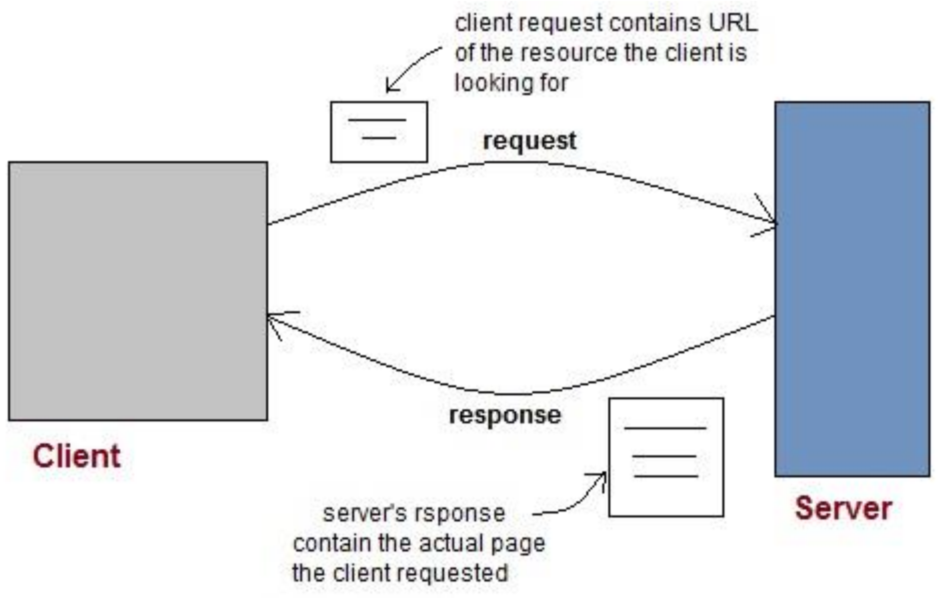
SERVLET TECHNOLOGY

Introduction to Web

Web consists of billions of clients and server connected through wires and wireless networks. The web clients make requests to web server. The web server receives the request, finds the resources and return the response to the client. When a server answers a request, it usually sends some type of content to the client. The client uses web browser to send request to the server. The server often sends response to the browser with a set of instructions written in HTML(HyperText Markup Language). All browsers know how to display HTML page to the client.


client request contains URL of the resource the client is looking for
request
response
Client
Server
server's rsponse contain the actual page the client requested

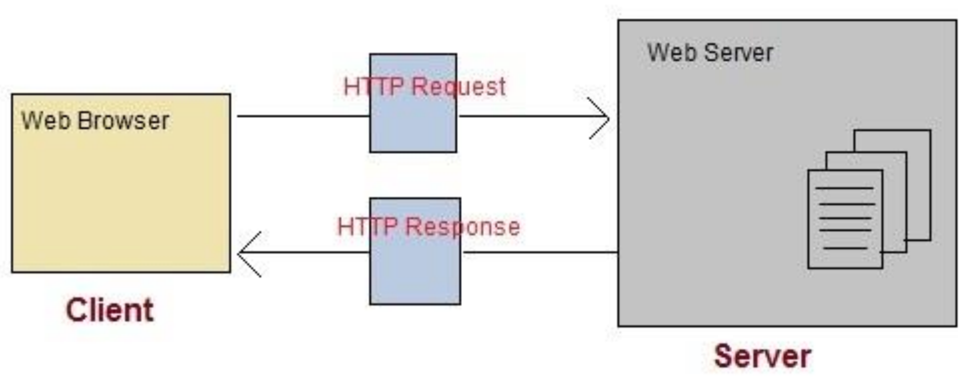
---

Web Application

A website is a collection of static files(webpages) such as HTML pages, images, graphics etc.
A **Web application** is a web site with dynamic functionality on the server. **Google**, **Facebook**, **Twitter** are examples of web applications.

HTTP (Hypertext Transfer Protocol)

- HTTP is a protocol that clients and servers use on the web to communicate.
- It is similar to other internet protocols such as SMTP(Simple Mail Transfer Protocol) and FTP(File Transfer Protocol) but there is one fundamental difference.
- HTTP is a **stateless protocol** i.e HTTP supports only one request per connection. This means that with HTTP the clients connect to the server to send one request and then disconnects. This mechanism allows more users to connect to a given server over a period of time.
- The client sends an HTTP request and the server answers with an HTML page to the client, using HTTP.



HTTP Methods

HTTP request can be made using a variety of methods, but the ones you will use most often are **Get** and **Post**. The method name tells the server the kind of request that is being made, and how the rest of the message will be formated.

HTTP Methods and Descriptions :

Method Name	Description
OPTIONS	Request for communication options that are available on the request/response chain.
GET	Request to retrieve information from server using a given URI.
HEAD	Identical to GET except that it does not return a message-body, only the headers and status line.
POST	Request for server to accept the entity enclosed in the body of HTTP method.
DELETE	Request for the Server to delete the resource.
CONNECT	Reserved for use with a proxy that can switch to being a tunnel.
PUT	This is same as POST, but POST is used to create, PUT can be used to create as well as update. It replaces all current representations of the target resource with the uploaded content.

Difference between GET and POST requests

GET Request	POST Request
Data is sent in header to the server	Data is sent in the request body
Get request can send only limited amount of data	Large amount of data can be sent.

GET Request	POST Request
Get request is not secured because data is exposed in URL	Post request is secured because data is not exposed in URL.
Get request can be bookmarked and is more efficient.	Post request cannot be bookmarked.

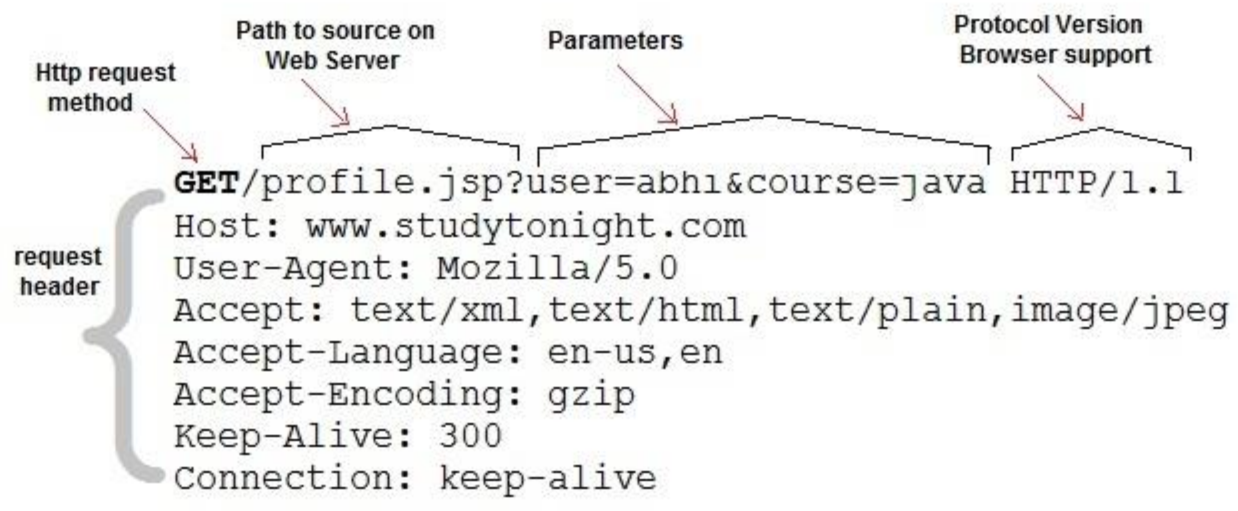
General Difference between PUT and POST methods

Following are some basic differences between the PUT and the POST methods :

- **POST** to a URL creates a child resource at a server defined URL while **PUT** to a URL creates/replaces the resource in its entirety at the client defined URL.
- POST creates a child resource, so POST to `/books` will create a resources that will live under the `/books` resource. Eg. `/books/1`. Sending the same post request twice will create two resources.
- PUT is for creating or replacing a resource at a URL known by the client.
- PUT must be used for CREATE when the client already knows the url before the resource is created.
- PUT replaces the resource at the known url if it already exists, so sending the same request twice has no effect. In other words, calls to PUT are **idempotent.**

Anatomy of an HTTP GET request

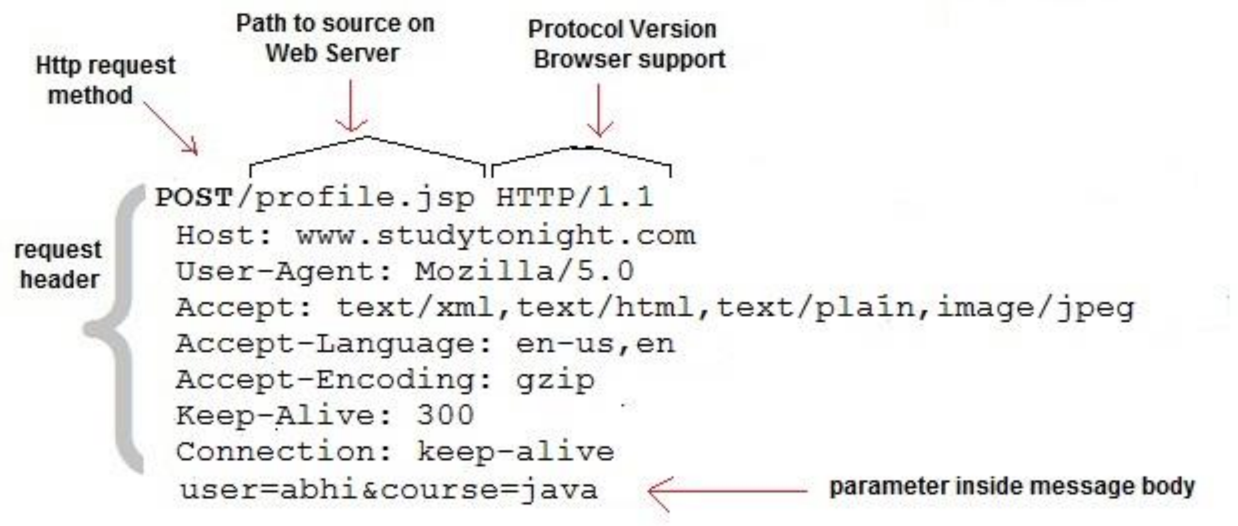
Get request contains path to server and the parameters added to it.


Http request method
Path to source on Web Server
Parameters
Protocol Version Browser support
request header

```
GET/profile.jsp?user=abhi&course=java HTTP/1.1
Host: www.studytonight.com
User-Agent: Mozilla/5.0
Accept: text/xml,text/html,text/plain,image/jpeg
Accept-Language: en-us,en
Accept-Encoding: gzip
Keep-Alive: 300
Connection: keep-alive
```

Anatomy of an HTTP POST request

Post requests are used to make more complex requests on the server. For instance, if a user has filled a form with multiple fields and the application wants to save all the form data to the database. Then the form data will be sent to the server in POST request body, which is also known as Message body.

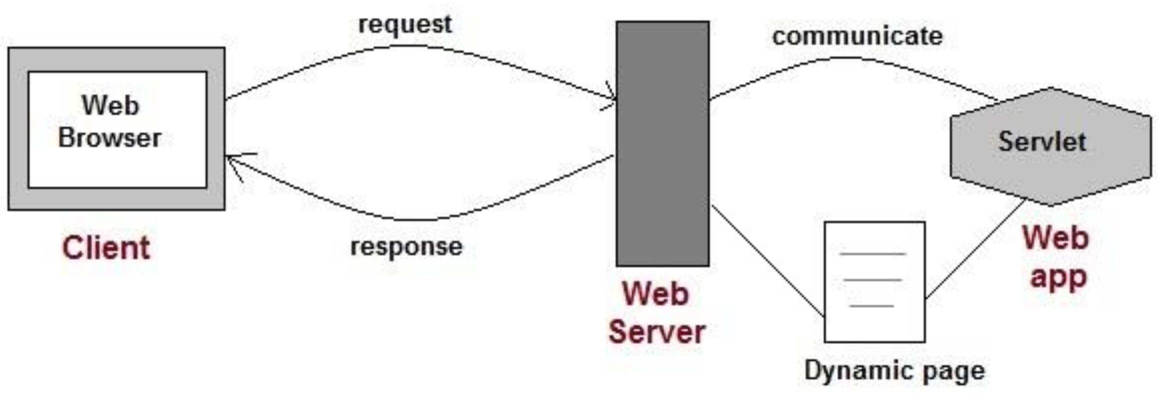

Http request method
Path to source on Web Server
Protocol Version Browser support
request header
parameter inside message body

```
POST/profile.jsp HTTP/1.1
 Host: www.studytonight.com
 User-Agent: Mozilla/5.0
 Accept: text/xml,text/html,text/plain,image/jpeg
 Accept-Language: en-us,en
 Accept-Encoding: gzip
 Keep-Alive: 300
 Connection: keep-alive
 user=abhi&course=java
```

Introduction to Servlet

Servlet Technology is used to create web applications. **Servlet** technology uses Java language to create web applications.

Web applications are helper applications that resides at web server and build dynamic web pages. A dynamic page could be anything like a page that randomly chooses picture to display or even a page that displays the current time.

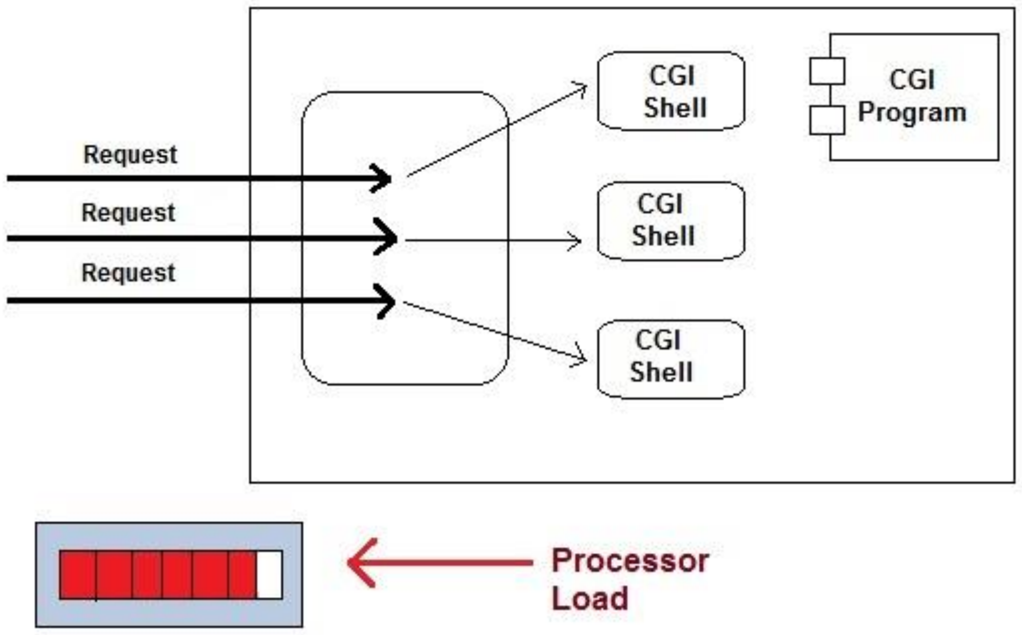


As Servlet Technology uses Java, web applications made using Servlet are **Secured**, **Scalable** and **Robust**.

CGI (Common Gateway Interface)

Before Servlets, CGI(Common Gateway Interface) programming was used to create web applications. Here's how a CGI program works :

- User clicks a link that has URL to a dynamic page instead of a static page.
- The URL decides which CGI program to execute.
- Web Servers run the CGI program in seperate OS shell. The shell includes OS enviroment and the process to execute code of the CGI program.
- The CGI response is sent back to the Web Server, which wraps the response in an HTTP response and send it back to the web browser.



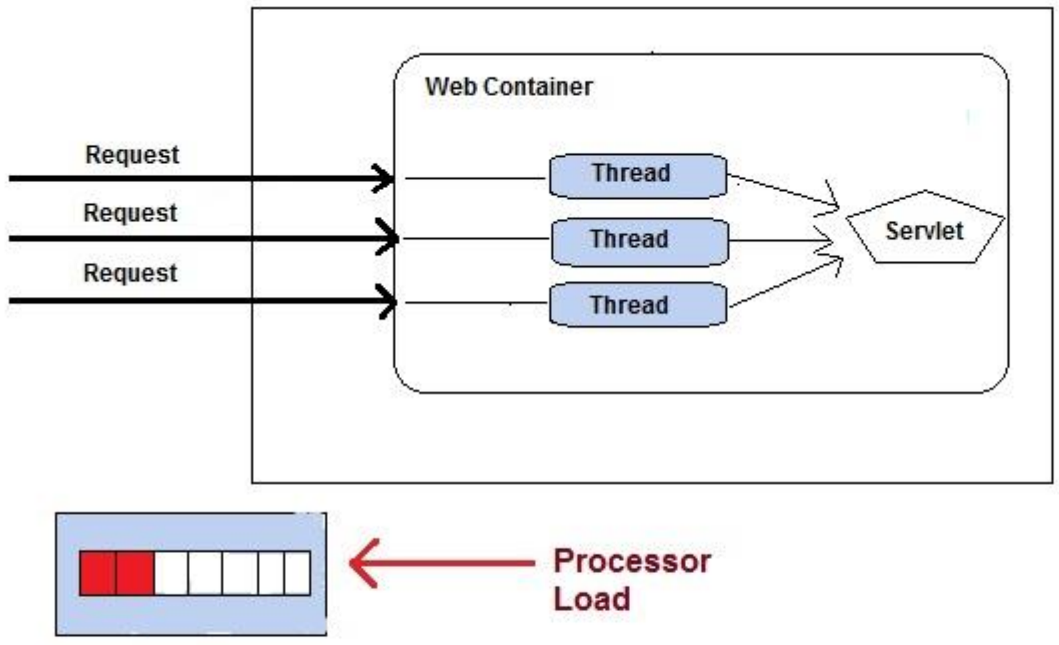
Drawbacks of CGI programs

- High resposne time because CGI programs execute in their own OS shell.
- CGI is not scalable.
- CGI programs are not always secure or object-oriented.
- It is Platform dependent.

Because of these disadvantages, developers started looking for better CGI solutions. And then Sun Microsystems developed **Servlet** as a solution over traditional CGI technology.

Advantages of using Servlets

- Less response time because each request runs in a separate thread.
- Servlets are scalable.
- Servlets are robust and object oriented.
- Servlets are platform independent.



Servlet API

Servlet API consists of two important packages that encapsulates all the important classes and interface, namely :

- **javax.servlet**
- **javax.servlet.http**

Some Important Classes and Interfaces of javax.servlet

INTERFACES	**CLASSES**
Servlet	ServletInputStream
ServletContext	ServletOutputStream
ServletConfig	ServletRequestWrapper

INTERFACES	**CLASSES**
ServletRequest	ServletResponseWrapper
ServletResponse	ServletRequestEvent
ServletContextListener	ServletContextEvent
RequestDispatcher	ServletRequestAttributeEvent
SingleThreadModel	ServletContextAttributeEvent
Filter	ServletException
FilterConfig	UnavailableException
FilterChain	GenericServlet
ServletRequestListener	

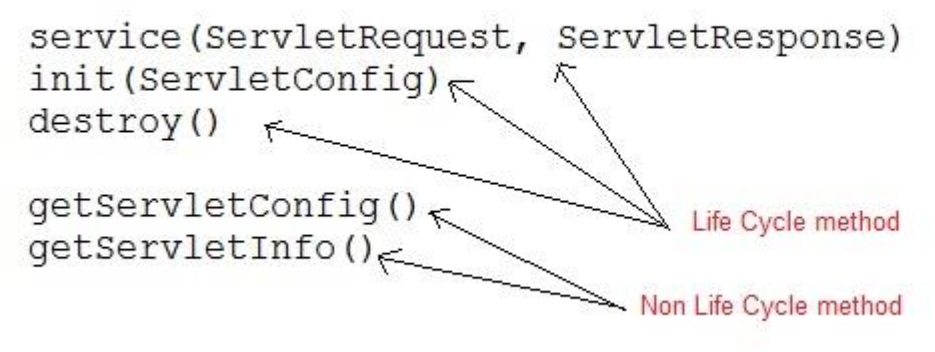
Some Important Classes and Interface of javax.servlet.http

CLASSES and INTERFACES	
HttpServlet	HttpServletRequest
HttpServletResponse	HttpSessionAttributeListener

CLASSES and INTERFACES	
HttpSession	HttpSessionListener
Cookie	HttpSessionEvent

Servlet Interface

Servlet Interface provides five methods. Out of these five methods, three methods are **Servlet life cycle**methods and rest two are non life cycle methods.



GenericServlet Class

GenericServlet is an abstract class that provides implementation of most of the basic servlet methods. This is a very important class.

Methods of GenericServlet class

- `public void init(ServletConfig)`
- `public abstract void service(ServletRequest request,ServletResposne response)`

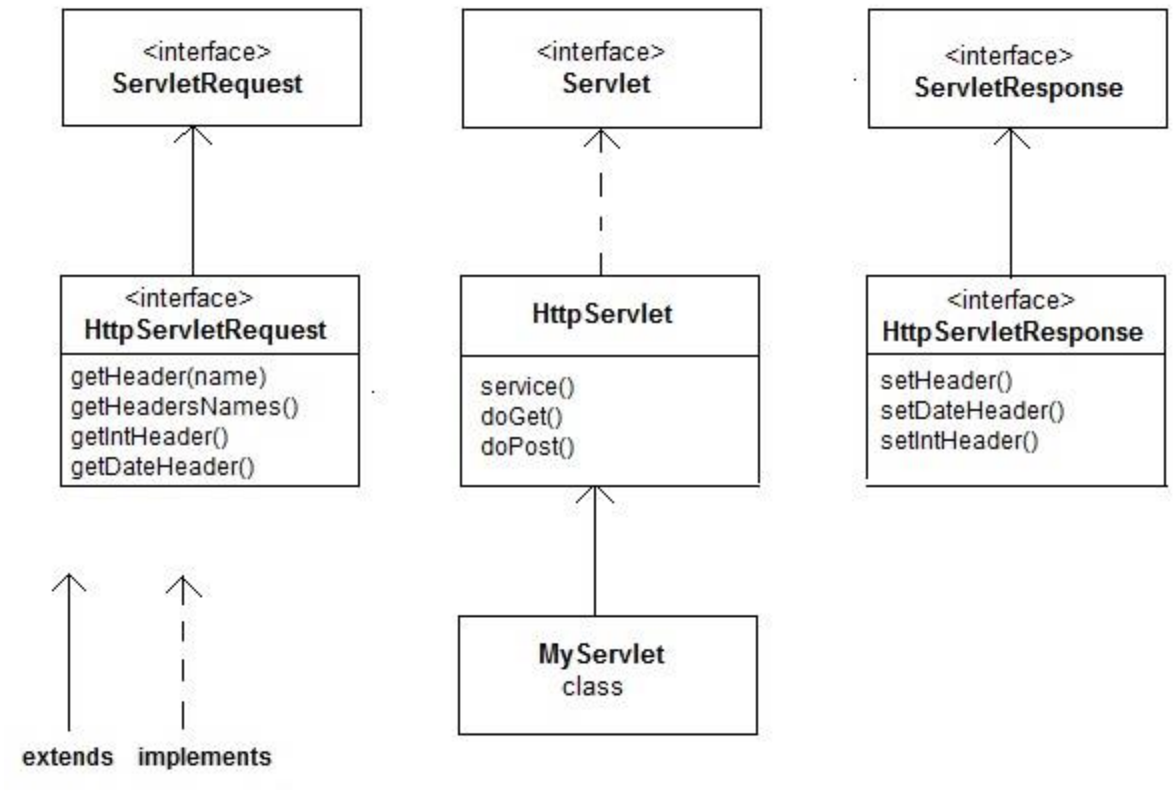
- `public void destroy()`
- `public ServletConfig getServletConfig()`
- `public String getServletInfo()`
- `public ServletContext getServletContext()`
- `public String getInitParameter(String name)`
- `public Enumeration getInitParameterNames()`
- `public String getServletName()`
- `public void log(String msg)`
- `public void log(String msg, Throwable t)`

HttpServlet class

HttpServlet is also an abstract class. This class gives implementation of various `service()` methods of**Servlet** interface.

To create a servlet, we should create a class that extends **HttpServlet** abstract class. The Servlet class that we will create, must not override `service()` method. Our servlet class will override only the `doGet()` and/or `doPost()` methods.

The `service()` method of **HttpServlet** class listens to the Http methods (GET, POST etc) from request stream and invokes `doGet()` or `doPost()` methods based on Http Method type.



How a Servlet Application works

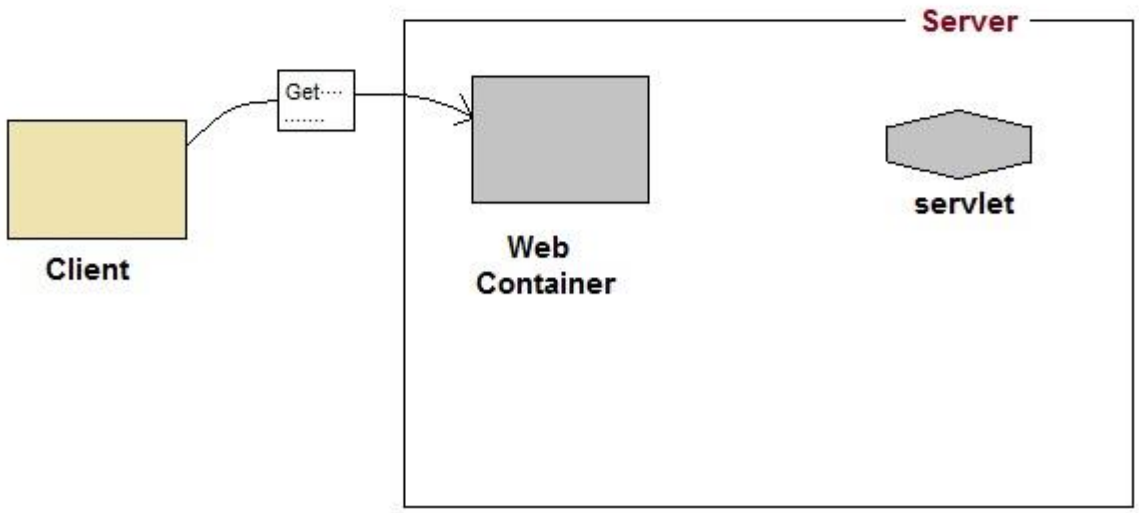
Web container is responsible for managing execution of servlets and JSP pages for Java EE application.

When a request comes in for a servlet, the server hands the request to the Web Container. **Web Container** is responsible for instantiating the servlet or creating a new thread to handle the request. Its the job of Web Container to get the request and response to the servlet. The container creates multiple threads to process multiple requests to a single servlet.

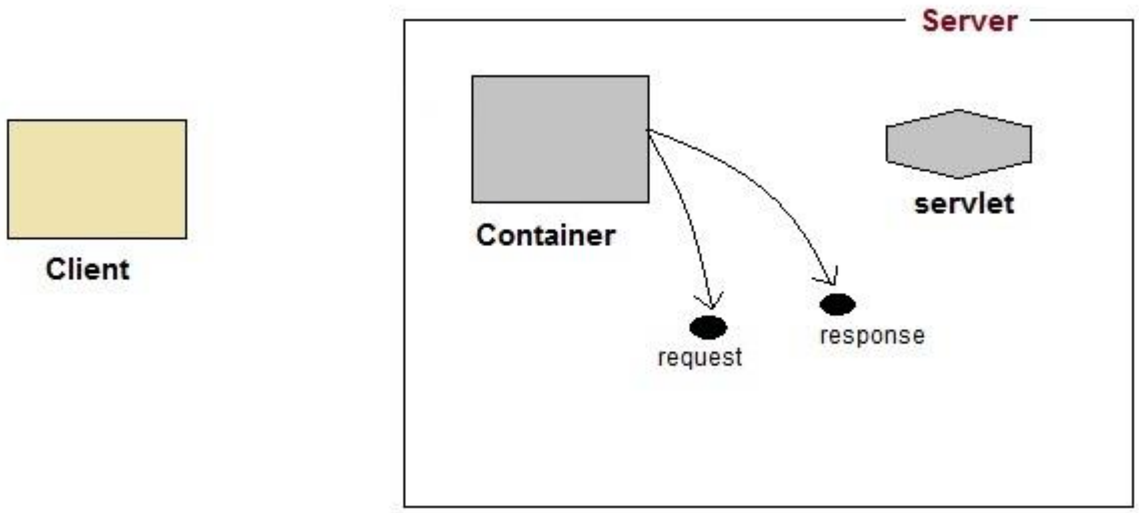
Servlets don't have a main() method. Web Container manages the life cycle of a Servlet instance.

Quick Revision on How a Servlet works

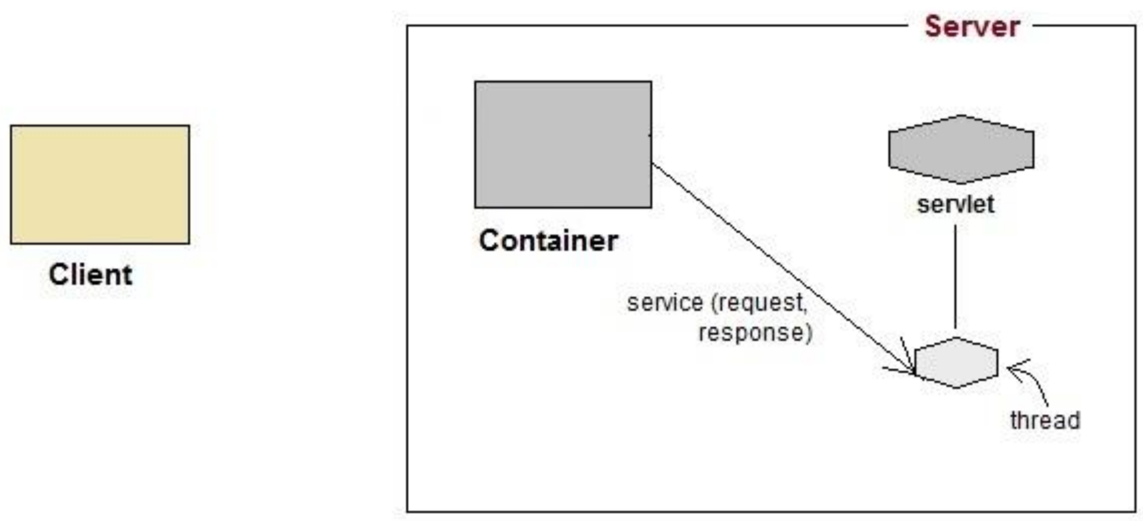
1. User sends request for a servlet by clicking a link that has URL to a servlet.



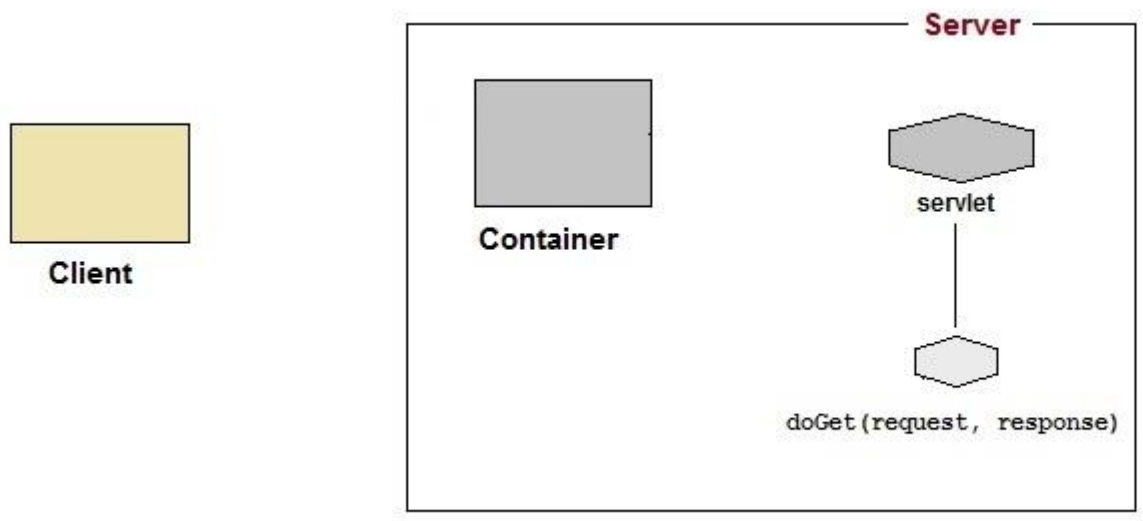
2. The container finds the servlet using **deployment descriptor** and creates two objects :
   a. **HttpServletRequest**
   b. **HttpServletResponse**



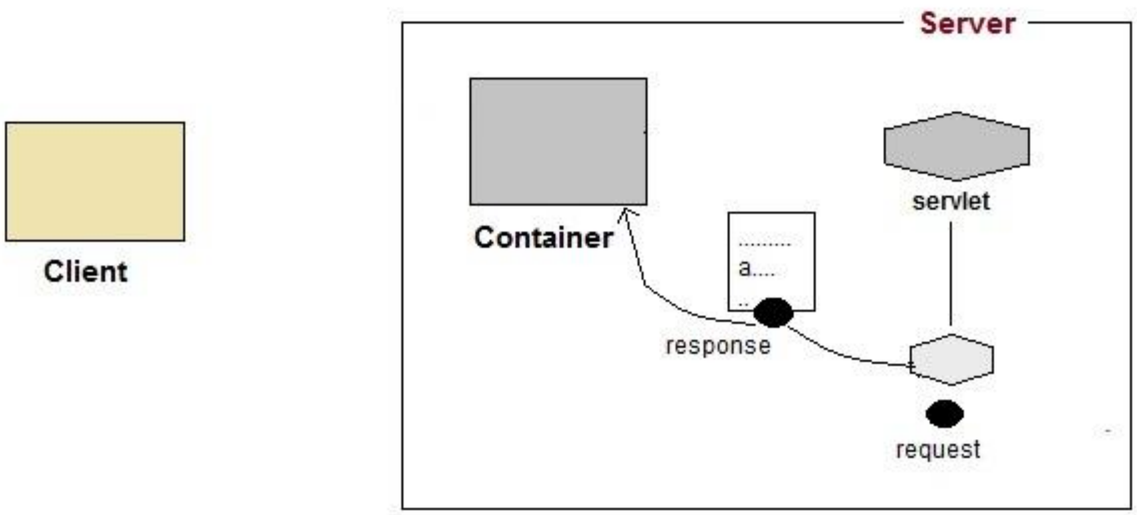
3. Then the container creates or allocates a thread for that request and calls the Servlet's `service()` method and passes the **request, response** objects as arguments.



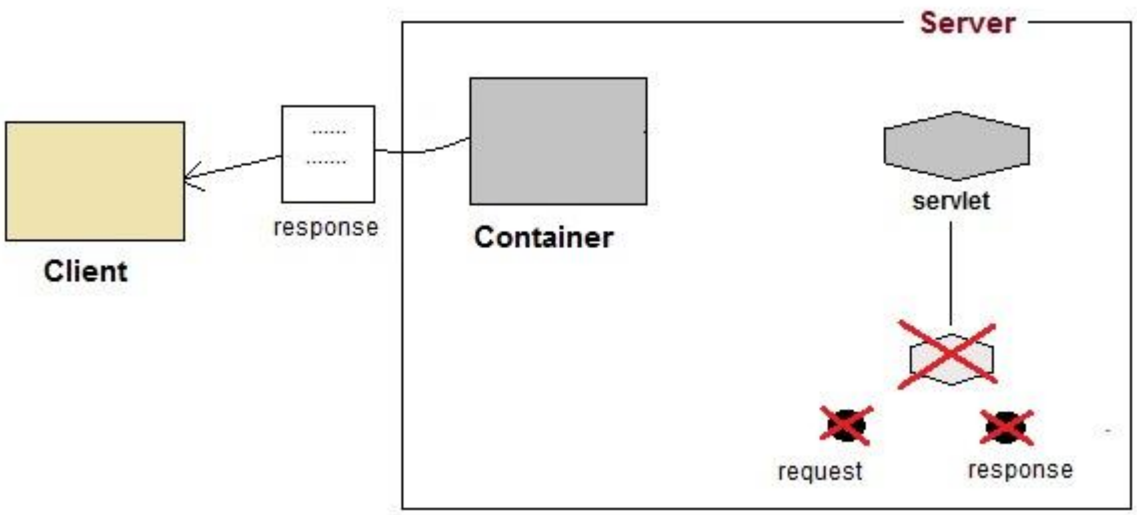
4. The `service()` method, then decides which servlet method, `doGet()` or `doPost()` to call, based on**HTTP Request Method**(Get, Post etc) sent by the client. Suppose the client sent an HTTP GET request, so the `service()` will call Servlet's `doGet()` method.



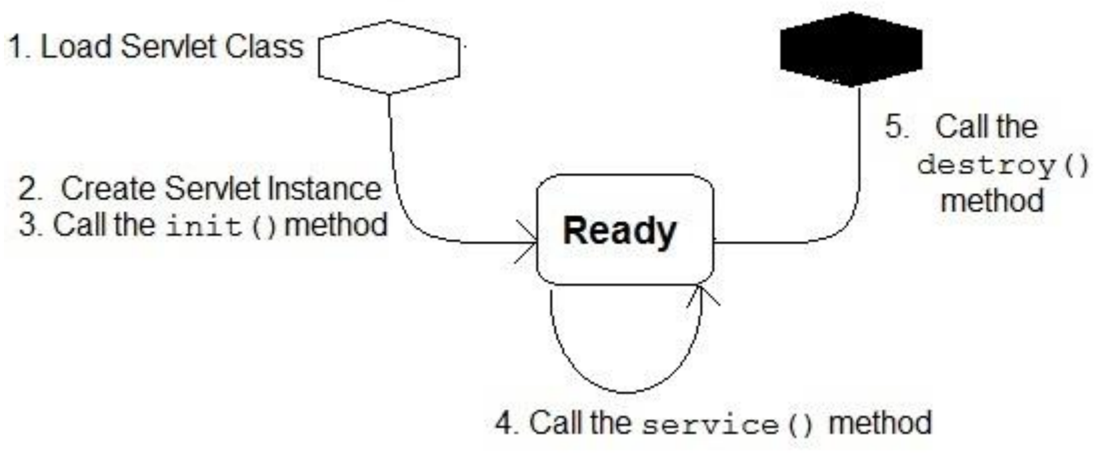
5. Then the Servlet uses response object to write the response back to the client.



6. After the `service()` method is completed the **thread** dies. And the request and response objects are ready for **garbage collection**.



Servlet Life Cycle


1. Load Servlet Class
2. Create Servlet Instance
3. Call the init() method
Ready
5. Call the destroy() method
4. Call the service() method

1. **Loading Servlet Class :** A Servlet class is loaded when first request for the servlet is received by the Web Container.
2. **Servlet instance creation :**After the Servlet class is loaded, Web Container creates the instance of it. Servlet instance is created only once in the life cycle.
3. **Call to the init() method :** `init()` method is called by the Web Container on servlet instance to initialize the servlet.

   Signature of init() method :

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

4. **Call to the service() method :** The containers call the `service()` method each time the request for servlet is received. The service() method will then call the `doGet()` or `doPost()` methos based ont eh type of the HTTP request, as explained in previous lessons.

   Signature of service() method :

   ```
   public void service(ServletRequest request, ServletResponse response) throws
   ServletException, IOException
   ```

5. **Call to destroy() method:** The Web Container call the `destroy()` method before removing servlet instance, giving it a chance for cleanup activity.

Steps to Create Servlet Application using tomcat server

To create a Servlet application you need to follow the below mentioned steps. These steps are common for all the Web server. In our example we are using Apache Tomcat server. Apache Tomcat is an open source web server for testing servlets and JSP technology. Download latest version of Tomcat Server and install it on your machine.

After installing Tomcat Server on your machine follow the below mentioned steps :

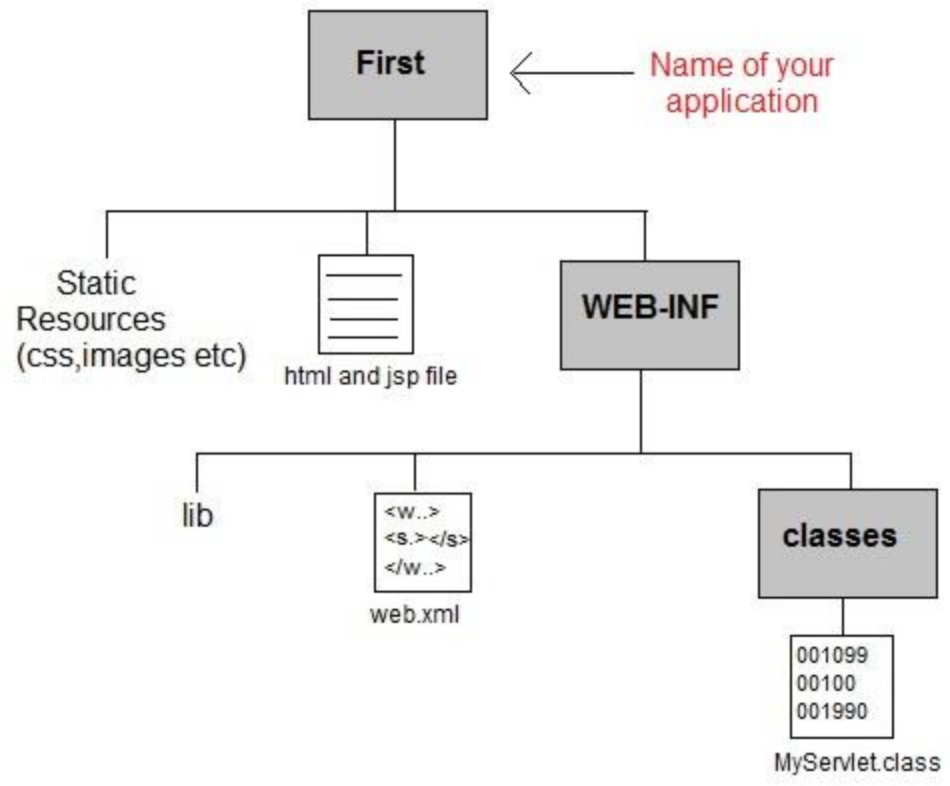
1. Create directory structure for your application.
2. Create a Servlet
3. Compile the Servlet
4. Create Deployement Descriptor for your application
5. Start the server and deploy the application

All these 5 steps are explained in details below, lets create our first Servlet Application.

---

1. Creating the Directory Structure

Sun Microsystem defines a unique directory structure that must be followed to create a servlet application.



Create the above directory structure inside **Apache-Tomcat\webapps** directory. All HTML, static files(images, css etc) are kept directly under **Web application** folder. While all the Servlet classes are kept inside `classes` folder.

The `web.xml` (deployement descriptor) file is kept under `WEB-INF` folder.

Creating a Servlet

There are three different ways to create a servlet.

- By implementing **Servlet** interface
- By extending **GenericServlet** class
- By extending **HttpServlet** class

But mostly a servlet is created by extending **HttpServlet** abstract class. As discussed earlier **HttpServlet** gives the definition of `service()` method of the **Servlet** interface. The servlet class that we will create should not override `service()` method. Our servlet class will override only `doGet()` or `doPost()` method.

When a request comes in for the servlet, the Web Container calls the servlet's `service()` method and depending on the type of request the `service()` method calls either the `doGet()` or `doPost()` method.

NOTE: By default a request is **Get** request.

```
import javax.servlet.*;

import javax.servlet.http.*;

import java.io.*;


public MyServlet extends HttpServlet

{

 public void doGet(HttpServletRequest request,HttpServletResposne response)

                     throws ServletException

 {

   response.setContentType("text/html");

   PrintWriter out = response.getWriter();

   out.println("<html><body>");

   out.println("<h1>Hello Readers</h1>");

   out.println("</body></html>");

 }

}
```

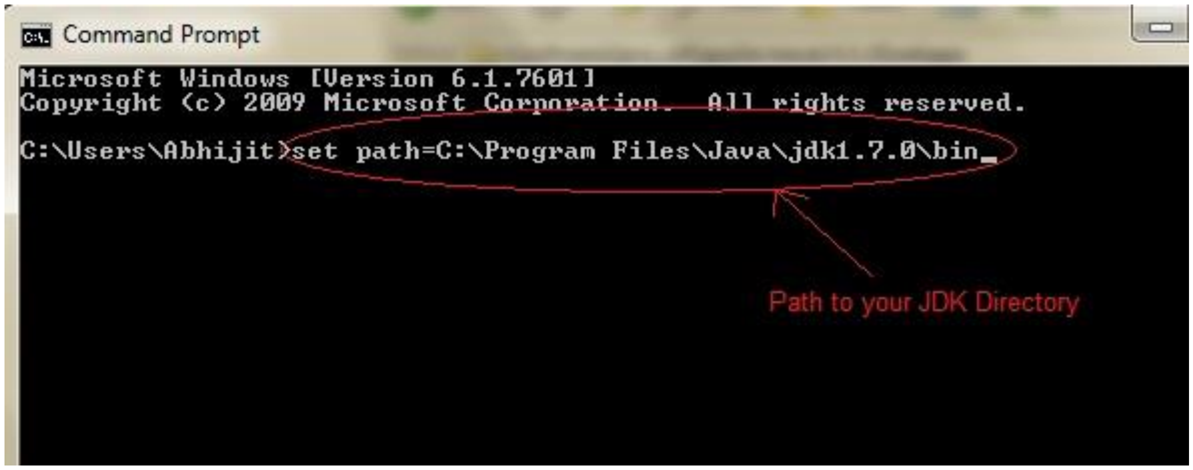
Write above code in a notepad file and save it as **MyServlet.java** anywhere on your PC. Compile it(explained in next step) from there and paste the class file into `WEB-INF/classes/` directory that you have to create inside **Tomcat/webapps** directory.

Compiling a Servlet

To compile a Servlet a JAR file is required. Different servers require different JAR files. In Apache Tomcat server `servlet-api.jar` file is required to compile a servlet class.

Steps to compile a Servlet

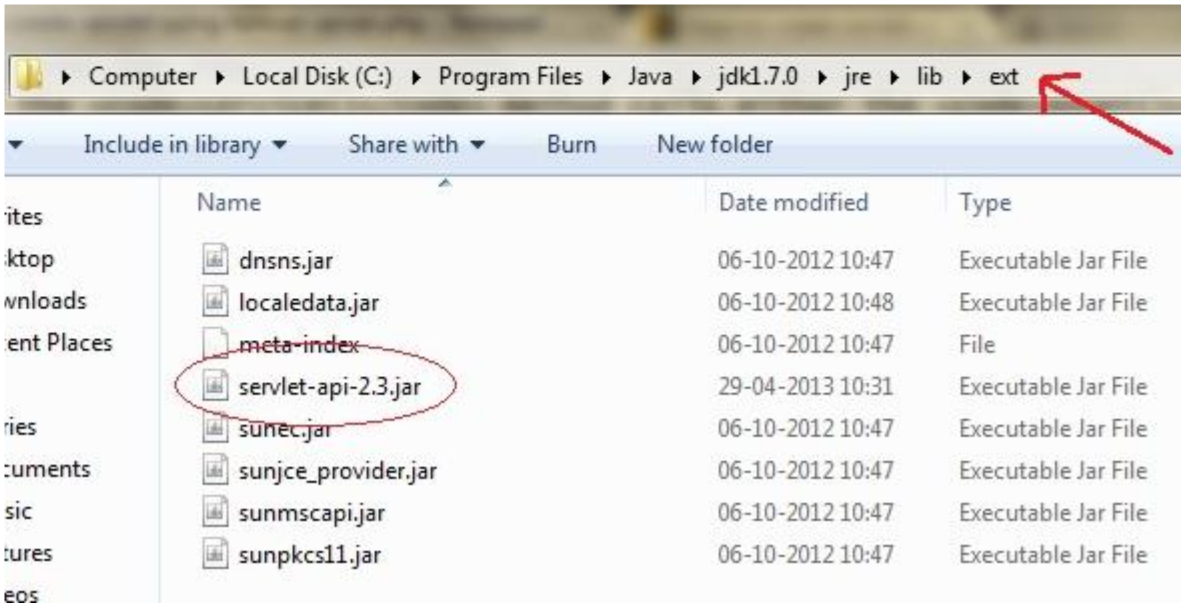
- Set the Class Path.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Abhijit>set path=C:\Program Files\Java\jdk1.7.0\bin_
```

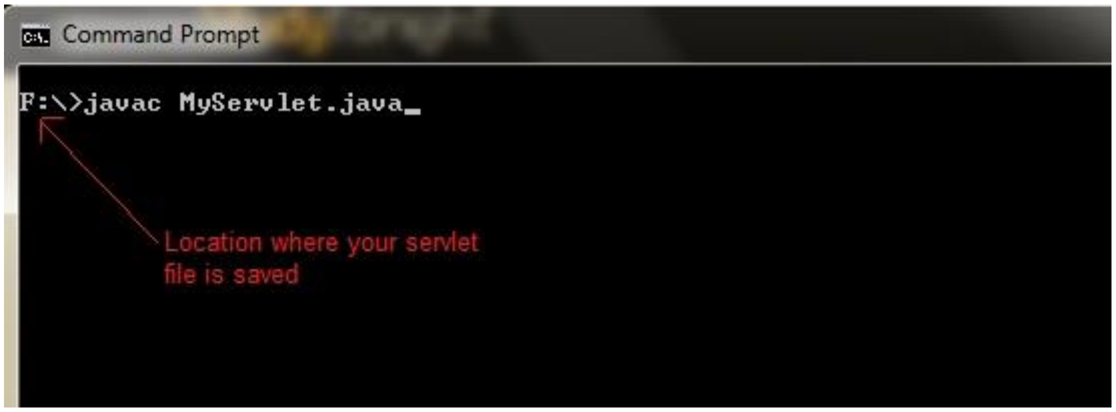
- Download **servlet-api.jar** file.
- Paste the servlet-api.jar file inside `Java\jdk\jre\lib\ext` directory.



Computer ▸ Local Disk (C:) ▸ Program Files ▸ Java ▸ jdk1.7.0 ▸ jre ▸ lib ▸ ext

Name	Date modified	Type
dnsns.jar	06-10-2012 10:47	Executable Jar File
localedata.jar	06-10-2012 10:48	Executable Jar File
meta-index	06-10-2012 10:47	File
servlet-api-2.3.jar	29-04-2013 10:31	Executable Jar File
sunec.jar	06-10-2012 10:47	Executable Jar File
sunjce_provider.jar	06-10-2012 10:47	Executable Jar File
sunmscapi.jar	06-10-2012 10:47	Executable Jar File
sunpkcs11.jar	06-10-2012 10:47	Executable Jar File

- Compile the Servlet class.



```
F:\>javac MyServlet.java_
```

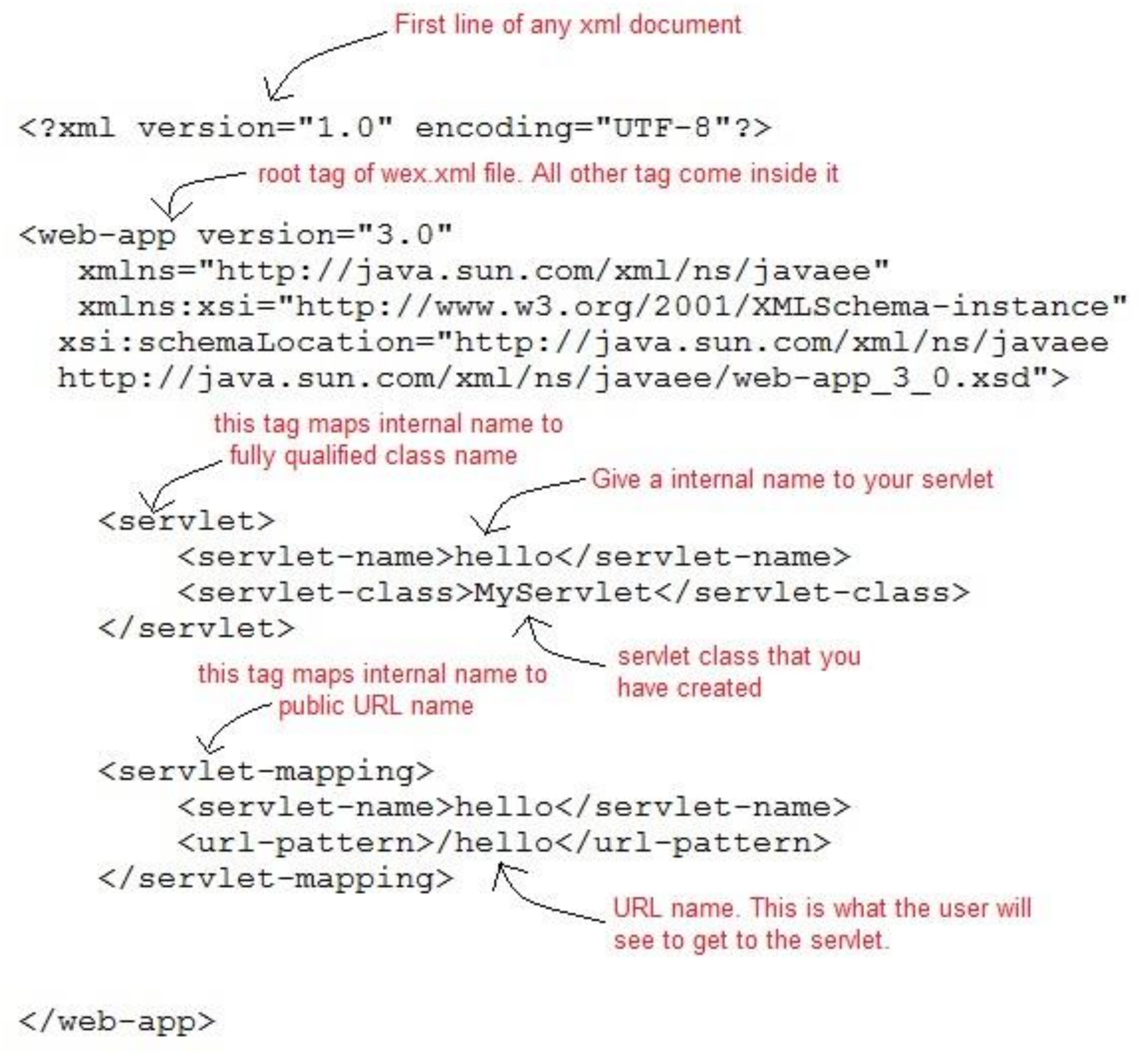
NOTE: After compiling your Servlet class you will have to paste the class file into `WEB-INF/classes/` directory.

Create Deployment Descriptor

Deployment Descriptor(DD) is an XML document that is used by Web Container to run Servlets and JSP pages. DD is used for several important purposes such as:

- Mapping URL to Servlet class.
- Initializing parameters.
- Defining Error page.
- Security roles.
- Declaring tag libraries.

We will discuss about all these in details later. Now we will see how to create a simple **web.xml** file for our web application.



```
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="3.0"
   xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">

    <servlet>
        <servlet-name>hello</servlet-name>
        <servlet-class>MyServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>


</web-app>
```

First line of any xml document
root tag of wex.xml file. All other tag come inside it
this tag maps internal name to fully qualified class name
Give a internal name to your servlet
servlet class that you have created
this tag maps internal name to public URL name
URL name. This is what the user will see to get to the servlet.

Start the Server

Double click on the **startup.bat** file to start your Apache Tomcat Server.

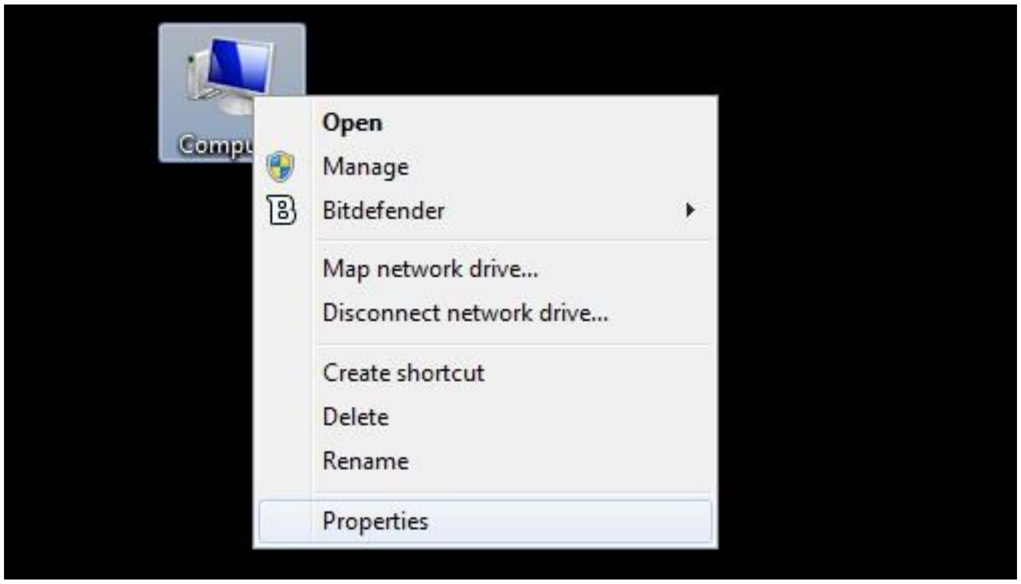
Or, execute the following command on your windows machine using RUN prompt.

```
C:\apache-tomcat-7.0.14\bin\startup.bat
```

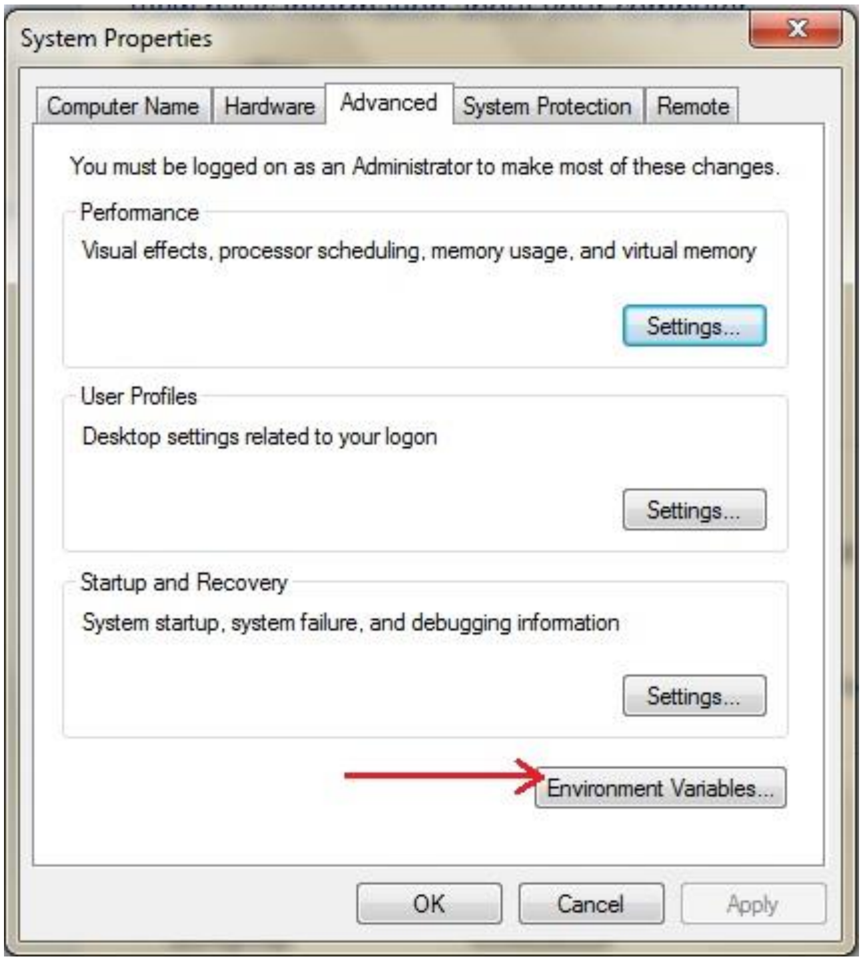
Starting Tomcat Server for the first time

If you are starting Tomcat Server for the first time you need to set JAVA_HOME in the Enviroment variable. The following steps will show you how to set it.

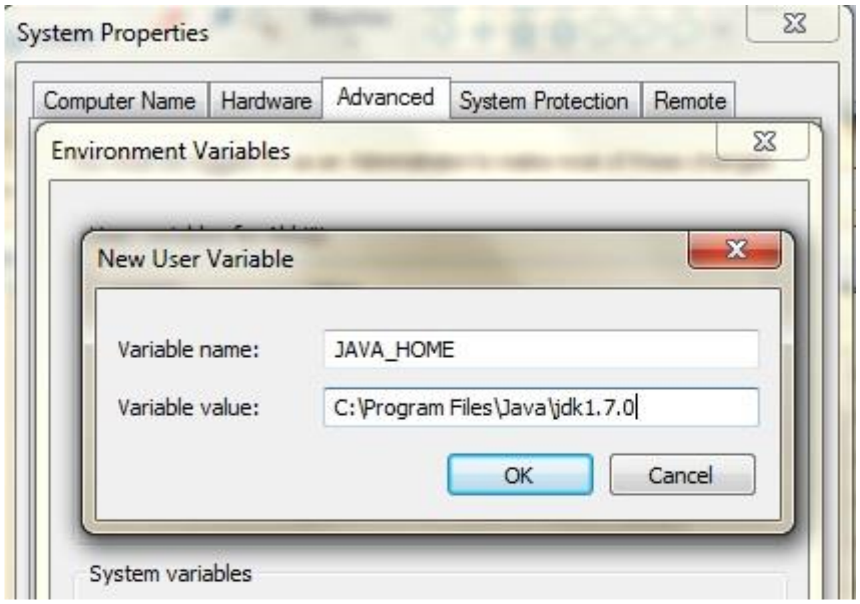
- Right Click on **My Computer**, go to **Properites**.



- Go to **Advanced** Tab and Click on **Enviroment Variables...** button.


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your logon
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

- Click on **New** button, and enter **JAVA_HOME** inside Variable name text field and path of JDK inside Variable value text field. Click OK to save.

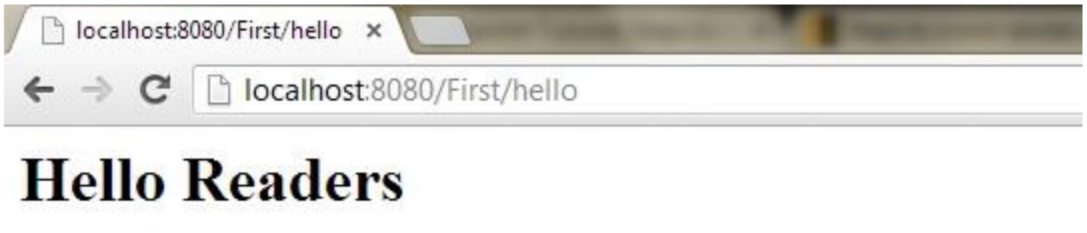


Variable name: JAVA_HOME

Variable value: C:\Program Files\Java\jdk1.7.0

Run Servlet Application

Open Browser and type **http:localhost:8080/First/hello**



Hello Readers

Hurray! Our first Servlet Application ran successfully.