SERVER SIDE PROGRAMMING

JAVA SERVLET

Client Side Scripting

1. Scripts run /executed on the clients m/c or browser
2. can't used to connect DB
3. doesn't provide security for data
4. Response is faster
5. Javascript, Jscript, VB Script

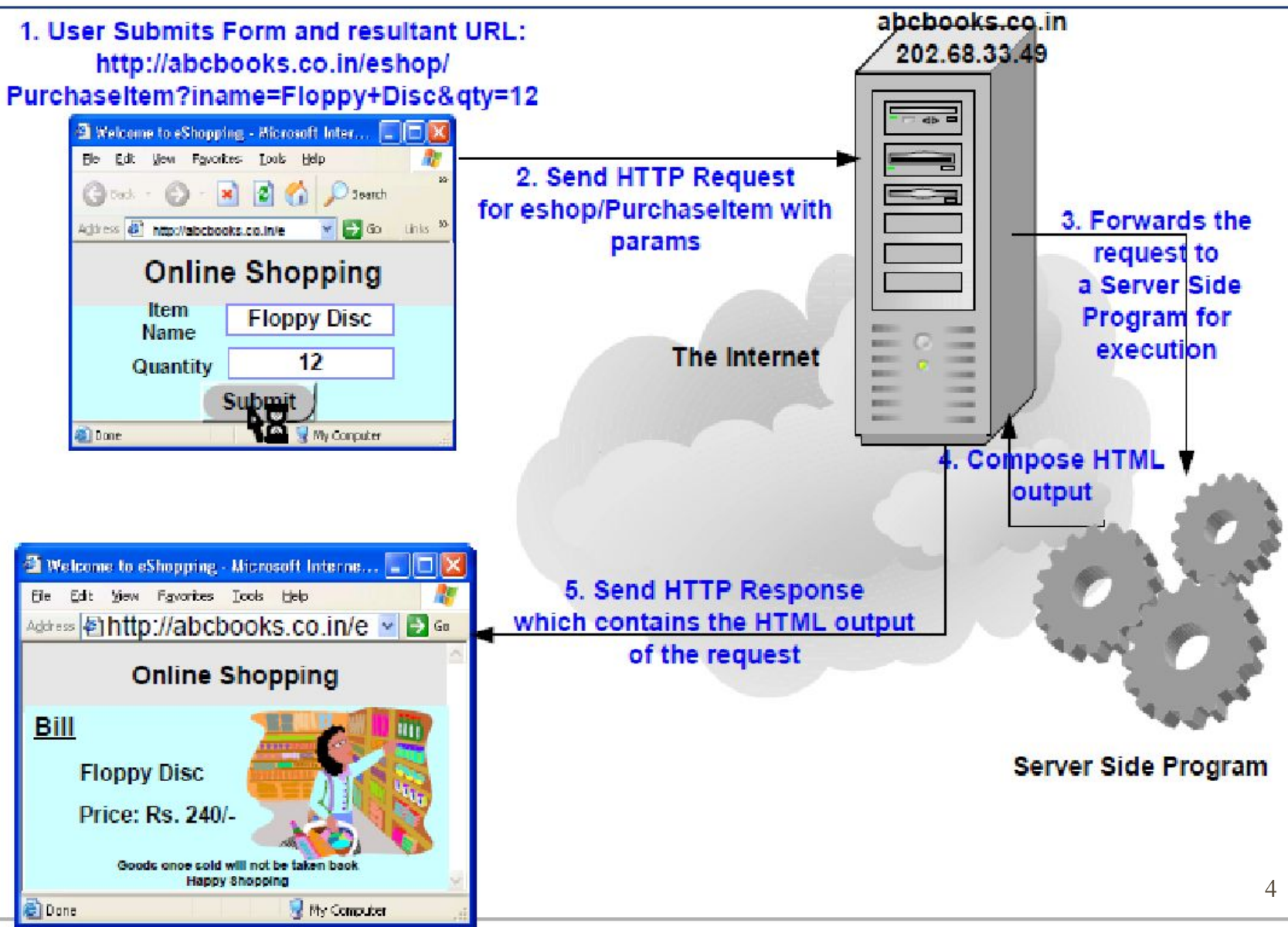
Server Side Scripting

1. Scripts runs on webserver to produce response that is customized for each user request to the website
2. Executed in the back end or web server
3. provides security for data
4. Response is Slower
5. PHP, Servlet, JSP, ASP.Net, Perl, Ruby etc

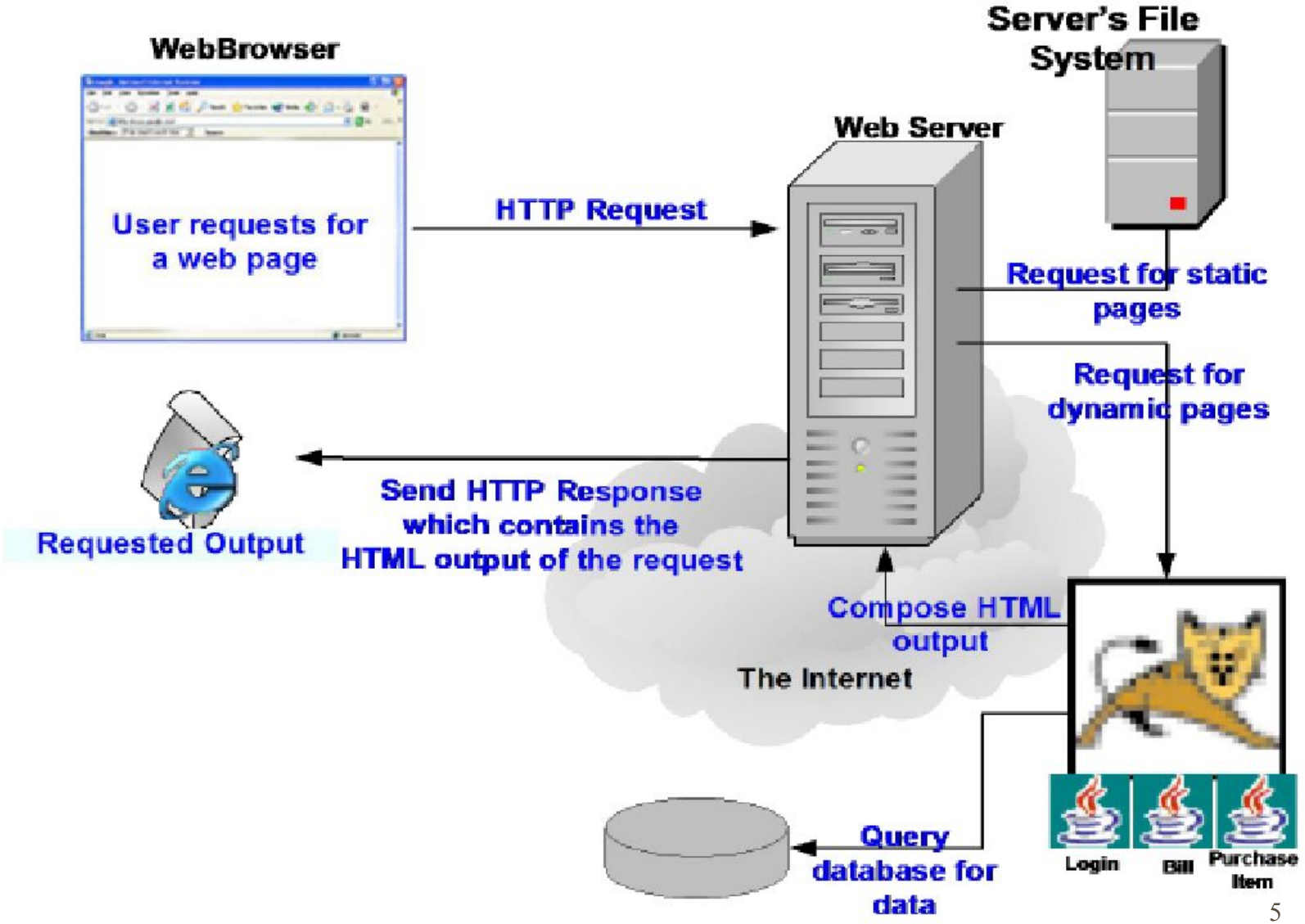
SERVLET

- Servlets are **java codes** that run on the **web server** to generate **dynamic contents**.
- collect **input from** users through web page **forms**, present records from a database or another source, and create web pages dynamically.
- The common protocol used for Servlet operation is HTTP.
- Servlet was introduced by Sun Microsystems as an effective alternative for CGI programs
- The output of the Servlet can be HTML

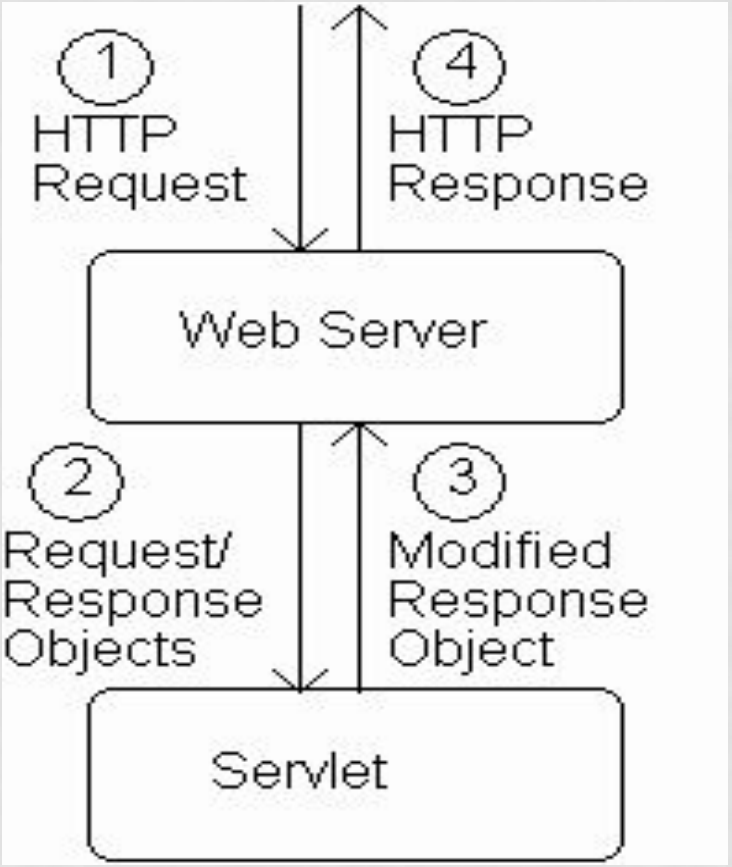
Dynamic Pages



Servlets


WebBrowser
User requests for a web page
HTTP Request
Server's File System
Web Server
Request for static pages
Request for dynamic pages
Requested Output
Send HTTP Response which contains the HTML output of the request
Compose HTML output
The Internet
Query database for data
Login
Bill
Purchase Item

HOW SERVLET WORKS



1. When server starts it instantiates servlets
2. Server receives HTTP request, determines need for dynamic response
3. Server selects the appropriate servlet to generate the response, creates request/response objects, and passes them to a method on the servlet instance
4. Servlet adds information to response object via method calls
5. Server generates HTTP response based on information stored in response object

SERVLET VS. APPLET

- Applets run by browser, servlets run by server.
- Applets are “client-side java”, servlets are “server-side java”.
- Applets makes appearance of web pages alive, servlets makes contents of web pages dynamic.
- Unlike applets, however, servlets have no graphical user interface. Implement only back-end processing.

CHARACTERISTICS

- Provide dynamic content
- Process and/or store the result submitted by html.
- Manage information
- Efficient, robust, persistent, portable and multithreaded.

TYPES OF SERVLET

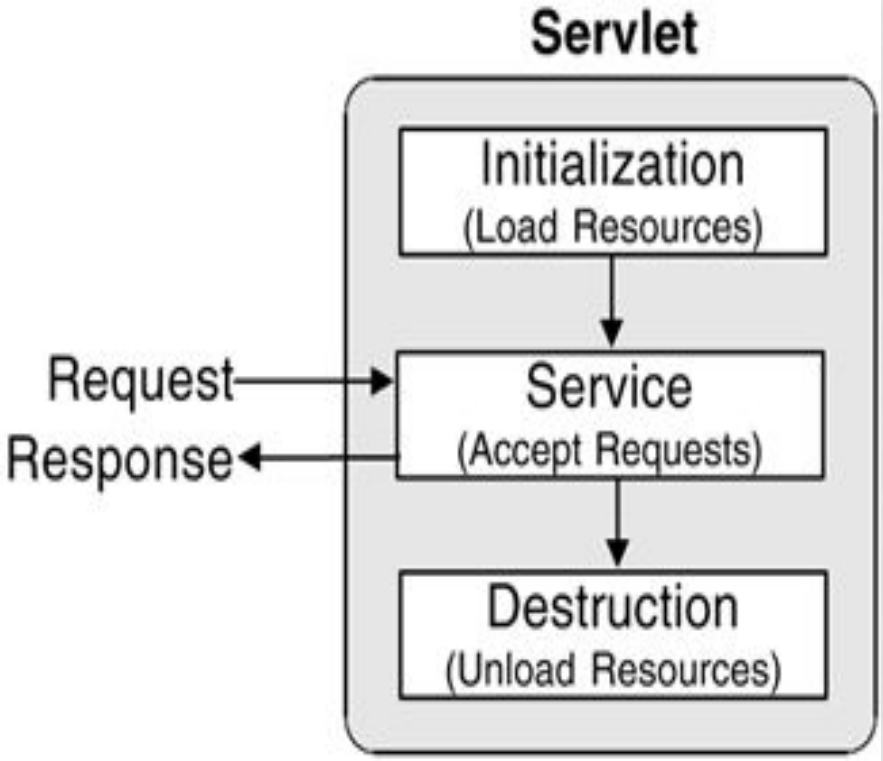
- **Generic Servlet** – super class of HTTP Servlet
  - **Protocol Independent -> handles all type of Protocols**
  - javax.servlet (package)
  - extends javax.servlet.Servlet
  - service method
- **Http Servlet** – Protocol dependent – only HTTP
  - javax.servlet.http (package)
  - extends javax.servlet.HttpServlet
  - doget(), doPost()….

- Generic Servlet:
- GenericServlet class is direct subclass of Servlet interface.
- Generic Servlet is protocol independent.It handles all types of protocol like http, smtp, ftp etc.
- Generic Servlet only supports service() method. It handles only simple request
- public void service(ServletRequest req,ServletResponse res).
- Generic Servlet supports only service() method.

- HttpServlet:
- HttpServlet class is the direct subclass of Generic Servlet.
- HttpServlet is protocol dependent. It handles only http protocol.
- public void service(ServletRequest req,ServletResponse res). protected void service(HTTPServletRequest req,HTTPServletResponse res).
- HttpServlet supports also doGet(),doPost(),doPut(),doDelete(),doHead(),doTrace(),doOptions()etc.

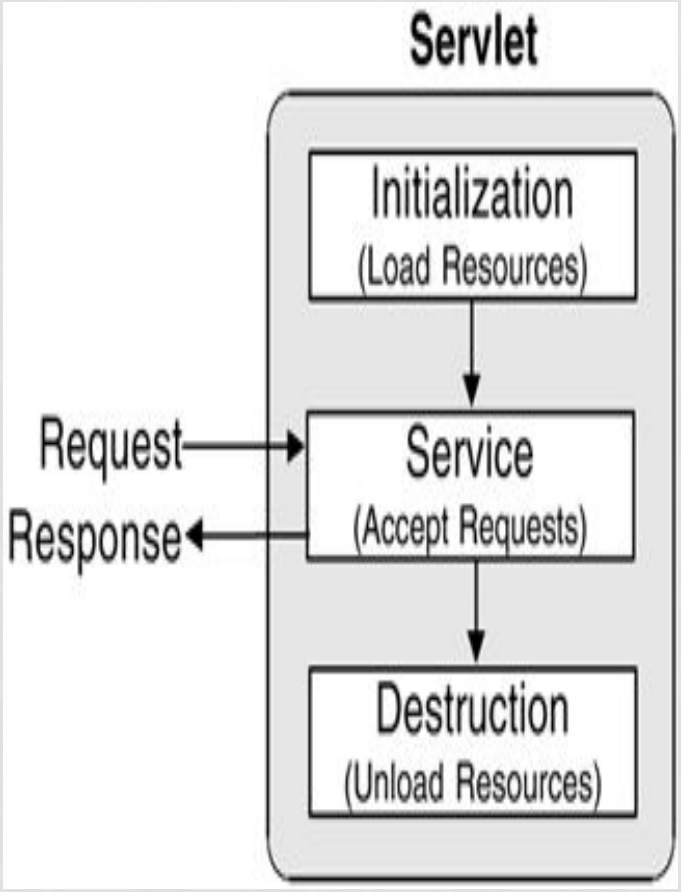
SERVLET LIFE CYCLE

- Managed through a well defined life cycle
- The life cycle is expressed in the javx.servlet.Servlet API by
  - Init() method
  - Service() method
  - Destroy() method
-



INIT()

- Executed once, when the servlet gets loaded for the first time
- Not called for each client request
- Put your initialization code here.(No constructor available)



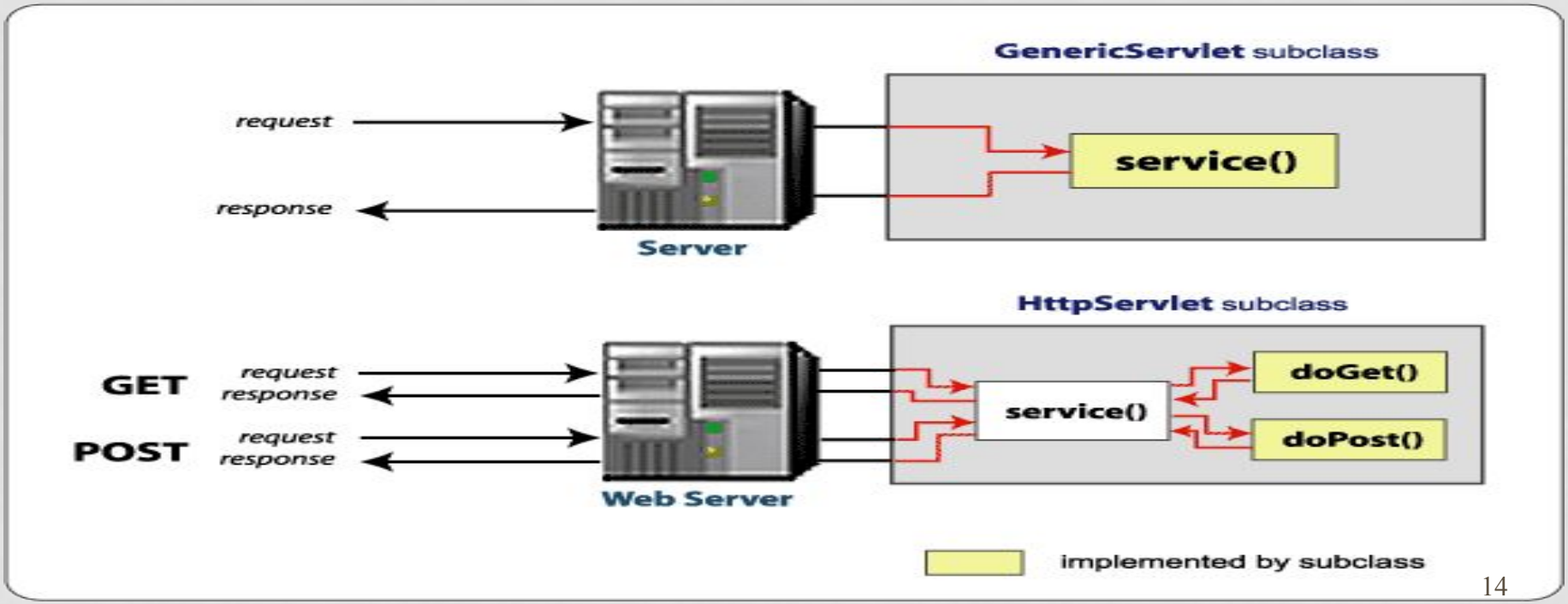
SERVICE()

- is the main method to perform the actual task.
- handle requests coming from the client(browsers) and to write the formatted response back to the client.
- **`doGet()`** - A GET request from an HTML form that has no METHOD specified and it should be handled by doGet() method. **shall be used when small amount of data and insensitive data**
- **doPost() -Shall be used when comparatively large amount of sensitive data has to be sent.**
- Examples are sending data after filling up a form or sending login id and password.

SERVICE()

public void **doGet**(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

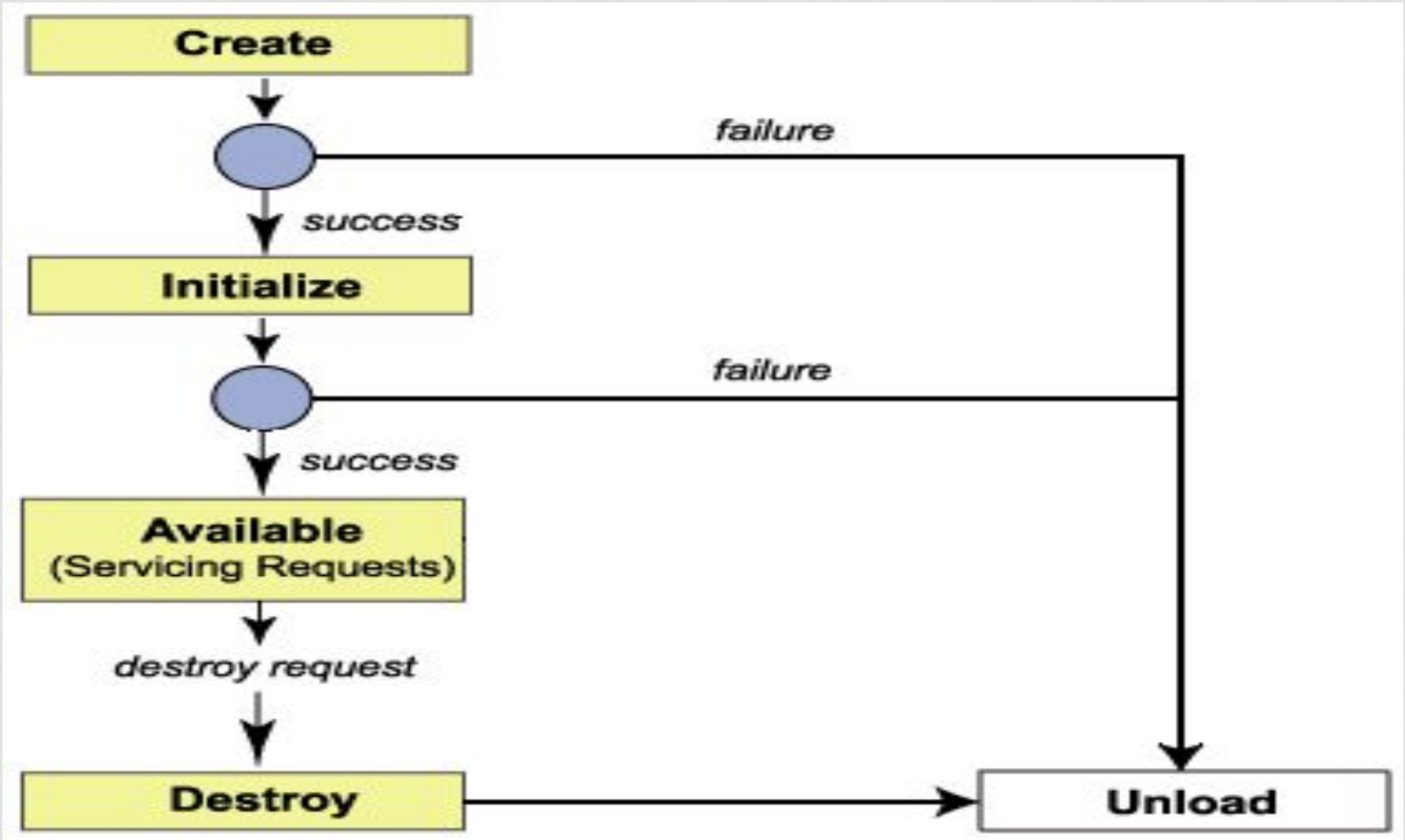
// Servlet code }



DESTROY()

- `destroy()` method is called only once
- Call takes place when
  - Application is stopped
  - Servlet container shuts down
- Allows resources to be freed
- After the destroy() method is called, the servlet object is marked for garbage collection.

SERVLET LIFE CYCLE SUMMARY



STRUCTURE OF SERVLET PROGRAM

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

public class <<servlet name>> extends HttpServlet {

public void doGet (HttpServlet request, HttpResponse
response) throws ServletException, IOException {

// code for business logic here

// use request object to read client's requests

// user response object to throw output back to the client

} // close doGet
} // end program
```

Figure-Basic Servlet Structure

HELLOWORLD

```
Import javax.servlet.*;
Import javax.servlet.http.*;
Import java.io.*;
public class HelloWorld extends HttpServlet {
//public void init() throws ServletException {
    // Do required initialization
    String message = "Hello World";
  }
```

```
public void doGet(HttpServletRequest request,
HttpServletResponse response)
      throws ServletException, IOException {
          // Set response content type
      response.setContentType("text/html");
      // Actual logic goes here.
      PrintWriter out = response.getWriter();
      out.println("<h1>" + message + "</h1>");   }
Public void destroy()
{ System.out.println(“destroy”);}
}
```

SERVLET ACCESSED FROM HTML

<body>

Click here to First Servlet

</body>

HelloWorld – Servlet name

- Java.io – I/O operations
- javax.servlet and javax.servlet.http – classes and interfaces required for operation of servlets
- Javax.servlet – most commonly used class in this package is GenericServlet and HTTPServlet
- Javax.servlet.http – HTTPServletRequest and HTTPServletResponse are commonly used Interfaces
- HTTPServletRequest – Enables the servlet to read data from HTTPRequest
- HTTPServletResponse - Enables the servlet to write data to HTTPResponse
- PrinrWriter() – output stream
- getWriter() – method used for obtaining output stream – sent to client as response

EXAMPLE-1
FIRST.HTML

```
<html>
    <head>   <title>LISTS</title></head>
    <body>
      <form method="get" action="LoginServlet">
       Enter the following Details
  Username: <input type="text" name="name" ><br/>
  Roll No    <input type="text" name="roll"><br/>
       <input type="submit"    value="Submit">

   </form>    </body></html>
```

LOGINSERVLET

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class LoginServlet extends HttpServlet
{
   protected void doGet(HttpServletRequest request, HttpServletResponse
   response) throws ServletException, IOException
   {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
            String name =request.getParameter("name");
            String roll=request.getParameter("roll");
            out.println("Thanks Mr/Ms "+name+"<br>");
            out.println("Please check you details</h1><br>");
            out.println("Name:"+name+"<br>");
            out.println("Roll no:"+roll+"<br>");
   }}
```

EXAMPLE 2

```
<html>
<head><title>First Page</title></head>
<body><form
action="http://localhost:8080/test1/First">
UserName <input type="text" name="n1"><br>
password<input type="text" name="n2"><br>
<input type="submit" value="store">
</form></body></html>
```

FIRST

- **import java.io.*;**
- **import javax.servlet.*;**
- **import javax.servlet.annotation.WebServlet;**
- **import javax.servlet.http.*;**
- **public class First extends HttpServlet {**

```
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {

PrintWriter pw=response.getWriter();
String name=request.getParameter("n1");
String pass=request.getParameter("n2");
pw.println("username: " +name+ "Password: "+pass);
}}
```

DISPLAY DATE AND TIME

- **Date.html:**

```
<html><head><title>display date</title> </head>
<body>
<form action="myservlet"
<input type="submit" value="get your date">
</form></body></html>
```

MYSERVLET

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class DateDisplay extends HttpServlet
{protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws
            ServletException, IOException
{          PrintWriter pw=response.getWriter();
      Date today=new Date();
pw.println("<html><body><h1> Today date</h1>");
pw.println(today);
    }}
```

EXAMPLE

- Write a code for converting currencies.
- 1 Dollar = 0.085 euro
- 1 Dollar = 75 Rupees
- 1 Rupee = .011 euro
- Input(Dollar or Rupee)
- Output(Dollar or Euro)

- Hint: (s1.equals(s2))

EXERCISES

- **1. Write a servlet application to print the current date and time.** import java.util.*; Date date = new Date(); date.toString();
- **2. Write a Servlet program that accepts the age and name and displays if the user is eligible for voting or not.** int age = Integer.parseInt(req.getParameter("age"));
- **3. Write a Servlet application to count the total number of visits on your website.** private int iHitCounter; public void init() throws ServletException { iHitCounter = 0; }

WORKING WITH MULTIPLE VALUES AGAINST A SINGLE FIELD

- Consider that a html form has got a field named hobbies. The field is a list of multiple choices.

```
<select multiple name="hobbies">
<option value="Swimmimg">Swimming</option>
<option value="Boxing">Boxing</option>
<option value="Music">Music</option>
<option value="Football">Football</option>
<option value="Cricket">Cricket</option>
</select>
```

```
{
String hobbies[];
hobbies = request.getParameterValues("hobbies");
for(int i=0; i < hobbies.length; i++)
{ pw.println(hobbies[i] + "<br>"); }
}
```

SESSION TRACKING MECHANISMS

- When a client requests a file from a Web server, the Web server locates and sends the requested file back to the client which is then rendered by the client browser.
- Once the Web server sends the request back to the client, the **connection is torn down.**
- No session maintained between the client and the server.

SESSION TRACKING

- HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.
- Solution: Session Tracking
- **To recognize the user** It is used to recognize the particular user.
- **Session** simply means a particular interval of time.
- **Session Tracking** is a way to maintain state of an user.
  - Three techniques
    - Cookies
    - Hidden Form Fields
    - URL Rewriting

COOKIES

- Cookies: A small piece of data stored on the user's computer by the web browser while browsing a website.
- A cookie is a name-value pair information
- Eg: cookie - Google Search www.google.co.in
- A webserver can assign a unique session ID as a cookie to each web client and for subsequent requests from the client they can be recognized using the received cookie.
- Hidden Form Fields: A web server can send a hidden HTML form field along with a unique session ID as follows: **<input type="hidden" name="sessionid" value="12345">**

COOKIES EXAMPLE

- www.rec.org/cse.html

1. Browser connects to server www.rec.org by making a request.
2. The server replies in the form of HTTP Response with cookies id = 1234
3. cse.html This is another request to the same server. By including cookies which contain id = 1234 server knows that this request is related to the previous one.

Gmail uses cookie technique for login. If you disable the cookie, gmail won't work.

COOKIES

- Creation

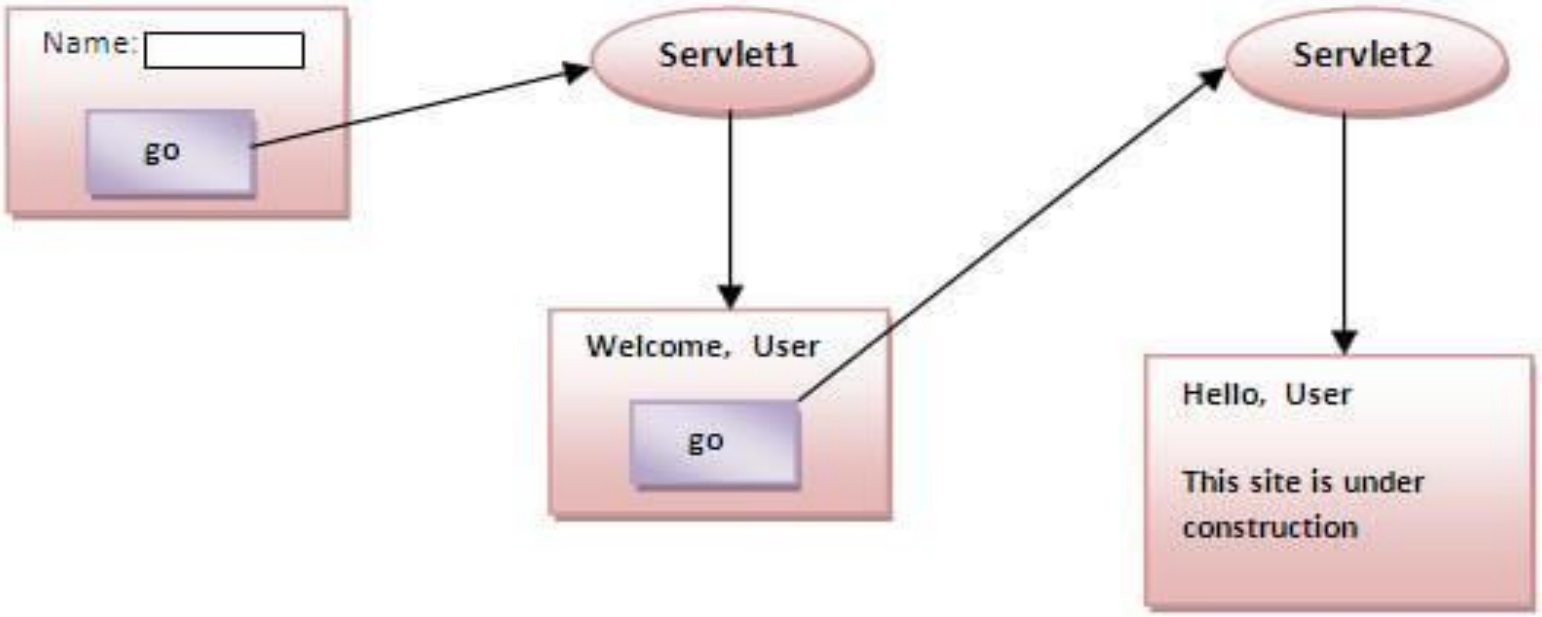
Cookie ck=new Cookie("user",“raja");//creating cookie object
response.addCookie(ck);//adding cookie in the response

- delete cookie. It is mainly used to logout or signout the user.

Cookie ck=new Cookie("user","");//deleting value of cookie
ck.setMaxAge(0);//changing the maximum age to 0 seconds
response.addCookie(ck);//adding cookie in the response

EXAMPLE OF SERVLET COOKIES

storing the name of the user in the cookie object and accessing it in another servlet



INDEX.HTML

```
<form action="servlet1" method="post">

Name:<input type="text" name="userName"/><br/>

<input type="submit" value="go"/>

</form>
```

SERVLET1

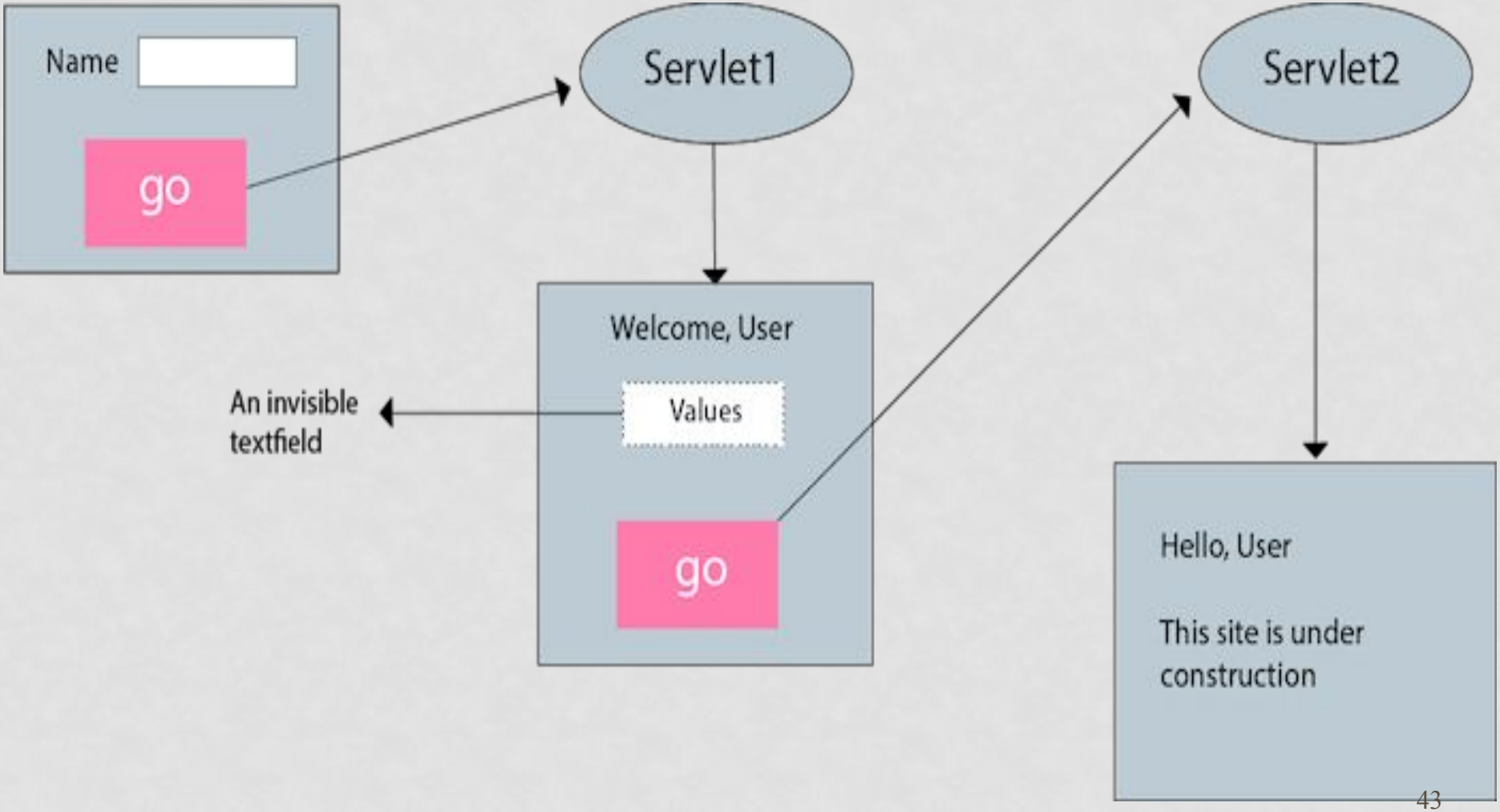
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class servlet1 extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response){
    try{        response.setContentType("text/html");
    PrintWriter out = response.getWriter();

   String n=request.getParameter("userName");
    out.print("Welcome "+n);
    Cookie ck=new Cookie("uname",n);//creating cookie object
    response.addCookie(ck);//adding cookie in the response
      //creating submit button
    out.print("<form action='servlet2'>");
    out.print("<input type='submit' value='go'>");
    out.print("</form>");
    out.close();
        }catch(Exception e){System.out.println(e);} } }
```

SERVLET2

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class servlet2 extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse response){
   try{
  response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    Cookie ck[]=request.getCookies();
    out.print("Hello "+ck[0].getValue());
    out.close();
     }catch(Exception e){System.out.println(e);}
  }  }
```

HIDDEN FORM FIELD

- Hidden Form Field **a hidden (invisible) textfield** is used for maintaining the state of an user.
- we store the information in the hidden field and get it from another servlet.
- <input type="hidden" name="uname" value="raja">
- uname is the hidden field name and
- raja is the hidden field value.


Name
go
Servlet1
Servlet2
Welcome, User
An invisible
textfield
Values
go
Hello, User
This site is under
construction

EXAMPLE

Index.html

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class FirstServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response){
      try{
       response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String n=request.getParameter("userName");
       out.print("Welcome "+n);
            //creating form that have invisible textfield
       out.print("<form action='SecondServlet'>");
       out.print("<input type='hidden' name='uname' value='"+n+"'>");
out.print("<input type='submit' value='go'>");
       out.print("</form>");
      out.close();
            }catch(Exception e){System.out.println(e);}
  } }
```

SECONDSERVLET

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SecondServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
        try{
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
          //Getting the value from the hidden field
           String n=request.getParameter("uname");
           out.print("Hello "+n);
          out.close();
              }catch(Exception e){System.out.println(e);}
    }
}
```

URL REWRITING

- Append a token or identifier to the URL of the next Servlet or the next resource.
- We can send parameter name/value pairs

url?name1=value1&name2=value2&??

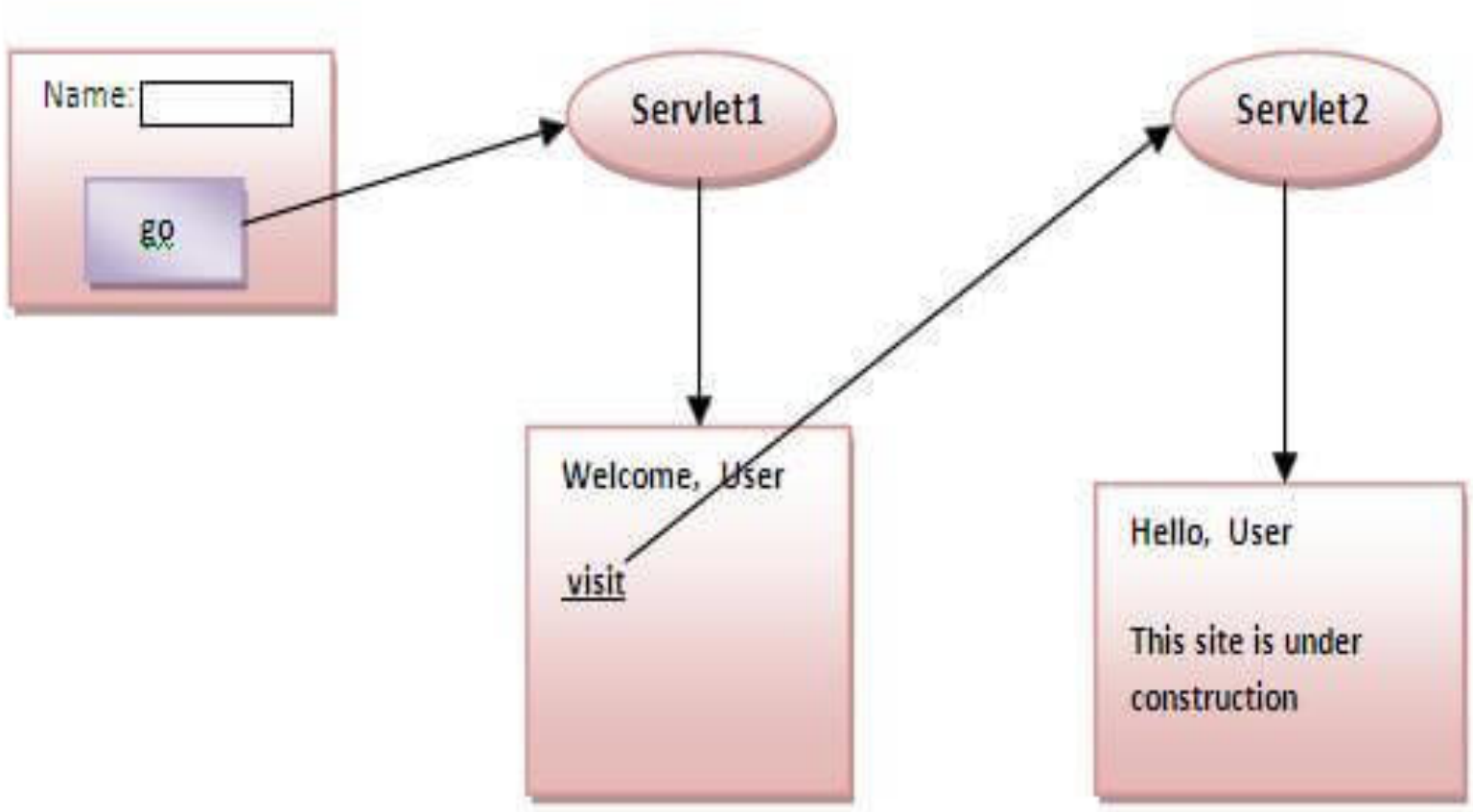
A name and a value is separated using an equal = sign,

a parameter name/value pair is separated from another parameter using the ampersand(&).

When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server.

From a Servlet, we can use getParameter() method to obtain a parameter value.

EXAMPLE


Name:
go
Servlet1
Servlet2
Welcome, User
visit
Hello, User
This site is under construction

FIRSTSERVLET

- **index.html**

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
 public class FirstServlet extends HttpServlet {
 public void doGet(HttpServletRequest request, HttpServletResponse response){
   try{          response.setContentType("text/html");
     PrintWriter out = response.getWriter();
      String n=request.getParameter("userName");
       out.print("Welcome "+n);
//appending the username in the query string
out.print("<a href='SecondServlet?uname="+n+"'>visit</a>");
     out.close();    }catch(Exception e){System.out.println(e);}     }  }
```

SECONDSERVLET

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

public class SecondServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
        try{
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        //getting value from the query string
        String n=request.getParameter("uname");
          out.print("Hello "+n);

        out.close();
                }catch(Exception e){System.out.println(e);}
    } }
```

HIT COUNTER FOR A WEB PAGE

- **total number of hits on a particular page of your website**

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class PageHitCounter extends HttpServlet {
  private int hitCount;
   public void init() {
        hitCount = 0;  // Initialize Global variable     }
```

SERVICE METHOD

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html"); // Set response content type
      hitCount++; // increment hitCount
     PrintWriter out = response.getWriter();
     String title = "Total Number of Hits";
     out.println("<html>\n" +
  "<head><title>" + title + "</title></head>\n" +
        "<body "<h2 " + hitCount + "</h2>\n" +
        "</body></html>“ );   }}
```

CLASSES AND INTERFACES:

- **Import a package java.sql.*** - provides you a network interface, that **enables to communicate between front end and back end.**
- Loading a driver **class.forName ( )** - used to load the class dynamically at runtime
- **DriverManager.getConnection ( )** - This provides you to established a **connection between url and database**.
- **executeQuery ( )** - This method is used to retrieve the record set from a table and store the record set in a result set. The **select statement** is used for this method.
- **next ( )** - This method is used to return the next element in the series.
- **getString ( ) -** This method **retrieve the value of the specific column in the current row of result set** object as string representation.

DATABASE CONNECTIVITY

- **Installation steps** X-operating system, Apache, Mysql, Php, Perl

1. Install XAMPP server for webserver, MySQL db, and PHP https://www.apachefriends.org/index.html
2. Download XAMPP and Install by clicking the downloaded file
3. Open XAMPP control panel, start the service for Apache(web server) and MySQL(for database)
4. open Browser, type http://localhost/dashboard/ to check the server is running
5. To create a **Database and a table** in MySQL – click **phpMyAdmin**
6. Click New (to create a new Database)
7. Provide Database name (Student)and click create
8. Create table (table name) with number of column's and click Go
9. Provide the field names with datatypes
10. insert the data by clicking Insert
11. View the data by clicking Browse

EXAMPLE: DB CONNECTIVITY

1. Create a Form to accept Cid, CName and CCity.
2. Store the values in the database
3. Using Cid, retrieve values of the concerned customer
4. Retrieve the Cid, CName, CCity from the database

CUSINSERT.HTML

```
<form action="AddCustomer" method="get">
Customer Id. :
<input type="text" name="cid"><br>
Customer Name :
<input type="text" name="cname"><br>
Customer City :
<input type="text" name="ccity"><br>
<input type="submit" value="Add Customer">
</form>
```

ADDCUSTOMER.JAVA

```
import java.sql.*;
@WebServlet("/AddCustomer")
public class AddCustomer extends HttpServlet {
protected void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
try {
Class.forName("com.mysql.jdbc.Driver");
String URL = "jdbc:mysql://localhost:3306/customer";
Connection conn = DriverManager.getConnection(URL, "root", "");
```

ADDCUSTOMER.JAVA

```
PreparedStatement ps = conn.prepareStatement("insert into cus_details
values (?, ?, ?)");
ps.setInt(1, Integer.parseInt(request.getParameter("cid")));
ps.setString(2, request.getParameter("cname"));
ps.setString(3, request.getParameter("ccity"));
int res = ps.executeUpdate();
if (res != 0)
out.println("Customer Details Inserted Successfully...");
else out.println("Customer Details Insertion Failure...");
ps.close(); conn.close();
} catch (Exception e) {
out.println(e);}}}
```

CUSVIEW.HTML

```
<form action="ViewCustomer1" method="get">
Customer Id. :
<input type="text" name="cid"><br>
<input type="submit" value="View Customer">
</form>
```

VIEWCUSTOMER1.JAVA

```
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("Thank You");
try {
Class.forName("com.mysql.jdbc.Driver");
String URL = "jdbc:mysql://localhost:3306/customer";
Connection conn = DriverManager.getConnection(URL, "root",
"");
```

VIEWCUSTOMER1.JAVA

```
PreparedStatement ps=conn.prepareStatement("select * from cus_details where cusid=?");
Integer cid1=Integer.parseInt(request.getParameter("cid"));
ps.setInt(1,cid1);
ResultSet rs=ps.executeQuery();
if(rs.next()){
out.println("<center><h1>Customer Details</h1></center>");
out.println("<hr>");
out.println("Customer Id :"+rs.getInt(1));
out.println("<br>")
```

VIEWCUSTOMER1.JAVA

```
out.println("Customer Name :"+rs.getString(2));
out.println("<br>");
out.println("Customer City :"+rs.getString(3));
out.println("<br>"); }
conn.close();
}catch(Exception e){ System.out.println(e);}
} }
```

TRY

1. Create a Database Employee
2. Create a Table Emp with empno, empname, BasicPay, HRA, DA, PF, Netsalary
3. Collect the Employee data like empno, empname and BasicPay thru Form and store it in Database
4. Calculate the HRA as 10% of BasicPay, DA as 35% of BasicPay, PF as 12% of BasicPay and NetPay as BasicPay+HRA+DA-PF. Store it in Database.
5. Display the employees whose Netpay > 40000