

# Java for server side programming

Unit 3

# JSP(Java server page)


- **JSP** technology is used to create dynamic web application. It executes in server side. A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain because we can separate designing and development.

# Creating a simple JSP Page

```
<html>  
<body>  
<% out.print("welcome to jsp"); %>  
</body>  
</html>
```

Output:

Welcome to jsp



```
<html>
<body>
<%
out.print(2*5);
%>
</body>
</html>
```

Output:  
10

# Add two number

```
<html>
<head><title>Sum of two number</title></head>
<body>
<form method="post" action="addition.jsp">
<br>First NO <input type="text" name="no1" value="">
<br>Second NO <input type="text" name="no2"
value="">
<br><input type="submit" value="Add" name="submit">
</form>
</body>
</html>
```

```
<html>
<head><title>Sum of two number</title></head>
<body>
Sum of number <br>
<%
int x,y;
x=Integer.parseInt(request.getParameter("no1"));
y=Integer.parseInt(request.getParameter("no2"));
out.print(x+y);
%>
</body>
</html>
```



# JSP implicit objects

- JSP implicit objects are used in a JSP page to make the page dynamic. JSP implicit objects are predefined objects that are accessible to all JSP pages. These objects are called implicit object because you don't need to instantiate these objects.
- The JSP container automatically instantiates these object while writing the script content

# Types of JSP implicit objects:

There are **9 jsp implicit objects** which are given below.

Object	Type
request	HttpServletRequest
response	HttpServletResponse
out	JspWriter
config	ServletConfig
application	ServletContext
session	HttpSession
pageContext	PageContext
page	Object
exception	Throwable



## 1) JSP request implicit object

□ The **JSP request** is an implicit object of type `HttpServletRequest` i.e. created for each jsp request by the web container. It can be used to get request information such as parameter, header information, remote address, server name, server port, content type, character encoding etc.

□ **Example of JSP request implicit object**  
**request.html**

```
<form action="request.jsp">  
<input type="text" name="uname">  
<input type="submit" value="go"><br>  
</form>
```

**request.jsp**

```
<%  
String name=request.getParameter("uname");  
out.print("welcome "+name);  
%>
```

## 2) JSP response implicit object

- In JSP, response is an implicit object of type `HttpServletResponse`. The instance of `HttpServletResponse` is created by the web container for each jsp request.
- It can be used to add or manipulate response such as redirect response to another resource, send error etc.
- **Example of response implicit object**

### **index.html**

```
<form action="response.jsp">  
<input type="text" name="uname">  
<input type="submit" value="go"><br/>  
</form>
```

### **welcome.jsp**

```
<%  
response.sendRedirect("http://www.google.com");  
%>
```

### 3) JSP **out** implicit object

- For writing any data to the buffer, JSP provides an implicit object named out. It is the object of JspWriter.

- **Example of out implicit object**

```
<html>
```

```
<body>
```

```
    <% out.print("welcome to nepal"); %>
```

```
    </body>
```

```
</html>
```

#### **4) JSP config implicit object**

- In JSP, config is an implicit object of type *ServletConfig*. This object can be used to get initialization parameter for a particular JSP page. The config object is created by the web container for each jsp page.

#### **5) JSP application implicit object**

- In JSP, application is an implicit object of type *ServletContext*.
- The instance of ServletContext is created only once by the web container when application or project is deployed on the server.
- This object can be used to get initialization parameter from configuration file . It can also be used to get, set or remove attribute from the application scope.

## **6) session implicit object**

- In JSP, session is an implicit object of type HttpSession. The Java developer can use this object to set, get or remove attribute or to get session information.

## **7) pageContext implicit object**

- In JSP, pageContext is an implicit object of type PageContext class. The pageContext object can be used to set, get or remove attribute from one of the following scopes: page
  - request
  - session
  - application

# Servlet

- Servlet is a Java program that can perform dynamic operations and send it to the web server. The web server then sends this response to the web client.



# Advantages of Servlet

- ❑ **Better performance:** because it creates a thread for each request, not process.
- ❑ **Portability:** because it uses Java language.
- ❑ **Robust:** JVM manages Servlets, so we don't need to worry about the memory leak, garbage collection, etc.
- ❑ **Secure:** because it uses java language.

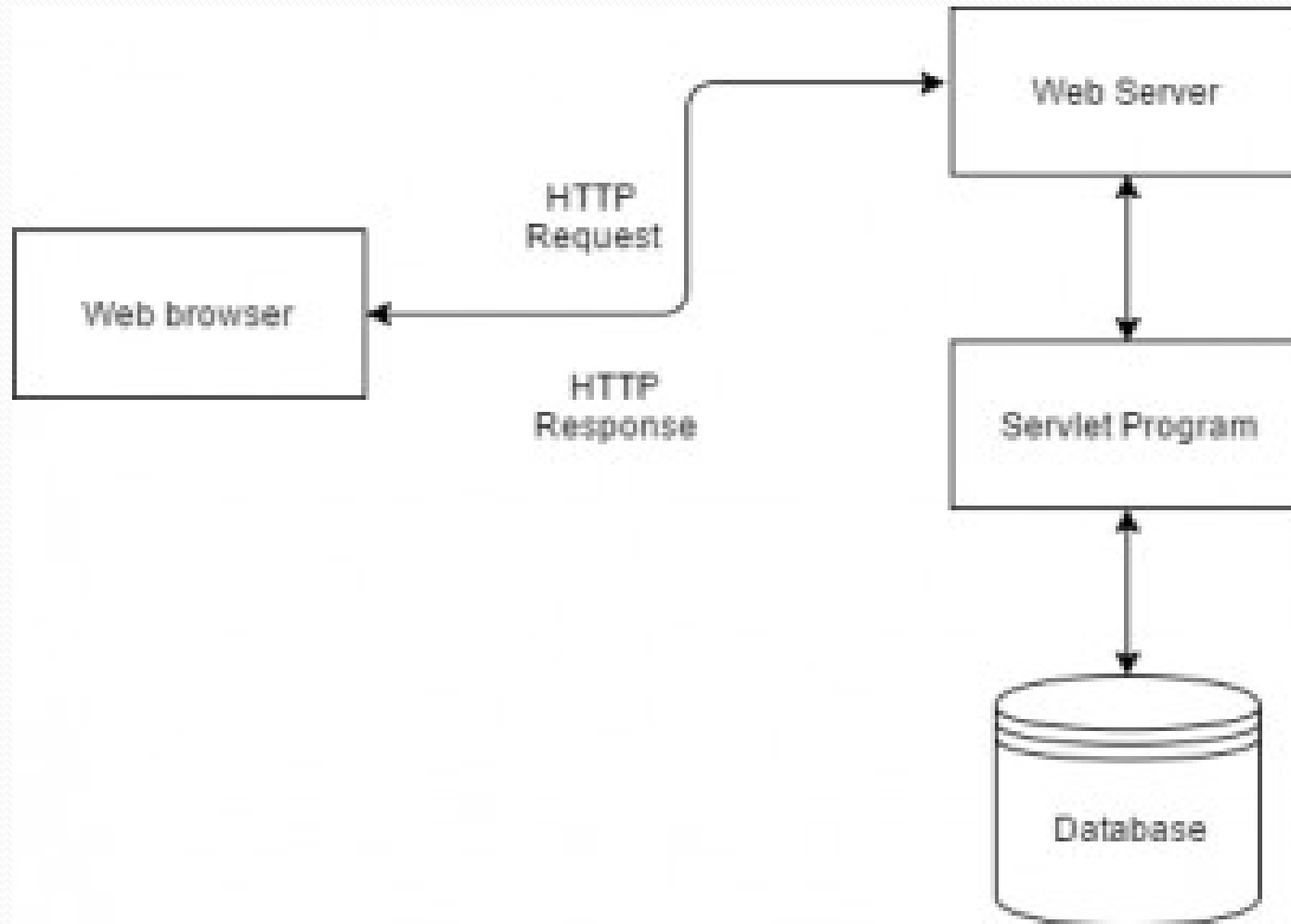
# Life Cycle of a Servlet

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.

- 1. Load Servlet Class:** Web container loads the servlet when the first request is received. This step is executed only once at the time of first request.
- 2. Create Servlet instance:** After loading the servlet class web container creates the servlet instance. Only one instance is created for a servlet and all concurrent requests are executed on the same servlet instance.
- 3. Call init() method:** After creating the servlet instance web container calls the servlet's init method. This method is used to initialize the servlet before processing first request. It is called only once by the web container.

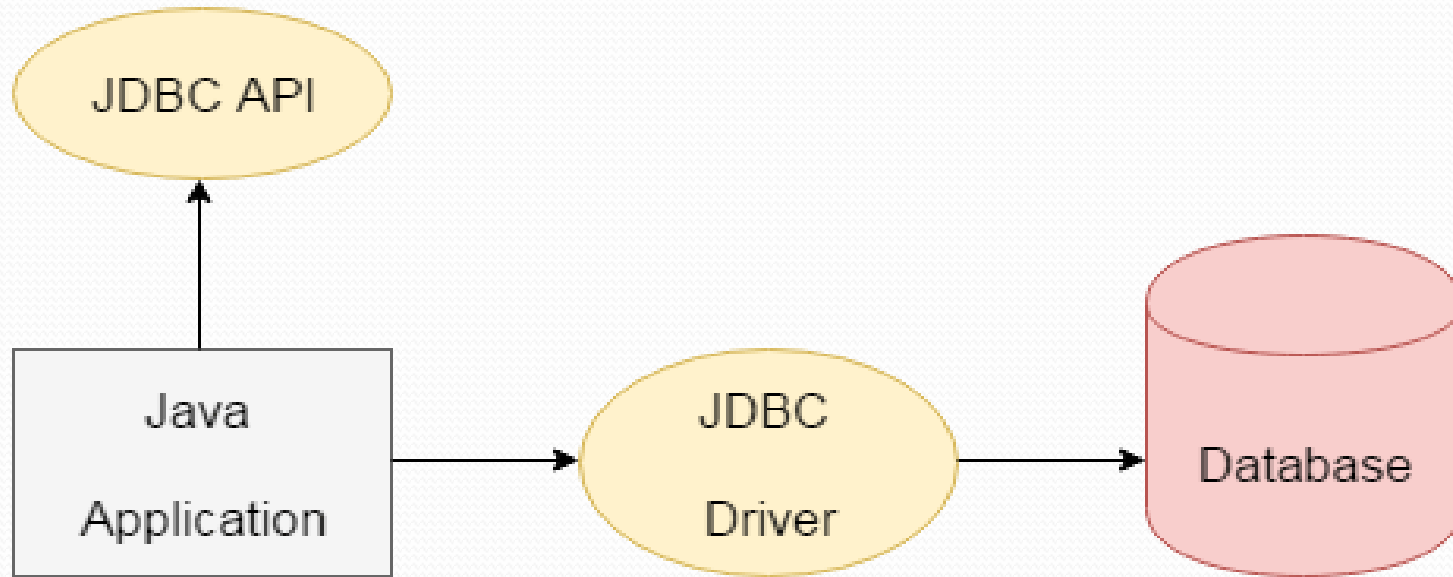
- 4. Call service() method:** After initialization process web container calls service method. Service method is called for every request. For every request servlet creates a separate thread.
- 5. Call destroy() method:** This method is called by web container before removing the servlet instance. Destroy method asks servlet to releases all the resources associated with it. It is called only once by the web container when all threads of the servlet have executed or in a timeout case.

# Fig servlet architecture



# Java JDBC

- JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database.





# Java Database Connectivity with 5 Steps

□ There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

- 1) Register the Driver class
- 2) Create connection
- 3) Create statement
- 4) Execute queries
- 5) Close connection

# 1) Register the driver class

- The **forName()** method of Class is used to register the driver class. This method is used to dynamically load the driver class.

## **Syntax of forName() method**

- `public static void forName(String className)throws ClassNotFoundException`

## **Example to register the mysql Driver class**

- Here, Java program is loading mysql driver to establish database connection.
- `Class.forName("com.mysql.jdbc.Driver");`

## 2) Create the connection object

- The **getConnection()** method of DriverManager class is used to establish connection with the database.

### **Syntax of getConnection() method**

- 1) public static Connection getConnection(String url) throws SQLException
- 2) public static Connection getConnection(String url, String name, String password) throws SQLException

### **Example to establish connection with the mysql database**

```
Connection con=DriverManager.getConnection(  
"jdbc:mysql://localhost:3306/test","root"," ");  
//
```

here test is database name, root is username and password

### **3) Create the Statement object**

- The `createStatement()` method of `Connection` interface is used to create statement. The object of statement is responsible to execute queries with the database.

**Syntax of `createStatement()` method is:**

- `public Statement createStatement()throws SQLException`

**Example to create the statement object is:**

- `Statement stmt=con.createStatement();`

## 4) Execute the query

- The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

### **Syntax of executeQuery() method**

- `public ResultSet executeQuery(String sql) throws SQLException`

### **Example to execute query**

```
ResultSet rs=stmt.executeQuery("select * from emp")
;
while(rs.next()){
System.out.println(rs.getInt(1)+" "+rs.getString(2));
}
```

## 5) Close the connection object

- By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. **Syntax of close() method**
- `public void close()throws SQLException`

### **Example to close connection**

- `con.close();`



## Example to Connect Java Application with mysql database

- In this example, test is the database name, root is the username and password both.

```
import java.sql.*;
class MysqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test","root","root");
//here sonoo is database name, root is username and password
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
con.close();
}catch(Exception e){ System.out.println(e);}
}
}
```



\*\*\*\*\*THE END\*\*\*\*\*