

(Non-static) ← needs object of class to access

Driver Manager.

```
Connection con = DriverManager.getConnection(...);
Statement st = con.createStatement();
```

(Statement ← class)

Java — Oracle
Java — MySQL

jdbc drivers → group of classes

.jar files

```
url = "jdbc:mysql://localhost:3306
      //databaseName";
```

mysql > use database
mysql > show tables
mysql > use class
mysql > describe
mysql > create
> name
> age
> per
> prima
mysql > sele

Table
Default Ta
Add rows

import j
main() t
{
Class
Conne
interface
Stat
Re
while (
sout (
}
st.
con.

Scanned by CamScanner

mysql > use databasename ;

mysql > show tables

mysql > use cloud ;

mysql > describe log ;

mysql > create table student (rollno int not null ,
> name varchar(50),
> age int(12),
> per double(20),
> primary key (rollno));

mysql > select * from table_name.

{ MVN Repository }

{ MySQL Workbench }

1. import
2. load & register the driver
3. Create connection
4. create a statement
5. execute query
6. process result
7. close connec^n.

Table
Default Table mode
Add rows to Default Table model

DDL { create
DML { insert
DQL - fetch
TCL - permission

Roll	Name
1	X
2	Y
3	Z

```
import java.sql.*;
main() throws Exception
{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("URL",
                 "uname", "pswd");
Statement st. con.createStatement();
ResultSet rs = st.executeQuery("select * from
                Tablename");
while (rs.next())
sout (rs.getInt(1)+"  "+rs.getString(2));
}
st.close();
con.close();
```

returns object

{ static }

{ for insertion
st.executeUpdate(query); }

Scanned by CamScanner

```
Scanner sc;
Connection con;
ResultSet rs;
Statement stmt, st2;

Person()
{
    sc = new Scanner(System.in);
    try
    {
      Class.forName("com.mysql.jdbc.Driver");
      con = DriverManager.getConnection("jdbc:mysql://
          + "sample", "root", "pswd");

    }
    catch(ClassNotFoundException ex)
    {  System.out.println("class not found");
    }catch(SQLException ex)
    {
      printStackTrace();
    }
  try
  {
    stmt = con.createStatement();
    stmt.executeUpdate(sql);

  }
  catch(SQLException ex)
  {
    Logger.getLogger(Person.class.getName()).log(Level.
  }
}
```

Scanned by CamScanner

*

PreparedStatement pst = con.PreparedStatement(sql);

SQL query: insert into Person(name, age, gender) values(?, ?, ?)

pst.setString("Amol", 1);
pst.setInt(25, 2);
pst.setString("male", 3);

H.W Repeat the Person program with Prepared statement.

String query = "insert into student values(4, 'xyz')";

OR for var → value("+userid+", '"+username+"')";

PreparedStatement pst = con.preparedStatement(query);

→ values(?, ?)

pst.setInt(1, userid);
pst.setString(2, username);
int count = pst.executeUpdate();

No need to pass query here.

Class pqr
{
  static // Static block.
  {
  ...
  }
  { // Instance block.
  ...
  }
}

---

main() throws exception
{
  Class.forName("pqr");

(static block)

  Class.forName("pqr").newInstance();

(both blocks)

DriverManager.registerDriver(new com.mysql.jdbc.Driver());

≡ Class.forName("com.mysql---");

Scanned by CamScanner

24-02-2018

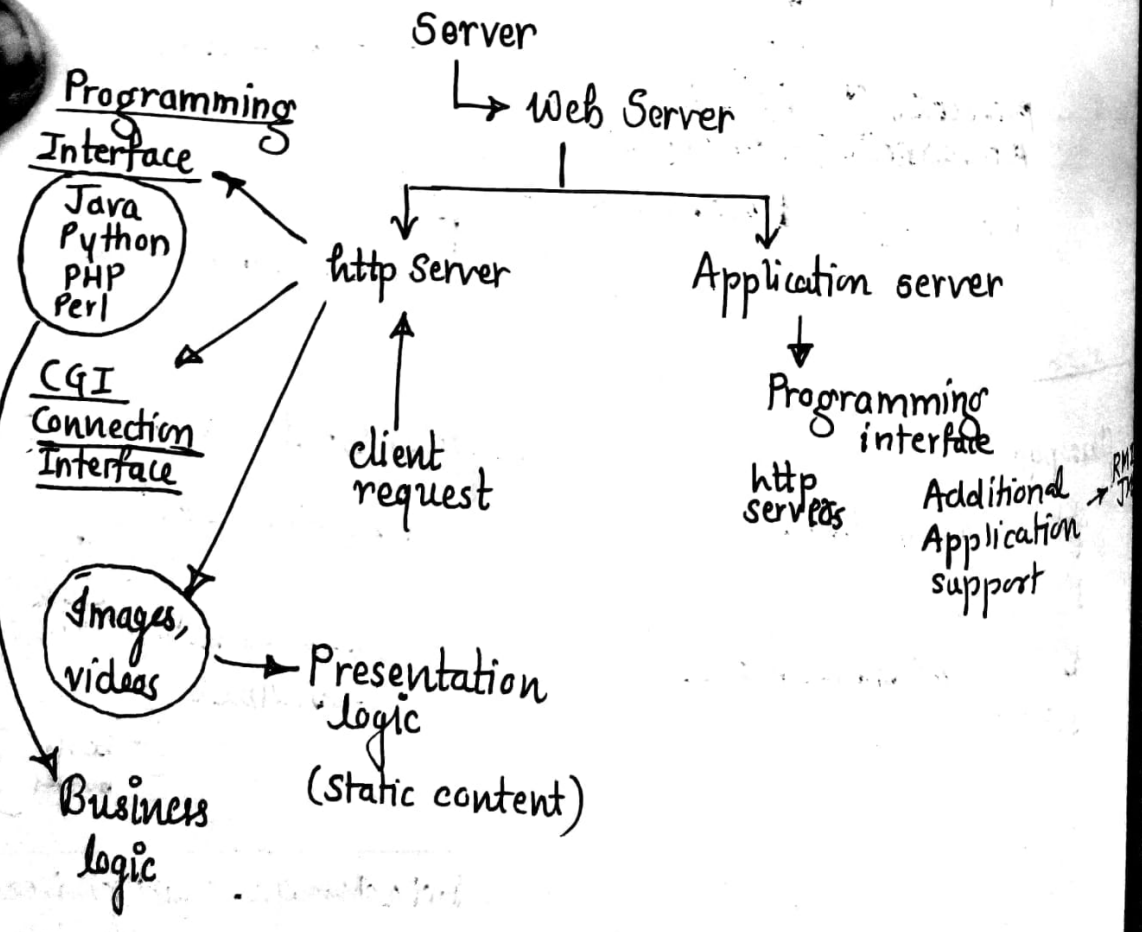
Java Web Application

glassfish 4.1.1/
glassfish/bin/asadmin.bat

Open glassfish server in C Driver.
Inside it search for folder glassfish.
Inside it search for bin folder.
Inside it search for file asadmin.bat.
Double-click on it to get server command window.
In the 'asadmin' command window, type the comm
start-domain domain1
default domain for glassfish serv

* Open any browser and enter your application details.

* To stop server write: stop-domain domain1



What is Serv
→ Object
receives
& respon
(html code
inside a

Important

* Servlets
mechanism
provides th
(presentati
* servlets a
* When se
processing
but the
* Servlet u
http sessio
* Writing
is optiona

H.W Read a
do Get
do Post
do Hea
do Put
do Del
do Opt
do Tra

Scanned by CamScanner

Application Server

- Glassfish
- Weblogic

What is servlet?

→ Object of servet class.

↓

receives the request & responses to request

(html code embedded inside a java code)

```
service (....)
{
    int x,y,z;
    z = x+y;
    sout(z);
    out.writeln("<html>");
    out.println("<h1>"+z+"</h1>");
}
```

Important points:

* Servlets are nothing but java objects which provides mechanism for processing the data as well as it provides the mechanism for representing (business logic) (presentation logic.)
* Servlets are generally subclass of http servlets class.
* When servlets are requested by the user, after processing of servlet, user can see only the contents but they cannot see the business logic.
* Servlet uses objects of servlet context, servlet config, http session, request dispatcher, print writer, etc.
* Writing alias name in URL pattern tag of web.xml is optional in JSP. but, it is mandatory in servlet.

H.W Read about the methods of http servlets class:

- doGet
- doPost
- doHead
- doPut
- doDelete
- doOptions
- doTrace

Life-cycle of servlet:

1. Servlet class is loaded
2. Servlet instance is created
3. init method is invoked
4. service method is invoked.
5. destroy method is invoked.

Scanned by CamScanner

28-02-2018

Servlet (Service)

→ Generic Servlet (empty configuration)

→ HttpServelet (HTTP Request)

→ DoGet() → processRequest

→ DoPost() → processRequest

No direct support for get() & post()

H.W : If the user enters a state name in text-field in index.html then your response should display list of cities in the state.

29-02-2018

```
String p = request.getParameter("state");
```

```
<select name="state" onchange="calls(this.value)">
    <option value="other">Other</option>
    <option value="mh">Maharashtra</option>
</select>

<script type="text/javascript">
   function call(v1){
      if(v1=="mh")
       document.location.href = "                    ";
      }

</script>
```

```
out.println("<body>");  →  out.println("<ul>");
  while(sc.hasNext())
  {
    out.println("<li>");
    "          (sc.nextLine());
    "          ("</li>");
  }
```

Scanned by CamScanner

```
        "    ("</ul>");
        "    ("<body>");
```

```
<div><a href = "Download File ? param = prideo">
  <h2><b> Video</b></h2></div>
        " Mp3 "          "param = pmp3 "
        "    "           "param = pimage"
        " Image
```

```
import org.apache.commons.io.FileUtils;
import org.apache.commons.io.IoUtils.

org.apache.commons.io.jar
```

```
response.setContentType("application/octet-stream");
String attr = request.getParameter("param");
File srcFile = null;
if (attr.equals("prideo")
{  response.setHeader("Content-Diposition",
"filename=\" " + fname1 + "\" ");
 srcFile = new File(path1 + fname1);
}
if (attr.equals("pmp3")){
   response.setHeader(" Content-Diposition",
                                  ");
                path2 + fname2

                path3 + fname3
```

Scanned by CamScanner

```
InputStream in = FileUtils.openInputStream(and
IOUtils.copy(in, response.getOutputStream());
```

Whenever we have to access init parameter, we use

```
ServletConfig config = getServletConfig();
int id = Integer.parseInt(config.getInitParameter
System.out.println("Value of Id", + id);
int localport = request.getLocalPort();
:
:
```

```
ServletContext context = getServletContext();
String myname = context.getInitParameter("myna
Sout("My Name" + myname);
```

To redirect a request to another resource.

```
response.sendRedirect("http://google.com");
response.sendRedirect(request.getContextPath()+"/inda
```

web.xml

<context-param>
<servlet
<init-param
</servlet>
<welcome-file-list>
<welcome-file>
:::

index

Nam
Age
ad
:
<inp
ty
na

for using forwa
class e Requ

RequestDis
rd.forwar

home.html

home /

authenti
you

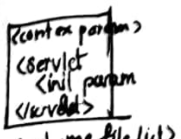
Sessio

1) hidden forr
2) URL rewritt
3) Http sessio
4) Cookies.

Scanned by CamScanner

05-March-2018

Web.xml index.html A.java B.java



Name
Age
action = "A"

<input
type = "text"
name = "name"

String name,

name = request.getParameter("name");

.....("age");

out.println("<html>");

("</html>");

forward method to send the same request to next servlet.

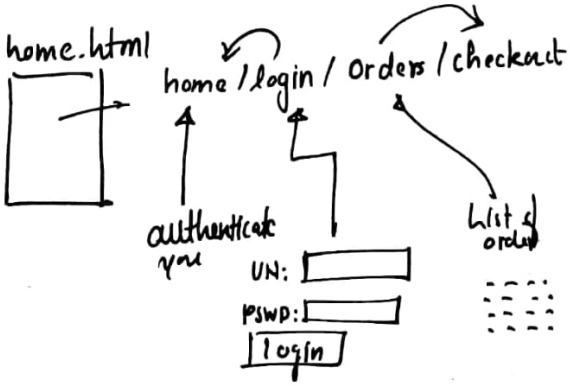
we can forward to one resource at a time.

response (optimal)

↑ when forwarding request, response can't be generated.

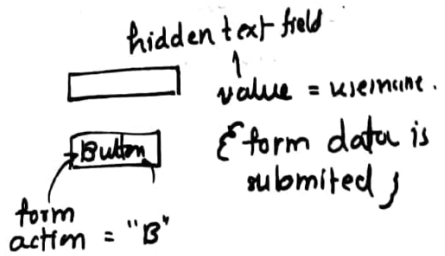
for using forward method, we need to create object of class ◆ RequestDispatcher.

```
RequestDispatcher rd = getRequestDispatcher("myhtml.html");
rd.forward(request, response);
```



Session Management

1) hidden form field.
2) URL rewritting
3) Http session
4) Cookies.



Scanned by CamScanner

 Home

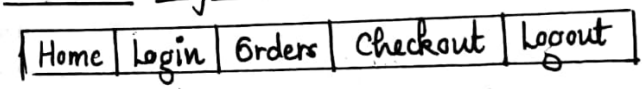
{Get request} internally myhtml.~~html~~jsp ?user = myuser ....

<% String user = request.setParameter ("user");
%>

myhtml.jsp

Whenever we use get method, the request is visible to the user.

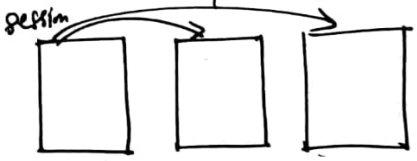
Session Object



bind parameters.

HttpSession session;

Session = request.getSession(); ← returns already created session object else it creates new object.



session.setAttribute ("user", str);

HttpSession session = request.getsession();

String username = (String) session.getAttribute (....)

H.W. Create a simple application with above mentioned tabs and manage the session using
i) Hidden form field.
ii) URL rewritting.

sout ("<h2> W
("<form
("
<inp
("</input>
("
<in

sout ("<h2>
+ userna
+ "<a
+ "<a
sout ("<h2>

session.se
session. se
session.ge
session =
session.

Scanned by CamScanner

```
sout ("<h2> Welcome" + id + "</h2>");
     ("<form action = \"ReceiveHidden\" method = \"post\">");
     ("<br><input type = \"hidden\" name = \"id\" value "+id">");
     ("</input>");
     ("<br><input type = \"submit\" value \"next\"
```

```
sout ("<h2><a href = \" MyHome Page?username = "
   + username + " \" > Home
   + "<a href = \"userlogin.html\">Login </a>"
   + "<a href = \"#\"> Order </h2><br><br>");
sout ("<h2> Welcome " + username + " to your Order </h2>");
```

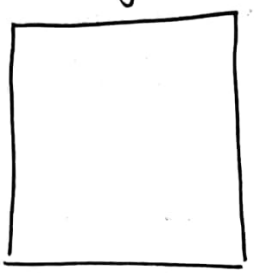
```
session.setAttribute ("user", idparam);
session. setMaxInactiveInterval (5*60);
session.getAttribute ("user");
session = requst.getsession();
session.invalidate(); ← logout.
```

Scanned by CamScanner

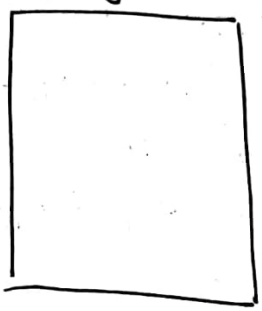
Session Management

08-03-20

A.java



B.java



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

getSession(false) ⟶ return existing session if it exits else nothing

```
session.setAttribute("key", "value");
```
(used)

```
string s = session.getAttribute("user");
```

setInterval(time in seconds)

Cookies :

```
Cookie cookie;
cookie = new Cookie("userid", "Saish");
response.addCookie(cookie);
```

* Array of cookies.

```
Cookie ck[];
ck = request.getCookies();
for(Cookie temp : ck)
{
    system.out.println(temp.getValue());
}
```

returns value ex: (sa

getName() ← retu

```
if(temp.
```

Server

creates a sessim id & a cookie object

session = new Cookie

response.ad

when cookie disabled in server uses url re approach.

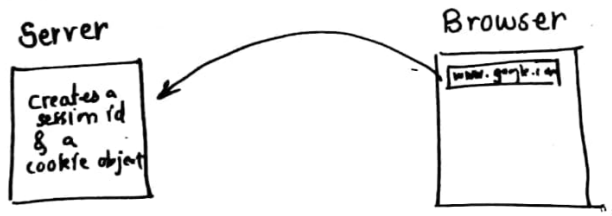
Servlet

Requst
respon
sessio
cookie
conte
confi

JSP (Jav

```
<html>
<head>.
<body>
  java
<%=
</bod
```

Scanned by CamScanner




```
session
= new Cookie("sessionid", id);
response.addCookie(session)
```

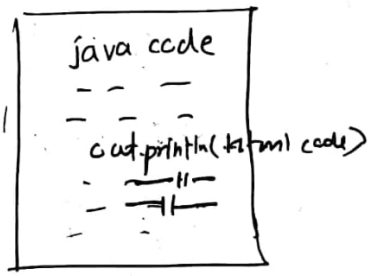
```
Cookie ck[] = request.getCookies();
// for(Cookie c : ck)
// {   // if many cookies are there.
// }
string value = ck[0].getValue();
string name = ck[0].getName();
```

When cookies are disabled in browser server uses:
url rewritten approach.

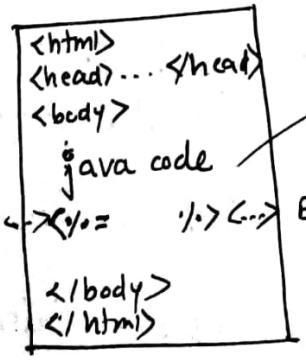
When url is like
www.google.com/----
browser sends same session id to server for verifica^n.

Servlet: objects

Request
response
session
cookies
context
config



JSP (Java Server Pages)



jsp

Expression tag can be used inside any tag

3 types of tags:
1) Expression tag
2) Scriptlet tag.
3) Declaration tag.

ex: <h2> Welcome <%= str or "Satish" %></h2>

Scanned by CamScanner

Scriptlet tag

```
<head>
<%
 String str = (String) request.getParameter("...");
%>
</head>

<h2> Welcome <%= str %></h2>
```

---

11 March 20...

9 implicit objects of JSP.

1) **out**: It is used to print the data over html similar to servelets.
   ex: out.println("Hello"); <% out.print...

2) **request**: It is similar to HttpServelet request object and performs all tasks similar to servlet request object.
   ex: <% String name = (String) request.getParameter...

3) **response**: Similar to servlet response object.

4) **Session**: This is same as Http session object used for session management.
   (This object is already created, we need not create it explicitly.)

5) **Exception**: Exception object is used for handling exceptions in the page

6) **config**: It is the same config object which we use in Servlet.
   This obj. is used to get the initial parameters if declared with the jsp

7) **page**: Ja... jsp pa...

8) **application**

9) **pageContext**

<%= ... %>
<% int ...
String na...
%>

<%! int x
public
{
}
%>

Scanned by CamScanner

```
<init-param>
<param-name>
<param-value>
</init-param>
```

ex: `int id = Integer.parseInt(config.getInit-Parameter("id"));`

7) **page**: To refer to the servlet instance of translated jsp page, this object is used.

8) **application**: It is used for getting and setting parameters for the whole application. This parameters can be accessed across all jsp pages in that application.

9) **pageContext**: This object is used for setting and getting the attributes according to the page, request, application and session's scope.

---

i) `<%= %>` → `out.println(x);` — Expression tag.

ii) Scriplet tag

```
<% int n1, n2, n3;
   String name = (String) request.getParameters("name");
   ....
%>
```

↑ Variables used here are accessible in expression tags.

`<%=n1%> <%=name%>.`

iii) Declaration tag — used for declaring methods var.

```
<%! int x, y;
    public int add(int x, int y)
    {
        return (x+y);
    }
%>
```

Scanned by CamScanner

Application object :

```
<html>
<body>
<%
    application.setAttribute("user","rohan");
    session.getAttribute("username","4th sem");
%>
```

```
<%
if(application.getAttribute("user").equals("rohan"))
{
    out.println("Welcome");
}
String username = session.getAttribute("username");
%>
```

```
session = pageContext.getSession();
pageContext.setAttribute("water","volume");
```

★ Four types of Scopes :
i) page
ii) request
iii) session
iv) application

→ PageContext.PAGE_SCOPE
or PageContext.REQUEST_SCOPE
— " — . SESSION_SCOPE

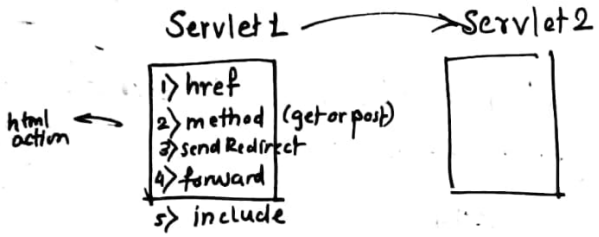
String username = session.getAttribute("username");
String ul = pageContext.getAttribute("user", PageContext.SESSION_SCOPE);

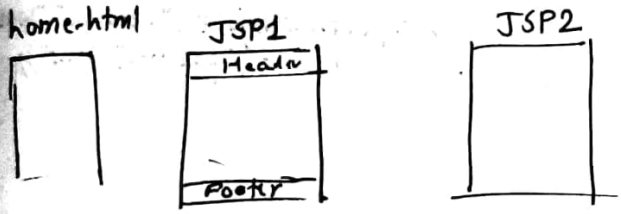
* pageContext.getAttribute("user1");

Directive Elements:

JSP has three directive elements:

1) Page directive
2) Include directive
3) taglib directive.



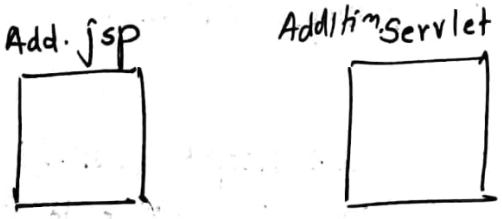


<%@ include file = "home.html" %>

Include directive.

Include directive is used to include the response of any resource in our current file.

<%@ include file = "Header.html" %>



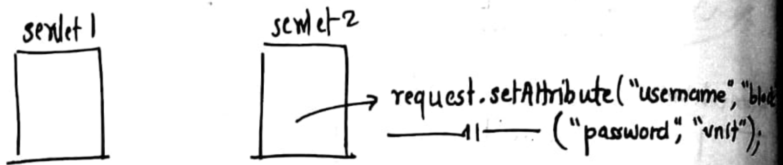
Scanned by CamScanner

```
<jsp: Include page = "/Addition" flush(true)" >
<jsp: param name= "num1" value="10"/>
<jsp: para name="num2" , what ... ?
```

```
<%@ include file = "sample2.jsp" %>
<jsp: include page = "/Addition flush="true"">
<jsp: param name = "num1"
```

15-03-2018

```
Request Dispatcher rd;
rd = request.getRequestDispatcher ("/Servelet2");
rd.include (request, response);
```



AJAX

AJAX is asynchronous javascript XML object used to communicate with server asynchronously.

__Creating AJAX object:__

```
<script>
    function getData()
    {
        Var request;   // simple variable.
        try
        {
            request = new XMLHttpRequest();
                        // assign ajax object to request
        }
```

```
catch
{ try {
    re
  }
  c
  {
  }
}
....
}
```

```
<script>
 function get
 {
   ....
   // Commu
   request.
   {
     if (
     {
       d
       =
     }
     reque
     reques
   } // end
</script>
```

* open() me
with server.
send() me
responseTex
from the
string or te

```
        catch (Exception e)
        {  try {
           request = new ActiveXObject("Microsoft.XMLHttp");
           }
           catch (Exception e2)
           {
              try {
                 ...
              }
              ...
           }
        ...
     }
```

```
<script7
   function getData()
   {
      ...
      // Communicate with server
      request.onreadystatechage = function()
      {
         if (request.readystate==4 && request.status==200)
         {
            document.getElementById("myid").innerHTML
            = request.responseText;
         }
      request.open("Get", "sample.jsp", true);
      request.send();
   } // end of getData

</script>
```

* open() method will open the connection with server.
  send() method sends a request to the server.
  responseText() is used for getting the response from the requested resource in the form of string or text.

Scanned by CamScanner

* onreadystatechange

A function will be stored and will be called automatically each time ready state property changes.

Ready state values:

i) 0 - request not initialized
ii) 1 - connection between server & request established.
iii) 2 - request is received on the server.
iv) 3 - request is processing.
v) 4 - request finished processing & response is ready.

Status:

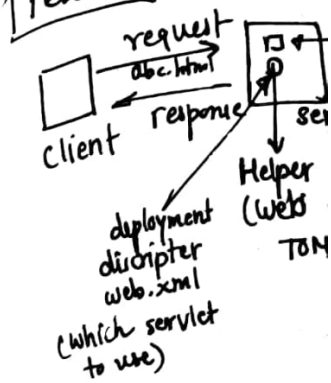
i) 200 - OK
ii) 404 - Page Not Found.

---

* To getting current time and date use,

java.util.Calender.getInstance().getTime();

Telusko : Servlets

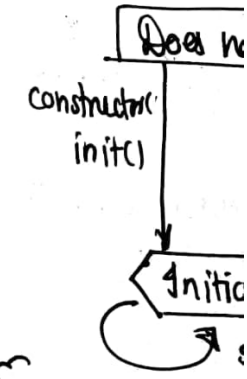


HTTP Request

Key element of a "request" stream

- HTTP method (action to be performed)
- The page to access (a URL)
- Form parameter

Servlet Life

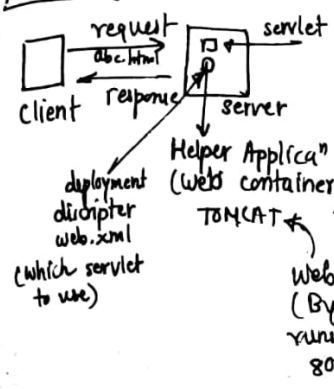


init is called only once if t servlet is alread

Scanned by CamScanner

Telusko : Servlets

annotations → to avoid xml on servlets



To make servlet class, extend HttpServlet class

ex:

```
public class Add extends
 HttpServlet(throws Exception
{ // Declaration section
  public void service
  (HttpServletRequest req,
   HttpServletResponse res)
  {
    int i =
Integer.ParseInt(req.getParameter("num1"));
    int j =
  ————————————("num2");
    int k = i+j;
*   PrintWriter out =
        res.getWriter();
    out.println(k);
  }
}
```

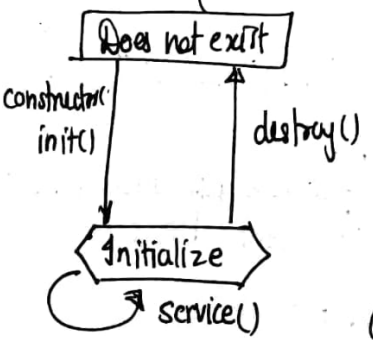
HTTP Request	HTTP Responce
Key element of a "request" stream	Key element of a "response" stream
➤ HTTP method (action to be performed)	➤ A status code (for whether the request was successful)
➤ The page to access (a URL)	➤ content type (text, pic, html, etc)
➤ Form parameter	➤ The content (the actual content).

Instead of service, we can use doGet, doPost.

OR

in defn of doPost & doGet call processRequest method.

Servlet Lifecycle



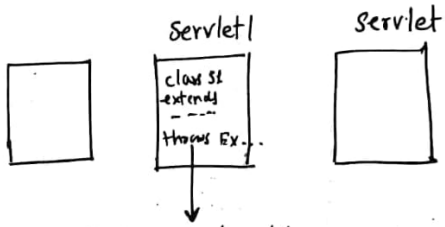
init is called only once if the servlet is already in container :)

Hierarchy

(Interface)	Servlet	
(Abstract class)	Generic servlet	- gt not overridden, implements init() method from the servlet interface.
(Abstract class)	Http servlet	gt is not overridden, implements service method()
(concrete class)	Your servlet	- We implement Http methods here

Scanned by CamScanner

If get & post are not mentioned, it is by default 'get'.



Request dispatcher
rd = request.getRequestDispatcher("servlet2");
rd.forward(request, response);
} shows first servlet in url bar

OR

response.sendRedirect("SecondServlet"); } shows second servlet in url bar

request from calling servlet not redirected

* Session management:

HttpSession session = request.getSession();

Interface

session.setAttribute("name1", "namedata")

Now, in servlet 2:

HttpSession session = request.getSession();

String str = session.getAttribute("name1").toString();

out.println(str);

* Cookie

Servlet1 {

Cookie cookie = new Cookie("name1", namedata)

response.addCookie(cookie);

↑ cookie gets added in the client's browser.

response.sendRedirect("secondServlet");

}

{

Cookie cookies[] = request.getCookies();

for(Cookie c : cookies)

{

if(c.getName().equals("name1"))

{

str = c.getValue();

}

}

URL rewritting (for

response.send

* Servlet → JSP

	JSP
Directive tag	<%@ import= %>
Declarative tag	<%! %>
Scriplet tag	<% %>
Expression tag	<%= %>

* login using servle

Scanned by CamScanner

URL rewritting (for get method)

response.sendRedirect("SecondServlet?name1="+str);

rewritting the url.

* Servlet → JSP

	JSP		Servlet
Directive tag	<%@ import="..." %>		
Declarative tag	<%! %>	→	class variables, methods.
Scriptlet tag	<% %>	→	service method.
Expression tag	<%= %>	→	out.println(..);

* login using servlet & JSP.

Scanned by CamScanner

End Sem

Page directive: It is used to provide many facilities like import, exception handling, etc. It supports 13 Attributes which are as follows:

1) buffer
2) autoflush
3) contentType
4) errorPage
5) isErrorPage
6) extends
7) import
8) info
9) isThreadSafe
10) session
11) isELIgnored
12) language ❤
13) trimDirectiveWhiteSpaces.

Exception handling mechanism in JSP
(At compile time)

```
<!DOCTYPE html>
<%@page isErrorPage="true" %>
<html>
    <body> Error Occured </body>
</html>
```

<%@ page content ty
<%@ page errorP
<%@ page isEr
<error-page>
<exception-t
<location>
</error-page>
(In exception
type as well a
Exception han
If you
error page th
through web.
using web.xm
web.xml
error page to
exception ty
the exception
We c
error-code,
type.

Scanned by CamScanner

<%@ page contentType = "text/html", pageEn . . . . >

<%@ page errorPage = "ErrorMsg.jsp" %>

<%@ page isErrorPage = "true" %>

```
<error-page>
    <exception-type> name of exception </exception-type>
    <location> /ErrorJsp.jsp </location>
</error-page>
```

(In exception tag you can provide exception type as well as error code)

Exception handling using web.xml

If you don't specify any page as an error page then also we can handle the exception through web.xml file. To handle the exception using web.xml we can use <error-page> tag in web.xml after inside <web-app> tag. The error page tag has two child tags which are exception type and location if we are handling the exception by using exception type.

We can also handle exception by using error-code, which will be in place of exception type.

Scanned by CamScanner

example:

```
<web.xml>
 <web-app>
   <error-page>
     <exception-type> java.io.FileNotFoundExcepti
     </exception-type>
     <location> /ErrorJsp-jsp </location>
   </error-page>
   <error-page>
     <exception-type> java.lang.Exception</exceptionty
     <location>/mylogin.html</location>
   </error-page>
 </web-app>
</web.xml>
```

H.W Determine whether exception handling approach uses send-redirect approach or forward approach internally.
Uncover

Java Beans

(getters, s
. for
instance

<jsp
scope = "req

Java bea
following con
i) It sho
should
ii) It sh
iii) It sh
values
known
The
reusabil

ex: public c
{

Scanned by CamScanner

Java Beans

: Bean is java class. It should have default arg. constructor.

(getters, setters) ← Bean class.
. for each instance variable

```
<jsp::useBean id="b2" class="beans.Employee"
scope="request"/>
```

05-April-2018

Java bean is a java class that should follow following conventions.

i) It should have a no argument constructor or should not have any constructor.

ii) It should be serializable.

iii) It should provide methods to set and get the values of the properties, these methods are also known as getter and setter methods.

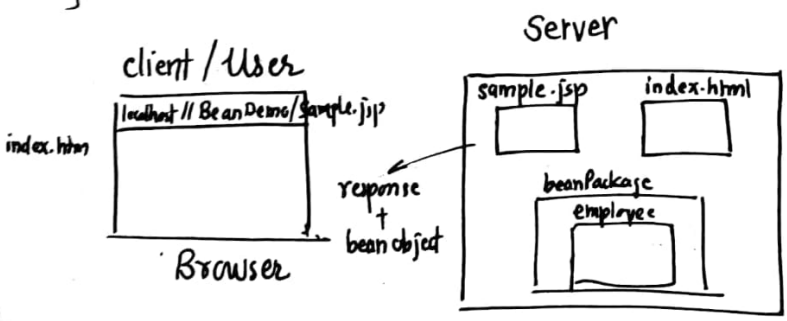
The main purpose of using java bean is the reusability and dependence free environment

ex:

```
public class Employee
{
    string name;
    int empid;
    public void setName (String n)
    {
        name = n;
    }
    public void setEmpid (int id)
    {
        empid = id;
    }
```

Scanned by CamScanner

```
public String getName()
{
    return name;
}
public int getEmpid()
{
    return empid;
}
}
```



Bean demo

<h1> Hello </h1>
response.

* action tags for using jsp bean

i) <jsp: useBin id = "objectName" class = "class Name with package name" type = "package name. class Name" />
scope = "page/request/session/application"

ex: <jsp: useBin id = "emp1" class = "beanex. Employee" type = "beanex. Employee" >

<jsp: useBin > tag is used to create bean object. Here id attribute specifies the new object name whereas the class attribute specifies the class name. Whenever we have to set or get the value of bean properties we need to create a bean object using useBin tag. Scope indicates the scope of newly created object.

2) <jsp : setProp
property = "
value = "

ex: <jsp : setPr
property =

ex: <jsp : set Pro
(for settin

3) <jsp : getP

Although multiple tag can values.

* getter, setter

Scanned by CamScanner

2) <jsp : setProperty name = "object name",
property = "property name" param = "parameter.name"
value = "property value" />

ex:
<jsp : setProperty name = "emp1"
property = "name" value = "Amit" />

ex:
<jsp : setProperty name = "emp1" property = "*" />
(for setting values to all properties with param)

3) <jsp : getProperty name = "object name" />

Although setProperty can be used to set multiple property values, but get getProperty tag can be used to get only one single values.

* getter, setter need get_Method_Name, set_Method_Name.

Scanned by CamScanner

Expression Language

<h3> Count = ${t1.giveCount()} </h3>

* Expression language makes it possible to access application data stored in java beans component. Expression language is also used to access several implicit objects.

Following are expression language implicit objects

i) pageScope - It is a map that contains the attribute set which are defined within the pageScope

ii) requestScope - Use to get the attribute values which are defined within the request scope.

iii) sessionScope - use to get the session values which are ------ 〃 ------

iv) applicationScope - use to get the application 〃 〃

v) param - use to get request parameter value. It returns a single value.

vi) paramValues - use to get the request param values in an array

vii) header - used to get request header information.

viii) headerValues - used to get headerValues array

ix) cookie - used to

x) initParam - used parame servlet

xi) pageContext - context

Expression langu

i) dot operator (.): ge

ex: ${t1.

ii) collection acce used to array.

iii) EL Arithmetic are provi expressions.

ex: ${

iv) EL Logical EL as

v) EL relation "!=" ,

vi) EL reserv le, ge

Scanned by CamScanner

ix) Cookie - used to get the cookie value in jsp.

x) initParam - used to get the context init parameters (we can't use it for servlet init parameters)

xi) pageContext - This is similar to jsp page context object.

Expression language operator.

i) dot operator (.): In EL dot operator is used to get the value of ~~bean~~ properties.

ex: ${t1.a}

ii) collection access operator "[]": This operator is used to get the data from list or array.

iii) EL Arithmetic Operators : EL arithmetic operators are provided for simple calculations in EL expressions.

ex: ${5+5}

iv) EL Logical operators: &&, ||, ! are used in EL as logical operators.

v) EL relational operators: They are equal to "==" "!=", "<", ">", "<=", ">="

vi) EL reserved words : and, or, not, eq, lt, gt, le, ge, false, null, instanceof, empty.

Scanned by CamScanner

* Points to Remember

1. EL expressions are always within curly { } braces prefix with $

   ${ expression }

2. We can disable EL by setting JSP page directive attribute isELIgnored = 'true'

3. EL implicit objects are different than JSP implicit objects except page context.

4. Expression language is null friendly and hence it doesn't throw any exception for null values

---

```
<jsp:UseBin . . . .
<jsp:setProperty . . . .
<style>
   div
   {
      color: <jsp:getProperty name="tl"
     property = "colors" />
      font-size : 20px;
   }
</style>
```

Alternative code

```
   div
   {
      color: ${tl.colors} ;
      font-size : 20px;
   }
```

(it must in bean)

Uncovering:

A web
i) ind
ii) co
iii) st
iv) ci

when user cl
it should be direct
the country.js
countries. As soon
user should be r
same operatio
page. use bean cl
appropriate valu

Scanned by CamScanner

Uncovering: A website has four jsp pages

i) index.jsp
ii) country.jsp
iii) states.jsp
iv) cities.jsp

when user clicks on hyperlink in index.jsp it should be directed to country.jsp page.

The country.jsp page provides a list of countries. As soon as user select any country, user should be redirected to states.jsp page.

same operation should be performed on states.jsp page.

use bean classes to manage the list and appropriate values.

Scanned by CamScanner

Custom tags.

Custom tags are user defined tags. They eliminate the possibility of scriplet tag and separate the business logic from the JSP page.

Advantages of Custom Tags:

1) Reusability
2) Elimination of scriplet tag
3) Separation of business logic from jsp.

Steps to create custom tags:

1) Create a Tag library Discriptor (tld) file. inside web-inf folder.
2) Create tag handler class file. (It is a java file)
3) Use taglib directory to define prefix.
4) Use created custom tag inside jsp.

ProjectName → new → tag library discriptor

ProjectName → new → tag handler

Extend simple tag support. (file creation + package)

Browse tld file from WEB-INF folder

In body content choose empty.

Click on finish

---

tld file

```
<tag>
    <name> xyz </name>
    <tag-class> fact.FactHandler </tag-class>
    <body-content> empty </body-content>
<tag>
```

```
<?xml version
<taglib version

<tlib-versio
<short-nam
<url> /
<tag>
    <nam
    <tag-
    <body
</tag>
</taglib>.

public class

{
    @overri
    public
        Js
    try{
    /* Js
      if (
        f
      }
    */
    out.pr

}
catch (
{ . . . .
}
```

Scanned by CamScanner

.tld file

```
<?xml version="1.0" . . . . >
<taglib version - - - - -
    <tlib-version> 1.0 </tlib-version>
    <short-name> customtag </short-name>
    <uri> /WEB-INF/tlds/CustomTag1 </uri>
    <tag>
        <name> xyz </name>
        <tag-class> fact. FactHandler </tag-class>
        <body-content> empty </body-content>
    </tag>
</taglib>.
```

FactHandler.java

```
public class FactHandler extends SimpleTagSupport
{
    @Override
    public void doTag() throws JspException{
        JspWriter out = getJspContext().getOut();

     try{
     /*JspFragment f = getJspBody()
      if (f!=null){
         f.invoke(out);
      }
    */
     out.print("Rohan");

}
catch(
{. . . .
}
```

Scanned by CamScanner

jsp page

```
<%@ page contentType = "text/html",
         pageEncoding = "UTF-8" %>
<%@ taglib url = "/WEB-INF/tlds/CustomTag1.tld"
         prefix = "RS" %>

<!DOCTYPE HTML>
   . . . .
   . . . .
   . . . .
<h1> <RS:Rohan a="20" /> </h1>
```

prefix, tag name, attribute.

```
<tag> . . . .
   <attribute>
      <name> a </name>          //(name should be same as defined in tag handler class)
      <required> true </required>
      <rtexpvalue> true </rtexpvalue>
   </attribute>
```

```
int i=1, fact=1;
for(i=1 ; i<=a ; i++)
{
   fact = fact * i ;
}
out.println ( a + "<h1>. . . . </h1>");
```

Uncovering: Create Custom tags for all seven Rainbow colours

MyTag.tld

```
<tag>
<name> MyFont </name>
<class-name> packa
<body-content> e
</tag>
```

MyDemo-jsp

```
<%@ taglib
<jsp : MyF
```

TagHandler.j

```
int num
doTag (
out.println (
out.print ("<span st
```

```
<attribute
<r
<r
</attribu
```

ColourHan

```
String st
```

```
MyDem
<%@ taglib
<jsp : My
Welcor
</jsp:
```

Tag Body OR Body Content

Scanned by CamScanner

MyTag.tld

```
<tag>
<name> MyFont </name>
<class-name> package.classname </class-name>
<body-content> empty / scriptless </body-content>
</tag>
```

can declare attributes here

MyDemo.jsp

```
<%@ taglib uri = "web/webinf / Mytag . tld " prefix = " jsp " %>
<jsp : MyFont />
```

TagHandler.java

```
int num;
doTag ( )
out.println ("Hi");
out.print ("<span style=font size=num>Hi </span>");
```

If there are attributes, there should be setter methods for each.

```
<attribute>
    <name> num </name>
    <required> true <required>
</attribute>
```

ColourHandler.java

```
String str, color
```

MyTag . tld

```
<tag>
  <name> MyColor </name>
  ...
</tag>
```

MyDemo.jsp

```
<%@ taglib uri = "..tld" prefix="jsp" %>
<jsp : MyColor str = "Hello" color = "red" >
    Welcome
</jsp : MyColor>
```

Tag Body
OR
Body Content

Scanned by CamScanner

StringWriter s = new StringWriter();

getJspBody().invoke(s);

s ← retrieve the content of s in tag handler
& store in variable

return String Buffer object to
out.print("<h1>" + s + "</h1>");

out.print("<h3>"

<awp:db user

<awp: MyColor

<awp: db usern
password=

<awp:db use
pas
(if it is se

Database Handler

<jsp:db username = "user1", password = "mysql" />

DBHandler

String username, password;

// setter method

...

doTag(...)

{

Class.forName("...");

Connection con = DriverManager.getConnection(...

Statement st = con.createStatement();

String str = "select age, city from person
where username = username and
password = password";

Scanned by CamScanner

```
out.print("<h3>" + rs.getString("city") + "</h3>");
   ——11——      rs.getInt("large") +
```

```
<awp:db   username = "... " , . . . . />
              ↑
            ${ . . . }

<awp: Mycolor   str = "${ . . . }"
                          ↑
                       param.str

<awp:dB   username = "${param.username}" ,
          password = "${param.password}" />

<awp:db   username = "${ username }" ,
          password = "${ password }"  >
```

(if it is session.setAttribute("username", username);
——11—— ("password", password);

Scanned by CamScanner

JSTL (JSP Standard Tag Library)

The JSP standard Tag Library represents a set of tags to simplify the JSP development.

Advantage:

1. Fast Development:
2. Code Reusability:
3. No need to use scriptlet tag:

```
<h3><c:out value="Hi Everyone" /></h3>
<c:out import url="http://www.google.com" />
<c:set var="x" value="12th April" />
<c:out value="x" />          ← prints x
<— " —  value="${x}" />      ← prints 12th April.
<c:remove scope="session" var="x" />
<h3> <c:out value="${x}" /></h3>   ← doesn't get printed.
```

```
<c:catch var="obj">
    <%= 1/0 %>
</c:catch>
```
} to catch exception

```
<c:out value="${obj}" />
```

Tag

```
<body>
  <c:if var
    (x will
      for
    ${x}
  </c:if>
```

The part written will get executed true.

```
<c:if
  <h
</c:if
```

```
<c:if
  =
</c:if
```

```
id="st
<c:if
  <
</c:if
```

Scanned by CamScanner

#last_class

TagDemo.jsp

```
<body>
  <c:if var="x" test="expression">
    (x will get the value of expression
      for ex: ${5<10} returns true)
    ${x} ← displays true.
  </c:if>
```

The part written within these ~~for~~ tags will get executed if the condition is true.

```
<c:if test="${x}">
    <h3> Welcome </h3>
</c:if>
```

```
<c:if test="${param.num} > 100">
</c:if>
```

```
id='st'
<c:if test="${s1.per} = 75.0}>
    <h2> Distinction </h2>
</c:if>
```

P.T.O

Scanned by CamScanner

```
<c:choose>
    <c:when test="....">
    ==========
    </c:when>
    <c:when test="....">
    ==========
    </c:when>
    <c:otherwise>
    ==========
    </c:otherwise>
</c:choose>
```

executes first when statement

```
<c:forEach begin="1" end="5">
    <c:out value="Hi"/>
    <h3> hi </h3>
</c:forEach>
```

```
<c:forEach begin="1" end="10" step="2">
    <c:out value="Hello"/>
</c:forEach>
```

```
<c:forEach item="${cl.colorlist}" var="value">
    <h3> ${value} </h3>
</c:forEach>
```

ArrayList object

```
<c:forEach item
    var="value"
    <h3> ${valu
</c:forEach>
```

```
<jsp:usebean
    setDept (Stri
    class. forNa
    string sql =
```

==========

```
while ( rs.
{
    string
    ename
}
```

```
${emp.set
<ul>
<c:forEach
    <li> ${
</c:forEach>
</ul>
```

in tld → ta
or attribute so
here.

Scanned by CamScanner

```
<c:forEach items="${c1.colorlist}"
   var="value" begin="0" end="9">
   <h3> ${value} </h3>
</c:forEach>
```

executes first when started

```
<jsp:usebean id="emp" ..... >
```

```
setDept(String dept)
  class.forName(....)
  String sql = "SELECT emp_name from
               Employee WHERE department='"
               + dept + "'";

  while(rs.next())
  {
     String str = rs.getString("empname");
     ename.add(str);
  }
```

```
${emp.setDept("CSE")}
<ul>
<c:forEach var="name" item="${tempname}">
   <li> ${name} </li>
</c:forEach>
</ul>
```

in tld → tag → we cannot set any property or attribute so bean is preferred over here.

Arraylist object

Scanned by CamScanner

<c:forTokens>

begin
end
items
step
var
delims

```
<c:set var="str"
    value="Learning_Languages_is_fun">

<c:forTokens items="${str}" var="value"
    delims="_">
    <h3> ${value} </h3>
</c:forTokens>
```

We can set any variable value in <c:set... >

<c:param>

```
<c:url = "JSTLDemo2.jsp" var="home" scope="session">
    <c:param name="num1"
        value="20" >
    <c:param name="num2"
        value="30" >
</c:url>
```

(We use <c:if> tag to check the condition)

```
<c:redirect url = "http://www.google.com" />
```

We can redirect to page not in context

JSTL tags
- Core tags
- function tags

JS

endsWith()
startsWith()
indexOf()
lastIndexO
length()
substring()

To use the
we have the

fn : endsWi
fn : startsW
fn : indexOf
fn : lastIn
fn : length
fn : subStri

ex: ${ f

Scanned by CamScanner

JSTL function tags

endsWith()
startsWith()
indexOf()
lastIndexOf()
length()
subString()

} these String func^{n} can be used in scriplet tags.

To use the above function ~~in~~ JSTL we have the following tags.

fn : endsWith
fn : startsWith
fn : indexOf
fn : lastIndexOf
fn : length
fn : subString

} we have to use it with expression language.

${ fn: ---

ex: ${ fn: endsWith ("KKR team", "team");

"KKR team" → input string
"team" → ~~prefix~~ suffix