***WEB BASED PROGRAMMING USING JAVA***

Advance Java

Web with java

J2EE/ Java EE/ Jakarta(latest)

For creating webpage, we have two parts

1. Static Part
2. Dynamic Part

|  |  |  |
| --- | --- | --- |
| **Static Part** | **Dynamic Part** | |
| 1. HTML 2. CSS 3. Bootstrap | Client Side   1. JavaScript | Server Site   1. Java 2. .Net 3. Python 4. php |
| * Informative part of web page | * Interactive Part of webpage. | |

**JEE: -**

It is a server which is responsible to develop middle tire of java web application. It contains two types of containers: -

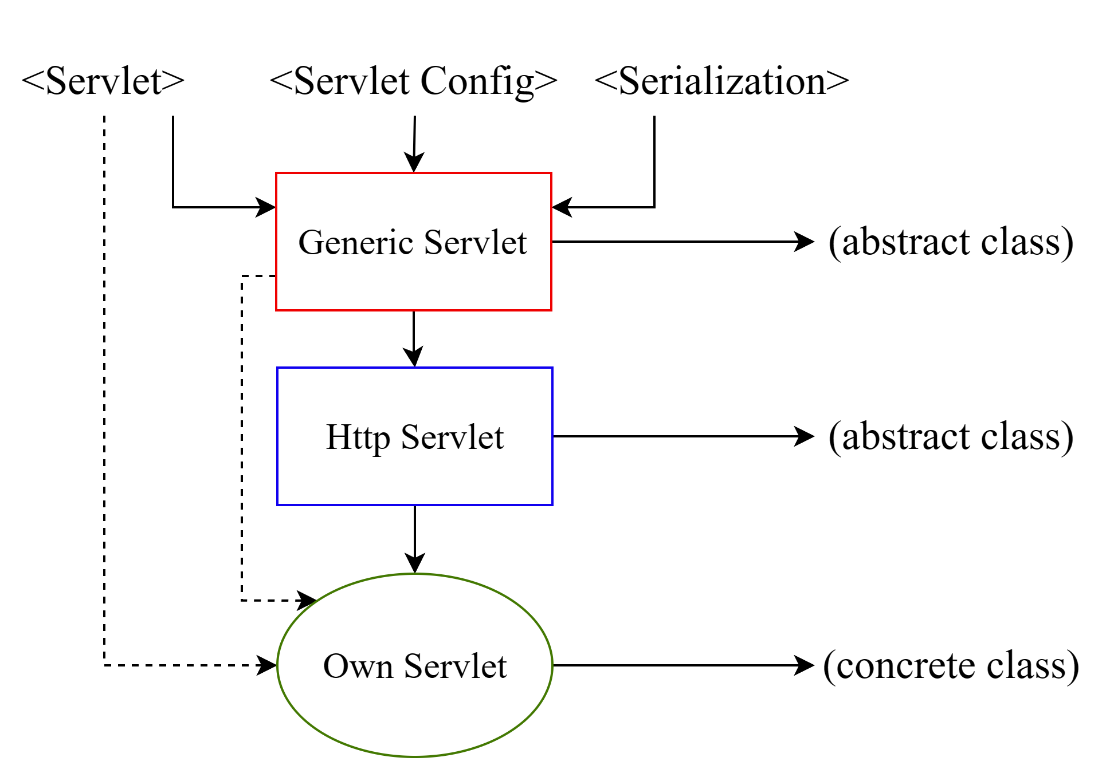
1. Web container (it consists servlet and JSP)
2. EJB container (which fascinated to develop distributed enterprise level application)

Java EE is also responsible to provide transection management, messaging services and security related components.

**SERVLET: -**

It is a component of web container which is responsible to execute on web/application server and exploits its functionality for managing clients request and servers’ response on server site.

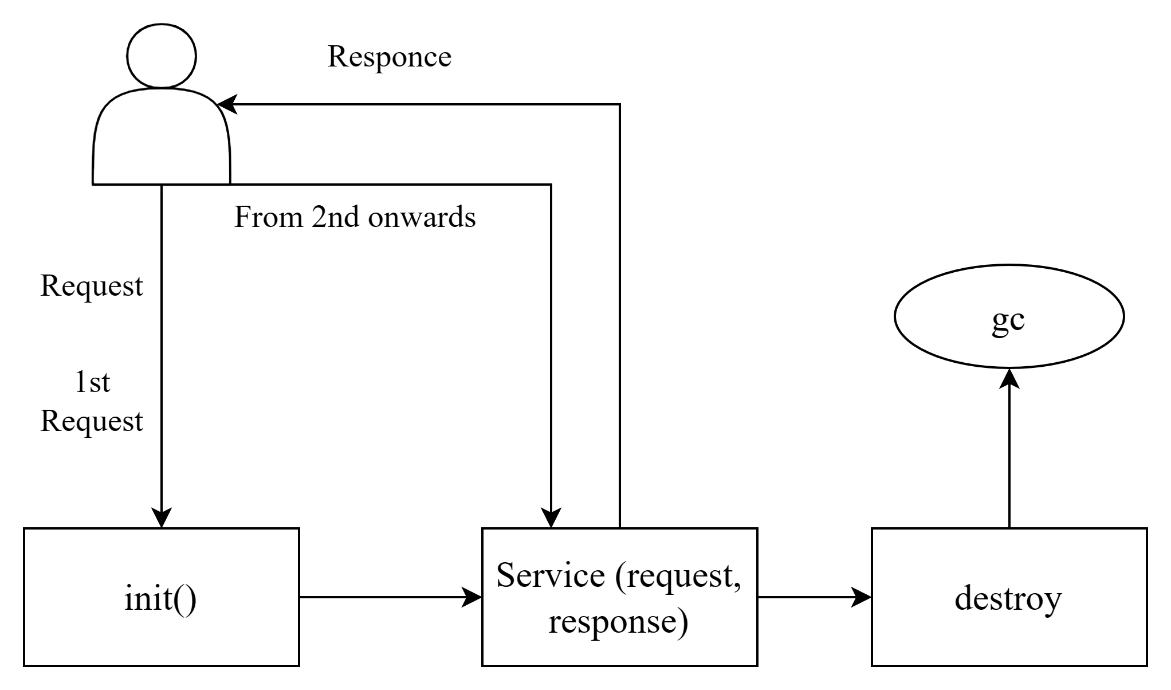
HIERARCHY: -



Difference between Generic Servlet and Http servlet: -

|  |  |
| --- | --- |
| **Generic Servlet** | **Http servlet** |
| It supports all protocol. | It supports only http protocol. |
| If we want to use functionality of other protocol in our servlet then it should be inherited from generic servlet. | If we don’t require other functionality other than http protocol then Http servlet should be inherited for creating our own servlet. |
| Inside generic servlet service method is in abstract format. | In Http servlet all methods are implemented along with its own resources. |

Life cycle of Servlet: -



Difference between doGet and doPost: -

|  |  |
| --- | --- |
| **doGet** | **doPost** |
| * It is a callback method | * It is not a call back method |
| * Performance wise it is faster | * Performance wise it is slower |
| * Limited data can be transfer from client to server | * Unlimited data can be transfer from client to server |
| * Since information which is transfer through URL it is visible to client that’s why it is less secure | * Information is transfer from client to server in encrypted format without showing it on URL that’s why it is more secure |

Fetching Info from Client to Server: -

1. String str = request.getParameter(“paraName”);

For less information

1. Enumeration en = request.getParameter(“”);

For more information

1. String str [] = request.getParameterValues(“paraName”);

For single parameter having multiple values

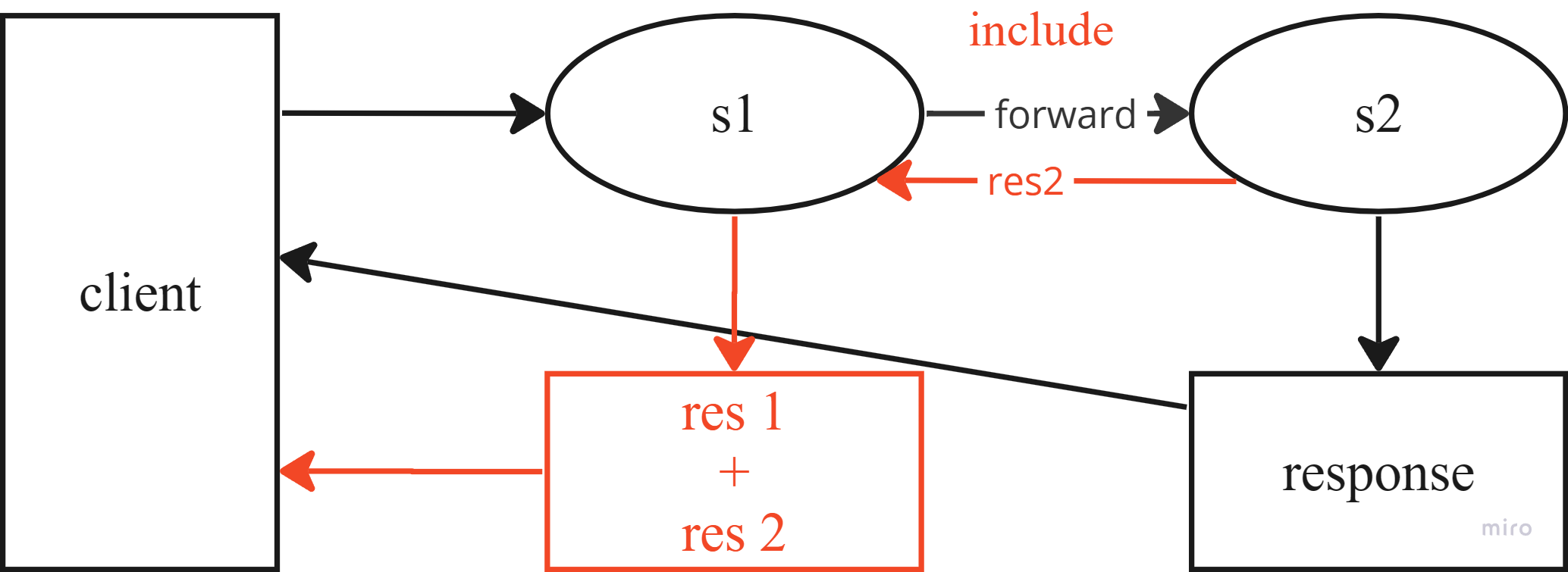
**InterServlet Communication: -**

1. By using RequestDispatcher

Methods: -

1. include (req, res): - It is the method of request dispatcher interface which is responsible to produce response of all the servlet to the client
2. forward (req, res): - Method of request dispatcher interface which is responsible to produce response of final servlet to the client.

**Note: -** Both the method works for more than one servlet communication within the application.

****

1. By using ServletContext and ServletConfig
2. ServletContext: - It is globally defined for the application and responsible to provide information which it contains to all servlets of the application.
3. ServletConfig: - Each servlet of the application has their own servlet config and which is responsible to provide information to particular servlet only

**Scope Resolution: -**

1. page (less use)
2. request
3. Session
4. Application (also known as Servlet Context)

All have two methods: -

1. setAttribute (key, value)
2. getAttribute (key)

**SESSION MANAGEMENT: -**

**Session: -** It is the period/duration of completion of any task.

**Session Management: -** Since web application uses Http protocol and by default this protocol is stateless protocol it means, it does not maintain the state of client so for maintaining the state of client we need to manage the session.

Http protocol can never be stateful protocol it may behaviour it stateful.

Four types: -

1. URL Rewriting
2. Hidden Field client side
3. Cookies
4. Session Api server

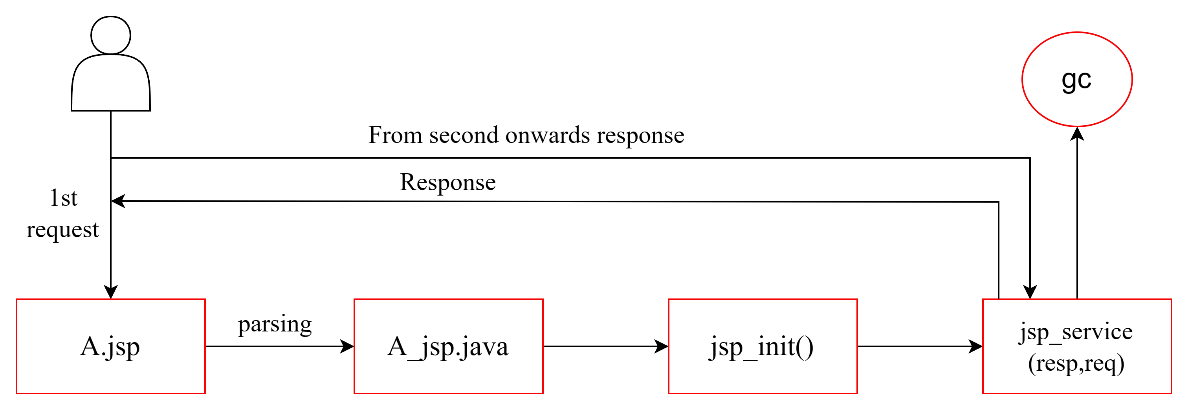
Difference between Cookies and Session Api

|  |  |
| --- | --- |
| **Cookies** | **Session Api** |
| * It is browser facility and works on client side | * It works on server site |
| * Limited data 4Kb   Limited data can be stored | * Unlimited data can be stored |
| * Less secure | * More secure |
| * Performance wise faster | * It is slower |
| * Types  1. Persistent cookie 2. Temporary cookie |  |

**JAVA SERVER PAGES (JSP): -**

It is one of the components of web container which is responsible to execute on web server and exploit their functionality. It is also known as server site template of java which will create dynamic webpage.

Life Cycle: -



Tag of JSP: -

1. Comments: -

<%-- comments --%>

1. Declarative tag

<%! for instance, variable/ user define method %>

1. Scriptlet tag

<% code which we want to write in service () %>

1. Expression tag

<%= substitute of out. println () %>

**Implicit Objects: -**

In JSP some frequently use objects are created implicitly at time of parsing from JSP to Servlet so that there is no need to create this object explicitly user can directly use all implicit objects.

Types of implicit objects: -

1. out 🡪 JSPWriter
2. request 🡪 HttpServletRequest
3. response 🡪 HttpServletResponse
4. config 🡪 ServletConfig
5. application 🡪 ServletContext
6. session 🡪HttpSession
7. page 🡪 Page
8. pageContext
9. exception 🡪 Exception

**JSP Directives: -**

Types of JSP Directives: -

1. page 🡪 <%@page import = “java. util. \*” %>

Attributes of page Directives: -

* import
* extends : - by default **value**
* language : - by default **java**
* session : - by default **true**
* containType : - by default **text/html**
* pageEncoding
* errorPage : - by default **value**
* isErrorPage : - by default **false**
* buffer : - by default **8 kb**
* autoFlush : - by default **true**
* isThreadSafe : - by default **false**
* isElIgnore (Expression language)
* info

Note: -

buffer and autoFlush works in pair

errorPage and isErrorPage works in pair

1. include 🡪 <%@include file =“”%>

Attributes of page Directives: -

* file

1. taglib 🡪 use for JSTL,

it is user define tag

**include directive is static by nature**

**include standard action is dynamic by nature**

**JSP Standard Tag Library (JSTL): -**

<%@ taglib prefix = “ ” uri = “ ”%>

**JAVA DATABASE CONNECTIVITY (JDBC): -**

It is a java API which is responsible to establish communication between java application and any backend SQL database.

It is only use for SQL not for NoSQL.

Driver: -

It is a software implementation which is responsible to establish communication between two different entities.

There used to be 4 types of drivers in JDBC: -

1. JDBC – ODBC Bridge driver (Type – I)
2. Native API Driver (Type – II): - Created using C language, partial Java Driver.
3. Net Protocol Driver (Type – III): - Fully Java Driver, by processing time was slow.
4. Thin Driver (Type – IV): - Fully Java Driver, direct communication between client and database.

ODBC is the driver of Microsoft

How to Establish Connectivity: -

Step 1 🡪 Loading the driver

Class.forName(“Driver (String)”); //Static method as it is call through class name.

Step 2 🡪 Fetching the connection

Connection con = DriverManager.getConnection(url, username, password); //It is the interface.

Step 3 🡪 Create Statement

Statement stmt = con.createStatement(); //non static method

stmt.executeUpdate(“Insert/Delete/Update”);

Step 4 🡪 Create ResultSet

ResultSet rs = stmt. execteQuery (“select \* from tablename”); //Interface and only if fetching data select query\

**In case of prepared Statement**

Step 3 🡪 Create Statement

PreparedStatement pst = con.prepareStatement(“Any Query”);

int a = pst.executeUpdate();

//To hold value, we use int a

Step 4 🡪 Create ResultSet

ResultSet rs = pst. execteQuery ();

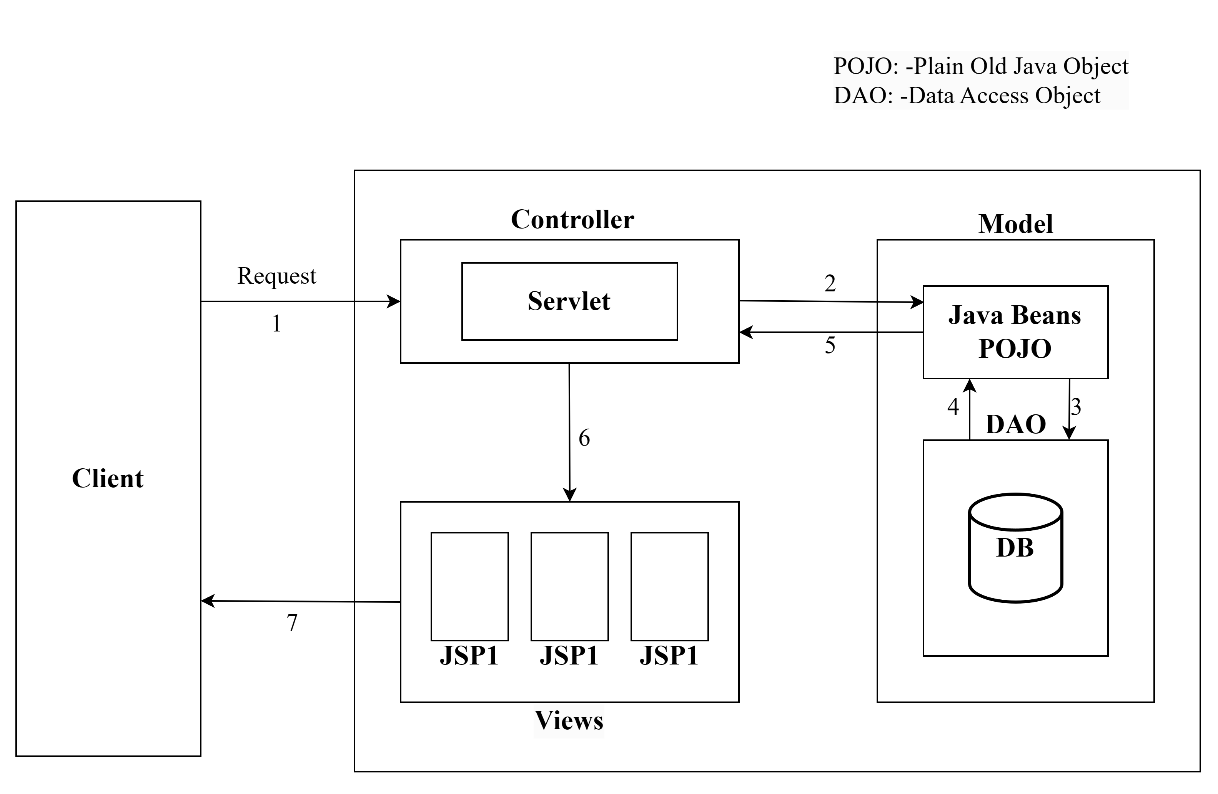
Statement **🡪** Select \* from Employee where id = 101;

PreparedStatement **🡪** Select \* from Employee where id = … and name = …;

**MODEL-VIEW CONTROLLER (MVC): -**

MVC-1: - Page centric has view + controller as one.

MVC-2: - Front controller, request is 1st fetch by controller then other request are fulfilled.



Model: - It is the component of MVC which contain the business logic to process the request render by the controller and interact with database with the help of DAO and generate response for controller.

View: - It is the component in which we put presentation logic and responsible to produce final response to the client.

Controller: - It is the most important component of MVC which is responsible to take decision on the basis of request from the client and response generated by its corresponding model that which view should be render to the client/browser.

**HIBERNATE: -**

It is the java framework which provide strong Object Relational Mapping (ORM) services. It is the extension of JDBC.

Object 🡪 Java class

Relation 🡪 Database

**Framework: -** It is the bundle of classes, interfaces, libraries and configuration files which facilitated to create and application with higher security, performance and efficient management. It is preferable to use framework for huge application.

Features: -

1. Transparent persistency: - it is abstracted.
2. It generates schema/table automatically.
3. It generates primary key automatically.
4. It enhances memory management: - it uses cache memory.
5. It enhances performance.
6. It supports scalability.
7. It uses query language independent to database.
8. It is industrial reliable.
9. It supports transection management.
10. It uses connection pulling.

Architecture of hibernate: -

