**Chapter 3-Back-end development**

**1. What is back-end development?**

* Back-end development means working on server-side software, which focuses on everything you can’t see on a website.
* Back-end developers ensure the website performs correctly, focusing on databases, back-end logic, application programming interface (APIs), architecture, and servers.
* They use code that helps browsers communicate with databases, store, understand, and delete data.
* back-end developers collaborate with front-end developers, product managers, principal architects, and website testers to build the structure of a website or mobile app.
* Back-end developers must be familiar with many kinds of tools and frameworks, including languages such as Python, Java, and Ruby.
* They make sure the back-end performs quickly and responsively to front-end user requests.

## 1.1 Back-end developer tasks and responsibilities

* Back-end developers are required to have technical expertise, analytical thinking, and excellent collaboration skills. As a back-end web developer, you should be able to work independently to design the web infrastructure.

Here’s what many back-end developers do on a day-to-day basis:

* **Build and maintain websites:**A back-end developer’s main responsibility is to use various tools, frameworks, and languages to determine how best to develop intuitive, user-friendly prototypes and turn them into websites. This requires an understanding of cross-platform functionality and compatibility.
* **Write high-quality code:**To produce sustainable web applications, developers must write clean and easily maintainable code.
* **Perform quality assurance (QA) testing:**Create and oversee testing schedules to optimize user interface and experience, ensuring optimal display on various browsers and devices.
* **Assess efficiency and speed:**Once a website is up and running, and during updates and edits, developers need to assess its performance and scalability, adjusting code as necessary.
* **Troubleshoot and debug:**Be able to troubleshoot issues and resolve them, while communicating them to project managers, stakeholders, and QA teams.
* **Train and support:**Maintain workflows with client teams to ensure ongoing support, along with leading training and mentorship for junior developers.

### **1.2 What tools do back-end developers use?**

* Web developers use a variety of tools to develop, test, and maintain web applications. Some common tools for back-end developers include:
* Programming languages:
  + Python
  + PHP
  + JavaScript
  + Ruby
  + Java
  + C#
* Frameworks:
  + Laravel
  + Django
  + Spring
  + Ruby on Rails
  + Meteor
  + Node.js
* Databases:
  + MongoDB
  + MySQL
  + Oracle
* Servers:
  + Apache
  + NGINX
  + Lighttpd
  + Microsoft IIS

**1.3 Back-end developer technical skills**

As a back-end developer, there are certain technical skills you will need to learn to navigate developing the back-end of the web or mobile application.

* **Programming languages:**Any back-end developer needs to be well-versed in back-end programming languages such as Python, Java, and PHP. These make the website function when used alongside databases, frameworks, and servers. Python is one of the most popular programming languages because it is compatible with artificial intelligence (AI) and machine learning, and works well for writing clear and logical code. Basic knowledge of front-end languages HTML, CSS, and JavaScript is a bonus.
* **Frameworks:**Frameworks are the libraries of back-end programming languages that help to build the server configuration. They tend to be linked with programming languages, so if you are familiar with Python, you’ll also know Flask, Django, or another Python-based framework, and so on.
* **Databases and servers:**You’ll need to understand how to stack and recover data from databases, as back-end programming controls access to this information, including storage and recovery. MongoDB and MySQL are popular database programs. The database stores and organizes the client’s data so that it can be easily arranged and recovered, just like you might use cloud storage for your photos. This database then runs on a server that provides data upon request.
* **Application Program Interface (API):**An API is a series of definitions and rules for developing application software. In addition to internet browser websites, companies often want a mobile app for iOS or Android. Knowledge of application-building languages like JavaScript will expand your job opportunities.
* **Accessibility and security clearance:**You should develop knowledge of network protocols and web security. Knowing how to secure databases and servers will be critical to your success as a back-end developer.

# **2. Introduction to Java Servlets**

**Servlet** technology is used to create a web application (resides at server side and generates a dynamic web page



What is a Servlet?

* Servlet can be described in many ways, depending on the context.



* Servlet is a technology which is used to create a web application.



* Servlet is an API that provides many interfaces and classes including documentation.



* Servlet is an interface that must be implemented for creating any Servlet.



* Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.



* Servlet is a web component that is deployed on the server to create a dynamic web page.



* Servlets are the Java programs that run on the Java-enabled web server or application server.
* They are used to handle the request obtained from the webserver, process the request, produce the response, then send a response back to the webserver.



* Properties of Servlets are as follows:

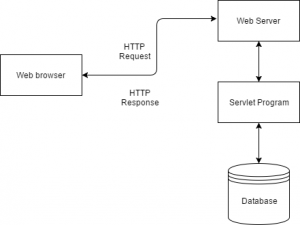


1. Servlets work on the server-side.
2. Servlets are capable of handling complex requests obtained from the webserver.



Servlet Architecture is can be depicted from the image itself as provided below as follows:





Execution of Servlets basicallyinvolves six basic steps:

1. The clients send the request to the webserver.
2. The web server receives the request.



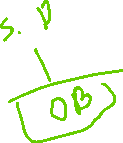
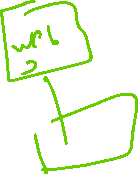
1. The web server passes the request to the corresponding servlet.
2. The servlet processes the request and generates the response in the form of output.



1. The servlet sends the response back to the webserver.



1. The web server sends the response back to the client and the client browser displays it on the screen.



**2.1 What is CGI?**

* **CGI** is actually an external application that is written by using any of the programming languages like **C** or **C++** and this is responsible for processing client requests and generating dynamic content.



In CGI application, when a client makes a request to access dynamic Web pages, the Web server performs the following operations :



* It first locates the requested web page *i.e* the required CGI application using URL.
* It then creates a new process to service the client’s request.



* Invokes the CGI application within the process and passes the request information to the application.

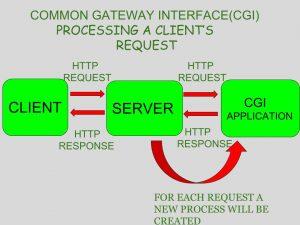


* Collects the response from the CGI application.
* Destroys the process, prepares the HTTP response, and sends it to the client.



So, in **CGI** server has to create and destroy the process for every request. It’s easy to understand that this approach is applicable for handling few clients but as the number of clients increases, the workload on the server increases and so the time is taken to process requests increases.





**Advantages of CGI:**



Can be written in variety of languages, primarily written in Perl  
Bugs does not crash the web server  
Easy for reference  
Execute in their own OS Shell, doesn’t have concurrency conflicts  
All service providers support CGI Programs  
  
**Disadvantages:**Response time is high, the creation of an OS Shell is an heavy weight activity  
CGI is not scalable  
Not always secure or object-oriented  
No separation of presentation and business logic  
Scripting languages are often platform-dependent



Adv . of Servlet

1. **Better performance:** because it creates a thread for each request, not process.



1. **Portability:** because it uses Java language.



1. **Robust:** [JVM](https://www.javatpoint.com/jvm-java-virtual-machine) manages Servlets, so we don't need to worry about the memory leak, [garbage collection](https://www.javatpoint.com/Garbage-Collection), etc.



1. **Secure:** because it uses java language.



**2.2 Difference between Servlet and CGI**

| **Servlet** |  | **CGI(Common Gateway Interface)** |
| --- | --- | --- |
| Servlets are portable and efficient. |  | CGI is not portable |
| In Servlets, sharing data is possible. |  | In CGI, sharing data is not possible. |
| Servlets can directly communicate with the webserver. |  | CGI cannot directly communicate with the webserver. |
| Servlets are less expensive than CGI. |  | CGI is more expensive than Servlets. |
| Servlets can handle the cookies. |  | CGI cannot handle the cookies. |

**2.3 Life Cycle of a Servlet (Servlet Life Cycle)**

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.



1. Servlet instance is created.



1. init method is invoked.

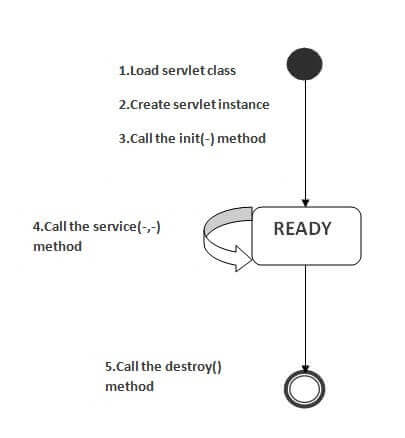


1. service method is invoked.



1. destroy method is invoked.





### 1) Servlet class is loaded



The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.



### 2) Servlet instance is created



The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.



### 3) init method is invoked:



|  |
| --- |
| The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below: |

**public** **void** init(ServletConfig config) **throws** ServletException



### 4) service method is invoked:



The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:



**public** **void** service(ServletRequest request, ServletResponse response)



**throws** ServletException, IOException

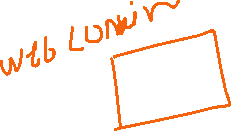
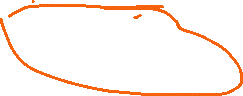


### 5) destroy method is invoked:

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:



**public** **void** destroy()



**2.4 Handling Form Data in Servlets**

Handling client data or request is vital feature of any web application. We all must have used this feature like Google search or filling a contact us page on any website etc.

## 2.4.1 HTTP Request method types

There are several HTTP request method types available like GET, PUT, POST, DELETE, HEAD, OPTIONS etc.

* **GET**- As it names suggests, it is to get a resource or data from server.
* **POST**- Submit or post data to server for processing.
* **DELETE**- To delete any resource from server.
* **HEAD**- It is same as GET but in a response, it just returns Header
* **PUT**- To upload data on server.
* **OPTIONS**- To know the methods supported by server.

Out of these HTTP Request method types, most commonly used methods are GET and POST so let’s discuss these methods in detail.

## 2.4.1.1 GET METHOD

Ideally this method should be used to get the data or resource from server but with GET method type we can send data to the server as well. There are several things which are important to understand about GET method.



* Data is submitted to the server using query parameter like “<http://localhost:8080/HelloWorld/hello?myParam=myValue>”
* As mentioned in above point , data is submitted in query param so it should never be used for sending sensitive information like passwords as data is clearly be seen in browser address bar.



* GET requests have length restrictions which means there is a limit on the size of data that can be submitted to server.
* Maximum length of URL can be 2048 characters.
* GET is default method type.
* GET request supports only ASCII characters as data .
* GET request invokes doGet() method of HttpServlet



## 

## 2.4.1.2 POST METHOD

This method is intended to submit the data to the server for processing. Some Important points about POST method type are –

* Data submitted with POST method type is sent in message body so it is secure and cannot be seen in browser address bar.
* There is no limit on the data that can be sent through POST
* POST request cannot be cached.
* POST request supports binary data as well. We can upload file to the server using POST type.
* POST request invokes doPost() method of HttpServlet.

## 2.5 Get vs. Post

There are many differences between the Get and Post request. Let's see these differences:

|  |  |
| --- | --- |
| **GET** | **POST** |
| 1) In case of Get request, only **limited amount of data** can be sent because data is sent in header. | In case of post request, **large amount of data** can be sent because data is sent in body. |
| 2) Get request is **not secured** because data is exposed in URL bar. | Post request is **secured** because data is not exposed in URL bar. |
| 3) Get request **can be bookmarked.** | Post request **cannot be bookmarked.** |
| 4) Get request is **idempotent** . It means second request will be ignored until response of first request is delivered | Post request is **non-idempotent.** |
| 5) Get request is **more efficient** and used more than Post. | Post request is **less efficient** and used less than get. |

**3.installation and configure apache tomcat web server**

<https://phoenixnap.com/kb/install-tomcat-windows> (All the details you will find over here)

# 4. 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. There are four types of JDBC drivers:



* JDBC-ODBC Bridge Driver,



* Native Driver,



* Network Protocol Driver, and



* Thin Driver

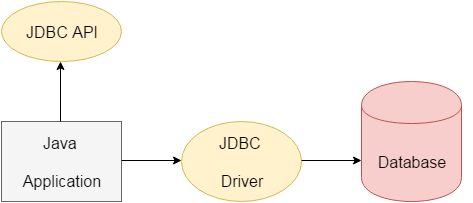


We have discussed the above four drivers in the next chapter.



We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.







The **java.sql** package contains classes and interfaces for JDBC API. A list of popular *interfaces* of JDBC API are given below:



* Driver interface



* Connection interface



* Statement interface



* PreparedStatement interface
* CallableStatement interface
* ResultSet interface
* ResultSetMetaData interface
* DatabaseMetaData interface
* RowSet interface

A list of popular *classes* of JDBC API are given below:

* DriverManager class
* Blob class
* Clob class
* Types class

### 4.1 Why Should We Use JDBC

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).



We can use JDBC API to handle database using Java program and can perform the following activities:



1. Connect to the database



1. Execute queries and update statements to the database



1. Retrieve the result received from the database.



# 4.2 JDBC Driver

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:



1. JDBC-ODBC bridge driver



1. Native-API driver (partially java driver)



1. Network Protocol driver (fully java driver)



1. Thin driver (fully java driver)



### 1) JDBC-ODBC bridge driver

|  |
| --- |
| The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver. |

Oracle does not support the JDBC-ODBC Bridge from Java 8. Oracle recommends that you use JDBC drivers provided by the vendor of your database instead of the JDBC-ODBC Bridge.



### Advantages:

1.easy to use.



2.can be easily connected to any database.



### Disadvantages:

Performance degraded because JDBC method call is converted into the ODBC function calls.



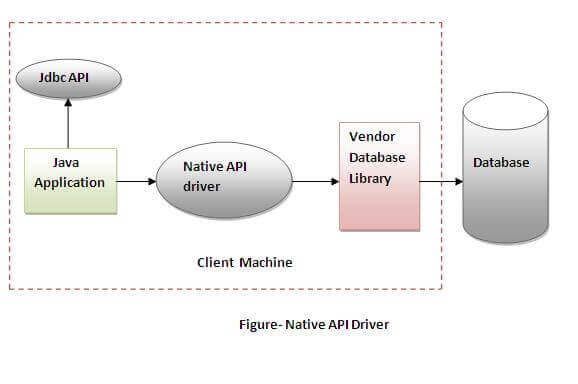
The ODBC driver needs to be installed on the client machine.



### 

### 2) Native-API driver

|  |
| --- |
| The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java. |



### Advantage:

### 1.performance upgraded than JDBC-ODBC bridge driver.



### Disadvantage:

2.The Native driver needs to be installed on the each client machine.



3.The Vendor client library needs to be installed on client machine.

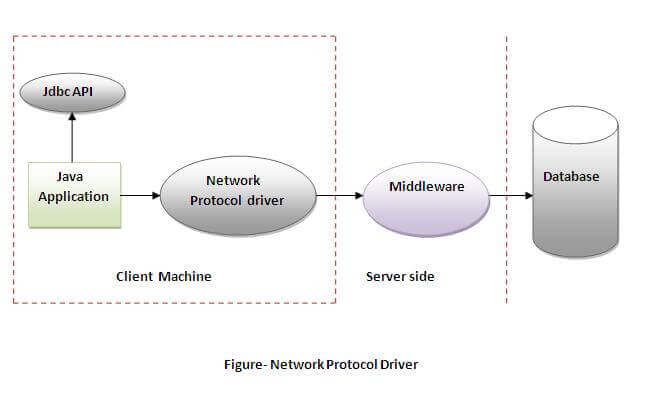


### 

### 3) Network Protocol driver

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.





### Advantage:

* No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.



### Disadvantages:

* Network support is required on client machine.



* Requires database-specific coding to be done in the middle tier.

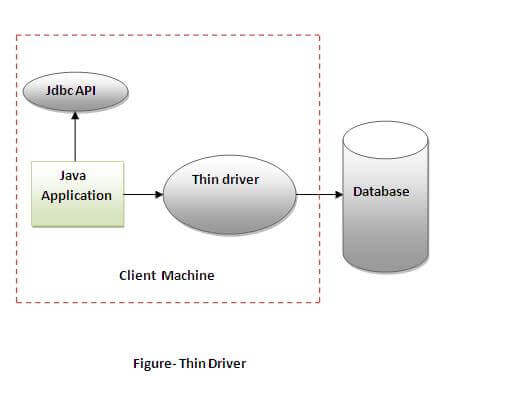


* Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.



### 4) Thin driver

|  |
| --- |
| The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language. |



### 

### Advantage:

1.Better performance than all other drivers.



2.No software is required at client side or server side.



### Disadvantage:

1.Drivers depend on the Database.

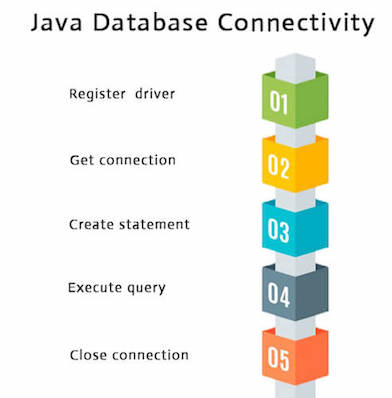


**4.3.Database Connectivity**

# 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:

* Register the Driver class
* Create connection
* Create statement
* Execute queries
* Close connection



### 

### 

### 1) Register the driver class

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

### Syntax of forName() method

1. **public** **static** **void** forName(String className)**throws** ClassNotFoundException

### Example to register the OracleDriver class

Here, Java program is loading oracle driver to establish database connection.

1. Class.forName("oracle.jdbc.driver.OracleDriver");

### 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 Oracle database

### Connection con=DriverManager.getConnection(

"jdbc:oracle:thin:@localhost:1521:xe","system","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

**public** Statement createStatement()**throws** SQLException

### Example to create the statement object

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();

# 

# 5. JSP

**5.1 -INTRODUCTION**

* It stands for **Java Server Pages**.
* It is a server side technology.



* It is used for creating web application.



* It is used to create dynamic web content.



* In this JSP tags are used to insert JAVA code into HTML pages.



* It is an advanced version of Servlet Technology.



* It is a Web based technology helps us to create dynamic and platform independent web pages.



* In this, Java code can be inserted in HTML/ XML pages or both.



* JSP is first converted into servlet by JSP container before processing the client’s request.



5.2 **Features of JSP**

* **Coding in JSP is easy** :- As it is just adding JAVA code to HTML/XML.



* **Reduction in the length of Code** :- In JSP we use action tags, custom tags etc.



* **Connection to Database is easier** :-It is easier to connect website to database and allows to read or write data easily to the database.



* **Make Interactive websites** :- In this we can create dynamic web pages which helps user to interact in real time environment.



* **Portable, Powerful, flexible and easy to maintain** :- as these are browser and server independent.



* **No Redeployment and No Re-Compilation** :- It is dynamic, secure and platform independent so no need to re-compilation.



* **Extension to Servlet** :- as it has all features of servlets, implicit objects and custom tags.



**5.3 Advantages of using JSP**



* + **It does not require advanced knowledge of JAVA**



* + **It is capable of handling exceptions**



* + **Easy to use and learn**



* + **It can tags which are easy to use and understand**



* + **Implicit objects are there which reduces the length of code**



* + **It is suitable for both JAVA and non JAVA programmer**



**5.4 Disadvantages of using JSP**

* + **Difficult to debug for errors.**



* + **First time access leads to wastage of time**



* + **It’s output is HTML which lacks features.**



**5.5 Process of Execution**Steps for Execution of JSP are following:-

* Create html page from where request will be sent to server eg try.html.



* To handle to request of user next is to create .jsp file Eg. new.jsp



* 3.Create project folder structure.



* 4.Create XML file eg my.xml.



* 5. Create WAR file.



* 6.Start Tomcat
* 7. Run Application



**5.6 The Lifecycle of a JSP Page**

The JSP pages follow these phases:



* Translation of JSP page to Servlet



* Compilation of JSP page(Compilation of JSP into test.java)



* Classloading (test.java to test.class)



* Instantiation(Object of the generated Servlet is created)



* Initialization(jspInit() method is invoked by the container)



* Request processing(\_jspService()is invoked by the container)



* JSP Cleanup (jspDestroy() method is invoked by the container)



1. **Translation of JSP page to Servlet:**



This is the first step of the JSP life cycle. This translation phase deals with the Syntactic correctness of JSP. Here test.jsp file is translated to test.java.



1. **Compilation of JSP page:**

Here the generated java servlet file (test.java) is compiled to a class file (test.class).



1. **Classloading:**

Servlet class which has been loaded from the JSP source is now loaded into the container.



1. **Instantiation:**

Here an instance of the class is generated. The container manages one or more instances by providing responses to requests.



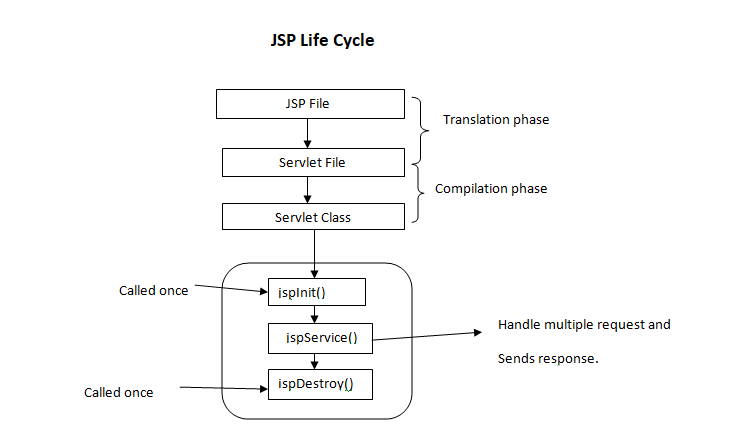
1. **Initialization:**

jspInit() method is called only once during the life cycle immediately after the generation of Servlet instance from JSP.



1. **Request processing:**



\_jspService() method is used to serve the raised requests by JSP. It takes request and response objects as parameters. This method cannot be overridden.



1. **JSP Cleanup :**

In order to remove the JSP from the use by the container or to destroy the method for servlets jspDestroy()method is used. This method is called once, if you need to perform any cleanup task like closing open files, releasing database connections jspDestroy() can be overridden.



**5.7 Example of Hello World**We will make one .html file and .jsp file  
**demo.jsp**

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Hello World - JSP tutorial</title>



</head>

<body>



<%= "Hello World!" %>



</body>

</html>

**5.8 What is JSP Page?**

* A JSP page is a normal web page with JSP elements for generating the parts of the web page that differ for each request.



* A simple JSP web page that contains the JSP elements and template text.



* Everything on the page that is not a JSP element is called Template text.



* Template text can be any text, i.e., HTML, XML, WML or even plain text.JSP has no dependency on HTML, it can be used with any markup language.



* Template text is always passed straight to the browser.



**5.8.1 JSP Elements**

* There are three types of JSP elements present:
  + Directive
  + Action
  + Scripting

### 



### **1. Directive Elements**



**page: <%@ page … %>**

* Defines page-dependent attributes, such as session tracking, error page, and buffering requirements.
* is used for importing a package.
* is used for Handling an Exception.

### **2. Action Tag/Element**

JSP Action tags or Elements are used to perform some specific tasks. The action tags are used to control the flow between pages and to use. There are many JSP action tags.

* **jsp:forward:** forwards the request and response to another resource.
* **jsp:include:** includes another resource.
* **jsp:useBean:** creates or locates bean object.
* **jsp:setProperty:** sets the value of property in bean object.
* j**sp:getProperty:** prints the value of the property of the bean.
* **jsp:plugin:** embeds another component such as applet.
* **jsp:param:** sets the parameter value. It is used in forward and includes mostly.
* **jsp:fallback:** can be used to print the message if the plugin is working. It is used in jsp:plugin.

### 

### 

### **3. Scripting Element**

Scriptlet elements must be written within the <% … %> tags.The Scriptlet tag allows writing Java code statements within the JSP page.The tag is responsible for implementing the functionality of “\_jspService()” by scripting the java code.The JSP engine will process any code written within the pair of <% and %> tags and any other code is treated as plain text while translating the JSP page. There are three main subdivisions of Scripting Elements in Java Server Pages.

* Expression Tag
* Scriptlet Tag
* Declaration Tag