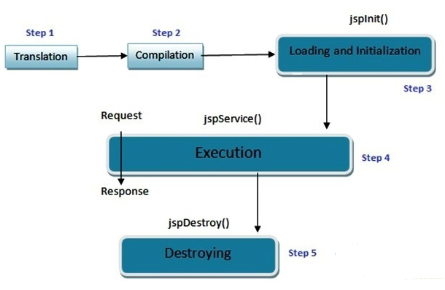
**Java Server Pages**

JSP or Java Server Pages is used to build or create dynamic web pages. By using a JSP tags, a java code can be added in the HTML pages. The use of the tags is to retrieve or restore the data from the database, sharing information from one page to another etc. NetBeansIDE and EclipseIDE are the commonly used IDE’s used for JSP pages.

**1 Life Cycle of JSP**



**1.1 Translation**

The first step, JSP document is translated by a web container into a corresponding Java code. The servlet in this is the java code. It is automatically translated by the Web server which validates, locates the correctness and automatically write the servlet on the JSP page.

**1.2 Compilation**

It compiles the java source by the JSP container in order to generate the class file and create the equivalent servlet.

**1.3 Loading and Initialization**

It will load the servlet by the JSP container of the two previous stages. After loading it properly, the JSP container will create an instance of the servlet class.

**1.4 Execution**

The *\_jspService()* method is created by the JSP engine. The *\_jspService()* method has 2 parameters. HttpservletRequest and HttpServletResponse is created once per request.



**1.5 Destroying**

In this last step it completes the life cycle of JSP. By using the *jspDestroy()* method the JSP is removed by the container.

**2 Java Server Page Demonstration**

Below is an example of JSP use. Take after these steps to begin with your first JSP Application using the EclipseIDE. Technologies used :

* EclipseIDE
* JSP
* Java 1.8
* Apache Tomcat 8

**Step 1:**

Click *File >> New >> Dynamic Web Project*

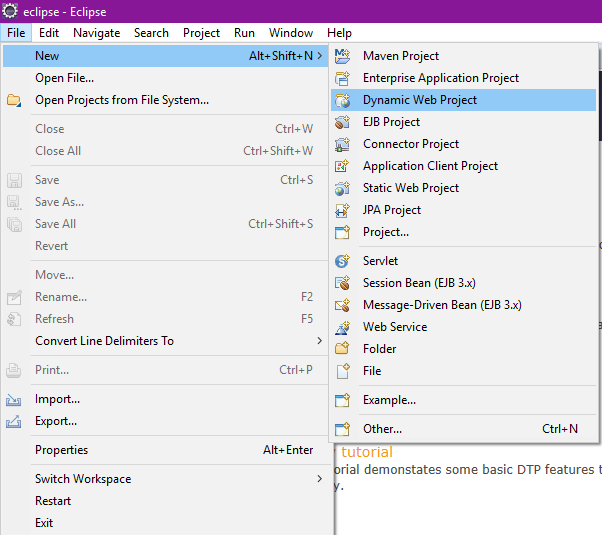


Figure 1: Creating a new Dynamic Web Project

**Step 2:**

Write *WebRankApplication* as the project name and choose your target runtime e.i *Apache Tomcat v9.0 .*Then click *Next.*

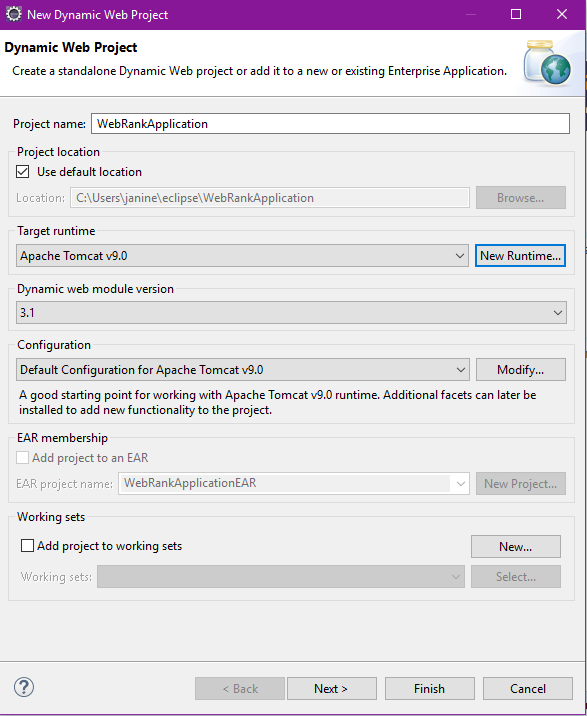


Figure 2: Adding Project

**Step 3:**

Skip or avoid this step for configuring project and just click the *Next* button as we are concentrating on a straightforward demo.

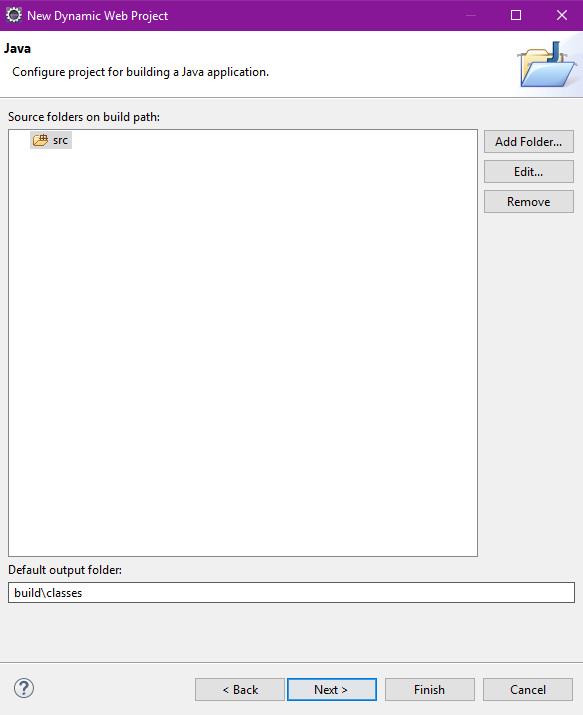


Figure 3: Configure a project

**Step 4:**

Abstain from generating web.xml file as it isn’t required in this demo.

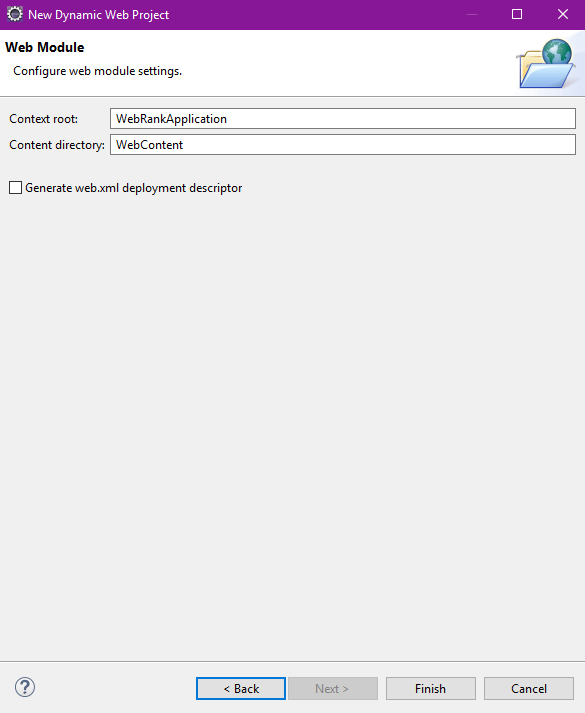


Figure 4: Web Module: Configuring Web module settings

**Step 5:**

After clicking the *Finish* button, the dynamic web project gets made or created. The content directory is visible on the left board or panel under “Package Explorer”.

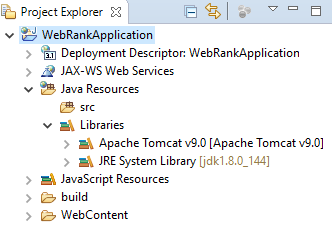


Figure 5: Project Directory Structure

**Step 6:**

On the *WebContent* folder, right click it then *New >> JSP File*

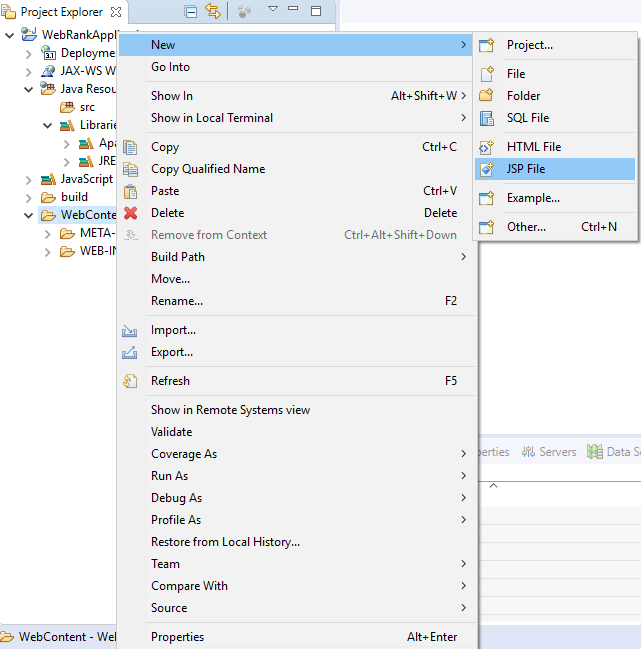


Figure 6: Adding JSP file in the dynamic web project

**Step 7:**

A new window will appear, on the File name, name a *NewRankFile.jsp* then click *Next.*

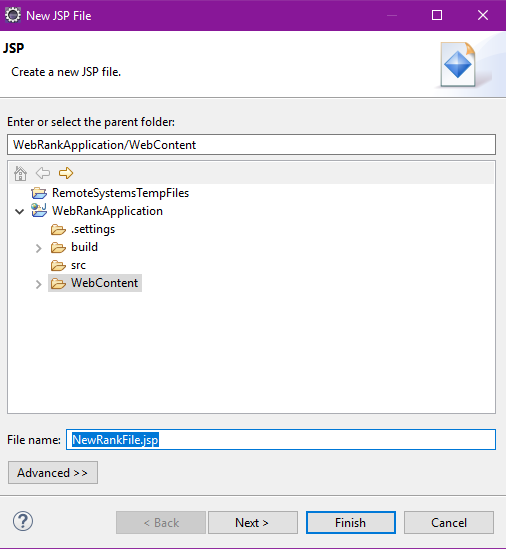


Figure 7: Creating and naming a new JSP file

**Step 8:**

Check the textbox “*Use JSP Template”.* To set the JSP template.

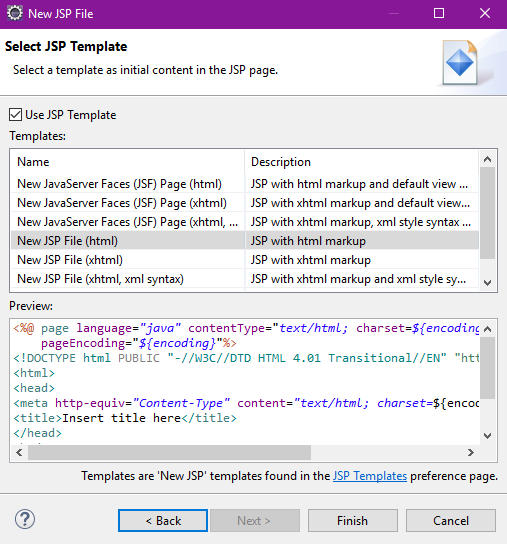


Figure 8: Setting the JSP Template

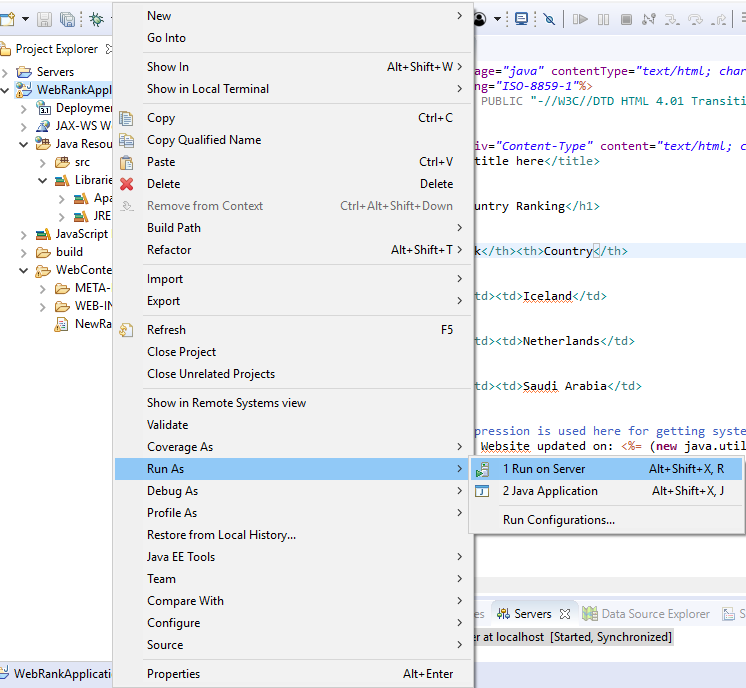
**Step 9:**

It will now create a new file. Then add given code below which will prints the Richest country ranking.



**Step 10:**

Right click the “*WebRankApplication” >> Run as >> Run on Server.*

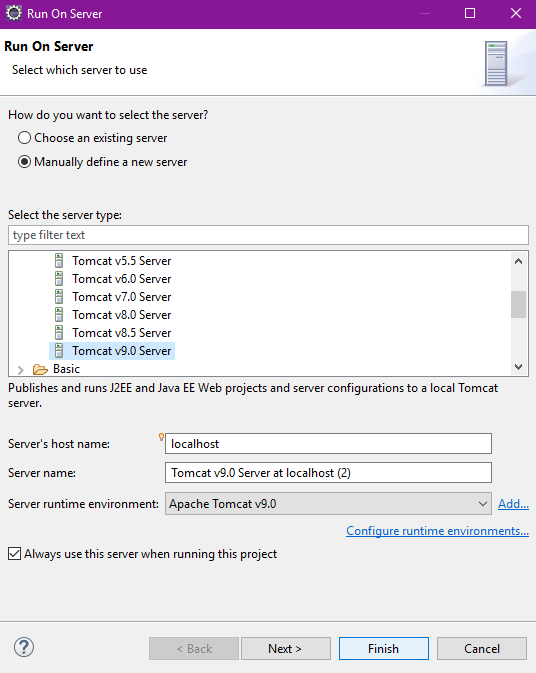


**Step 11:**

After clicking the “*Run on Server”,* a new window will appear and click the “*Manually define a new*

*server” Tomcat v9.0 server.* And don’t forget to check the “*Always use this server while running this*

*project”* to avoidredundancy.



**Step 12:**

Click the “*Finish”* button. Then the *Richest Country Ranking* website will run. The website is updating time because of using JSP Expressions. Link: <http://localhost:8080/WebRankApplication/NewRankFile.jsp>

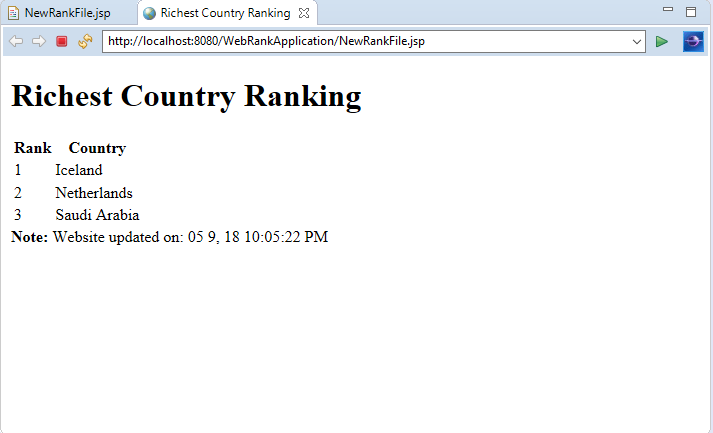


Figure 9: Displaying the *Richest Country Ranking* Website with updating time.

**4 Directives**

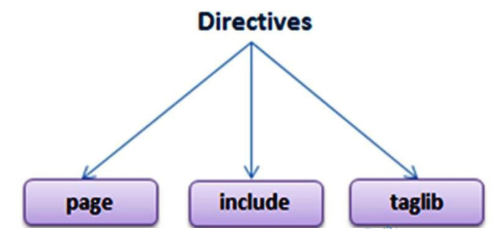
Directive tags are utilized to give directions and guidelines used at the interpretation stage of JSP life

cycle.

**4.1 Syntax**



**4.2 Types**



**4.2.1 Page directives**

The page directive tag gives the guidelines utilized by the translator at the phase or stage of the life

cycle of JSP. It can be incorporated anywhere, however according to convention it is considered as a

decent programming style of including at the top.

Syntax:



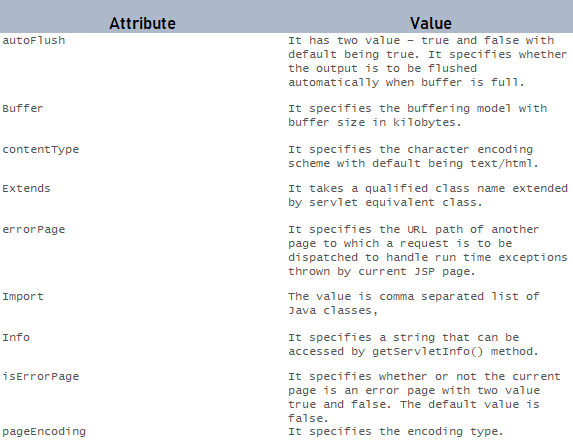
XML equivalent:



With attribute as import:



Attribute utilized in the page directive tag:



**4.2.2 Include Directives**

This tag is utilized amid the translation phase of JSP lifecycle to incorporate a file. It combines the

content of more than 2 files. Incorporate directives can be incorporated anyplace in the page.

Syntax:



**4.2.3 Taglib**

It is utilized to characterize a custom tag library in a JSP page. This is done as such that the related tags

can be utilized or used as a part of a similar page.

Syntax:



**5 Scripting Tags**

JSP scripting tags permit adding script code into the java code of a created JSP page. This page is created

by the JSP translator.

**3 Types of Scriptlet tags:**

* Scriplet Tag - it is a tag that implements the *\_jspService* method by composing script/java code. It is utilized for composing java code in JSP page.
* Declarative Tag – it is utilized to declare class variables and executing class methods *jspInt and jspDestroy.*
* Expression Tag – it is utilized to compose a java expression. Place the semi colon (;) inside an expression tag.

**6 Expression Language**

A language that permits developers of JSP for getting to application data stored in JavaBeans

components. It was presented in JSP2.0.

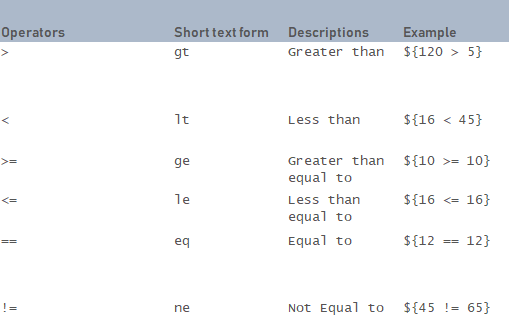
The EL expressions are encased between “$” and “}” characters.

**6.1 Operators**

**6.1.1 Arithmetic Operators**

An arithmetic operators that can be utilized in EL expressions : -, \_, \*, %

**6.1.2 Relational Operators**



**6.1.3 Logical Operators**

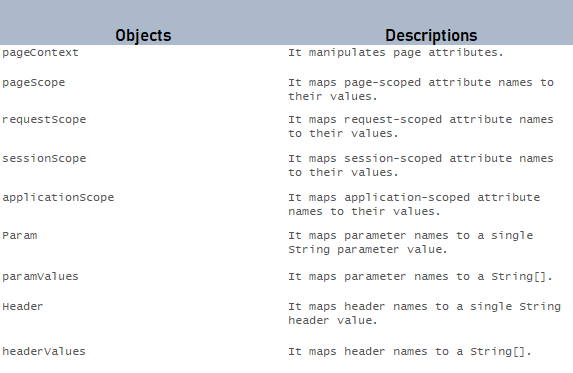
* && (AND Operator)
* || (OR Operator)
* ! (NOT Operator)

**6.2 Implicit Objects**

The implicit objects can straightforwardly be utilized as a part of an El expressions. Users can utilized

these objects to get attributes from various parameter values and scopes.

Types of implicit objects:



**7 Exception Handling**

An exception is an abnormal or unexpected condition in the ordinary execution flow of a program. These

exception may happen because of invalid information, getting to inaccessible files on desk and so forth.

**7.1.1 Errors**

These are the issues which are outside the ability to control users.

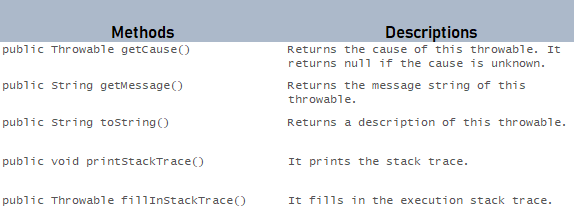
**7.1.2 Checked Exceptions**

Exceptions which can’t be ignored at the compilation and is considered as user mistake.

**7.1.3 Runtime Exceptions or Unchecked Exceptions**

These exceptions can be overlooked at runtime.

**7.2 Methods**



**7.3 ErrorPage and isErrorPage**

ErrorPage attribute of directive’s page is utilized for setting up an error page.

isErrorPage attribute is utilized for creating the exception instance variable by including the

accompanying order.

**8 Internationalization**

Web Application are gotten to by individuals everywhere throughout the world. This ascents the need of

Internalization. Internationalization empowers a web application to work in various nations and districts

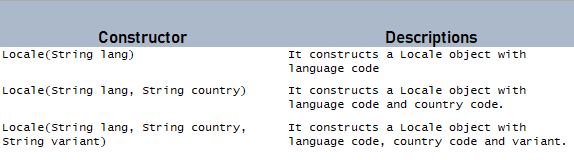
without recompiling and change to the code.

**8.1 Locale Class**

Uses *java.util.Locale.class*

Locale defines geographic region and language.

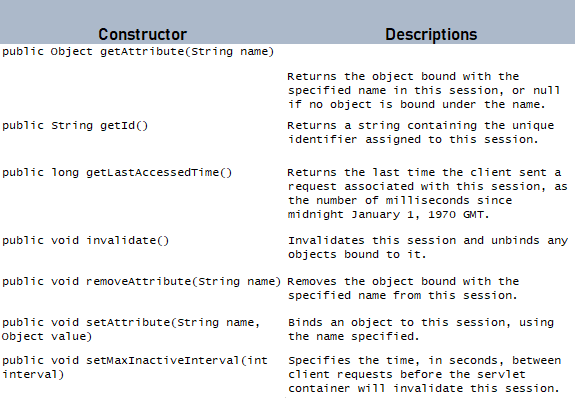
**8.1.1 Constructors of Locale Class**



**9 Sessions**

Session is a gathering or collection of Http request between client and server.

**9.1 Methods**



**10 JavaBean**

JavaBeans are Java classes that are written in java for creating dynamic content. It isolates business

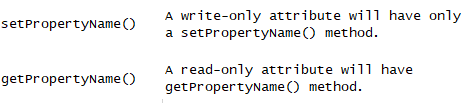
rationale from introduction logic.

Presentation code and business rationale can be overseen independently. JavaBeans also guarantee c

communication between them.

**10.1 JavaBean Properties**

JavaBean property can be an attribute with any datatype.



**10.2 Attributes of Tag**

**10.2.1 id**

It represents the variable name relegated to the id attribute of and is utilized to locate existing bean

instance.

**10.2.2 scope**

It represents the scope in which the bean instance must be located.

* page scope
* request scope
* session scope
* application scope

**10.2.3 class**

It is a class name for making instance of bean, yet the class ought not be an abstract class.

**10.2.4 beanName**

It takes a qualified expression or class name.

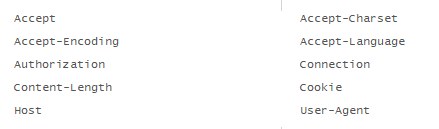
**10.2.5 type**

It takes a qualified interface or class name.

**11 Request and Response Objects**

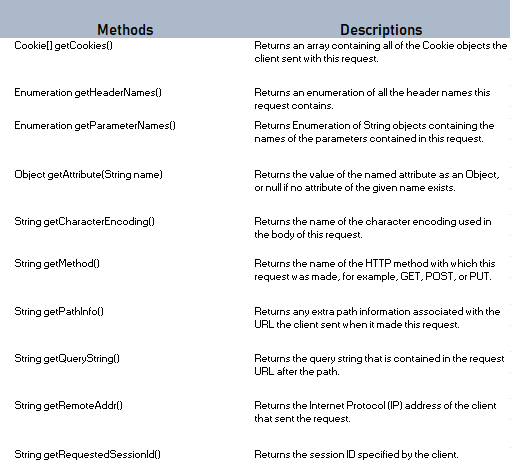
**11.1 Request**

A Web page is asked for by a web browser to send data or information to the web server.



**11.2 HttpServletRequest**

It is an instance of javax.servlet.http.HttpServletRequest object**.**



**11.3 HttpServletResponse**

It is an instance of javax.servlet.http.HttpServletResponse object.

