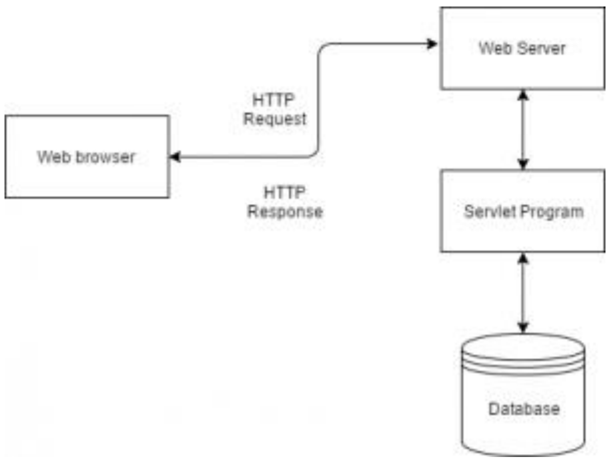
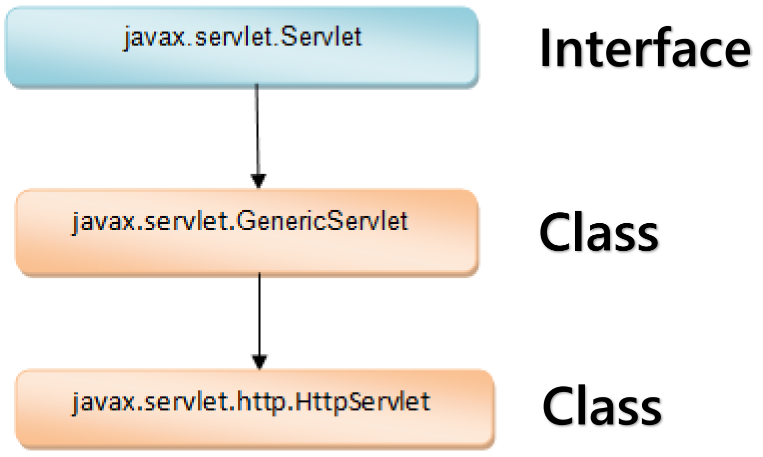
What is Servlet?

It is a server side technology which is used to handle the client request, process the request and generates the dynamic response

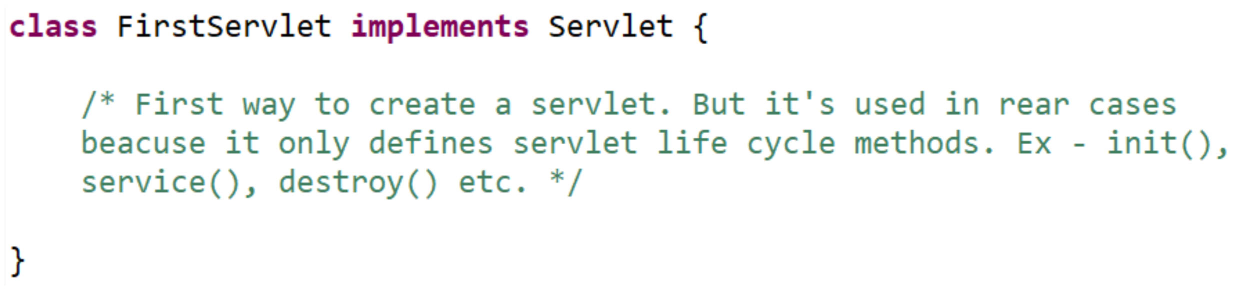


Servlet Hierarchy



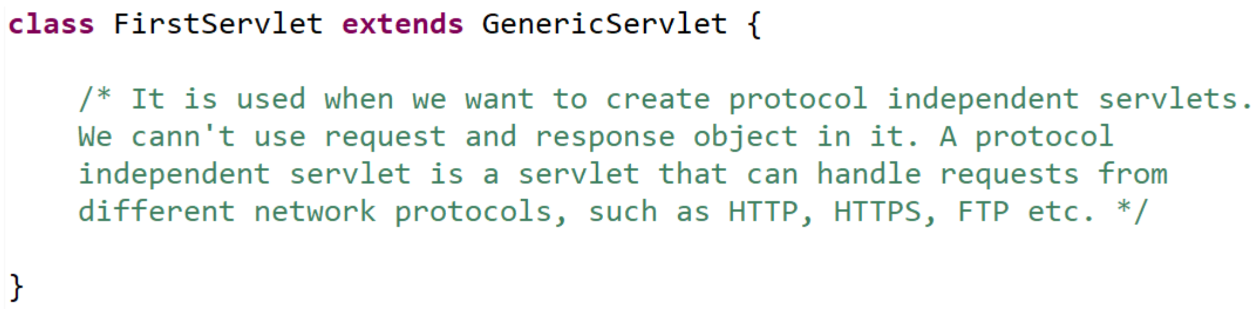
Now it is changed into jakarta.servlet.GenericServlet, jakarta.servlet.Servlet, jakarta.servlet.HttpServlet

1. First way to create a servlet is to implement the Servlet interface



```
class FirstServlet implements Servlet {

    /* First way to create a servlet. But it's used in rear cases
    beacuse it only defines servlet life cycle methods. Ex - init(),
    service(), destroy() etc. */

}
```

2. Second way is to extend the GenericServlet class



```
class FirstServlet extends GenericServlet {

    /* It is used when we want to create protocol independent servlets.
    We cann't use request and response object in it. A protocol
    independent servlet is a servlet that can handle requests from
    different network protocols, such as HTTP, HTTPS, FTP etc. */

}
```

3. Thrid way is to extend the HttpServlet class (*)

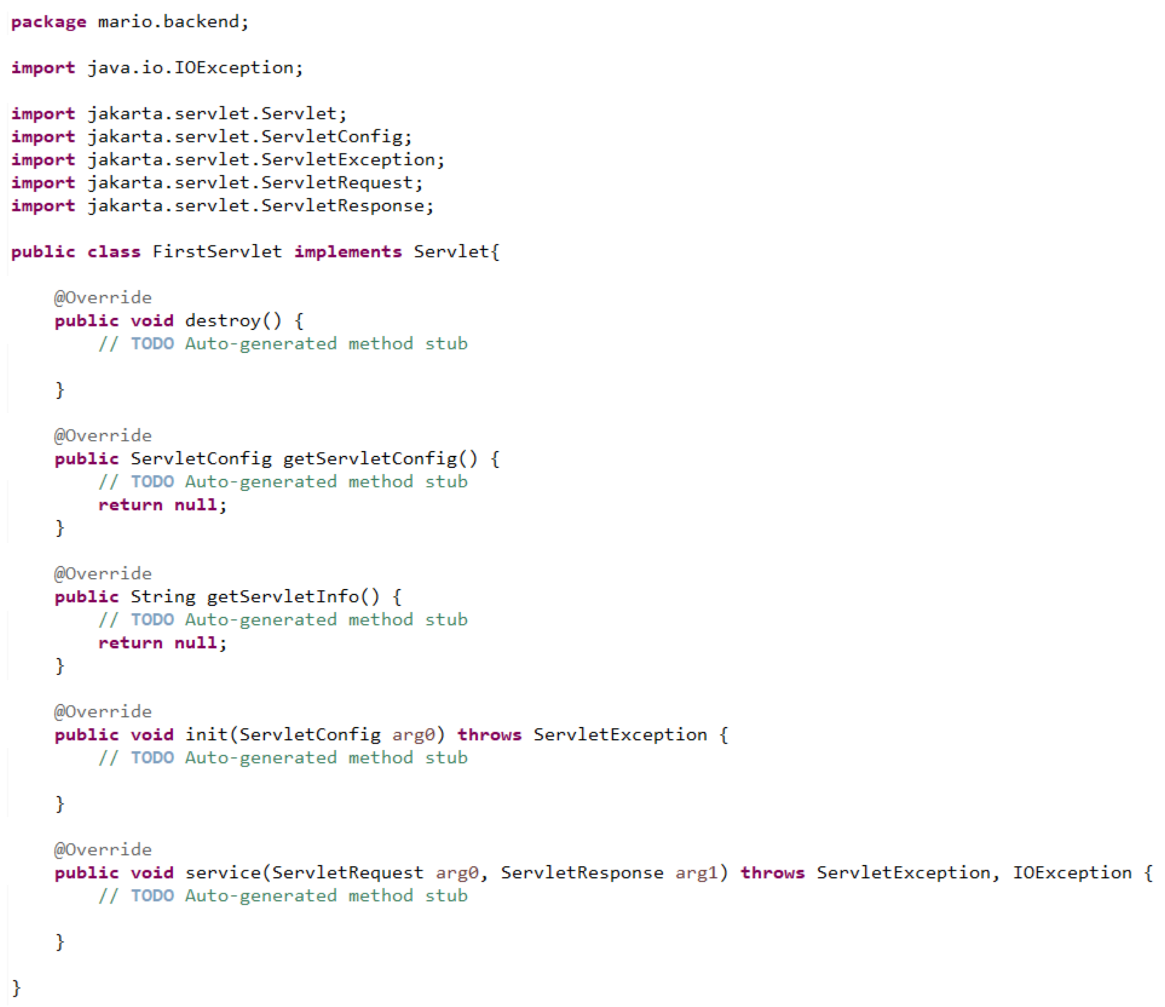
```
class FirstServlet extends HttpServlet {

    /* It is used when we want http specific methods and
       request-response object */

}
```

So, our first step is to extends the HttpServlet class. Then we will override the doGet or doPost method.

```
class FirstServlet extends HttpServlet {

    @Override
    doGet(HttpServletRequest, HttpServletResponse)
    //backend code

    /* or */

    @Override
    doPost(HttpServletRequest, HttpServletResponse)
    //backend code

}
```

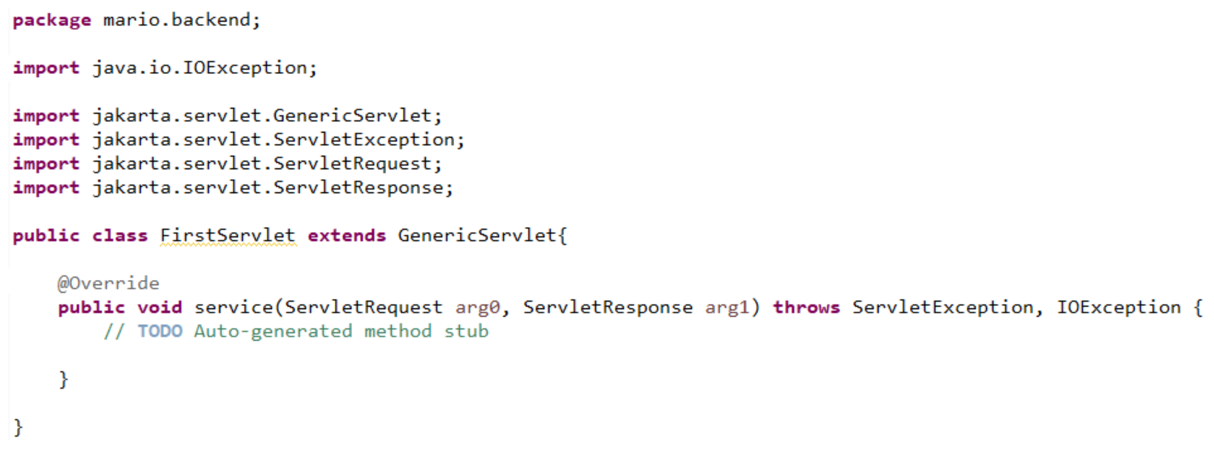
We have to create one file, that is Deployment Descriptor file – web.xml

1. First way



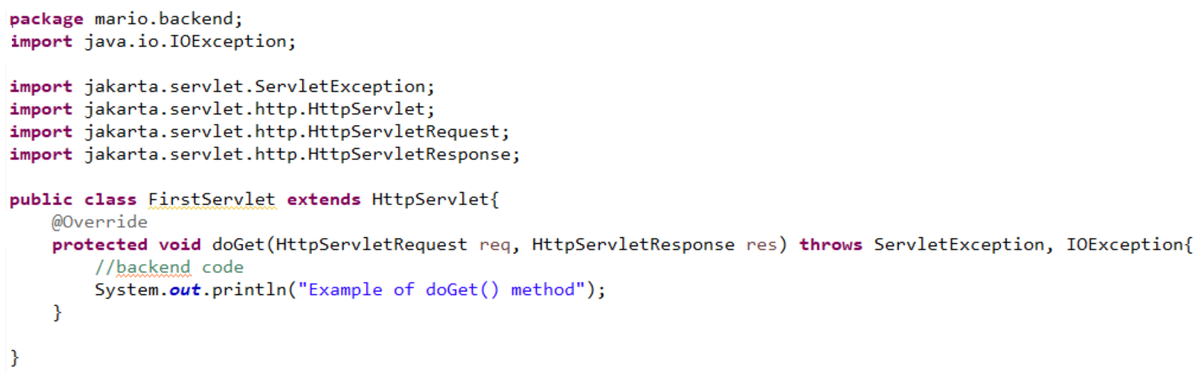
```java
package mario.backend;

import java.io.IOException;

import jakarta.servlet.Servlet;
import jakarta.servlet.ServletConfig;
import jakarta.servlet.ServletException;
import jakarta.servlet.ServletRequest;
import jakarta.servlet.ServletResponse;

public class FirstServlet implements Servlet{

    @Override
    public void destroy() {
        // TODO Auto-generated method stub

    }

    @Override
    public ServletConfig getServletConfig() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public String getServletInfo() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void init(ServletConfig arg0) throws ServletException {
        // TODO Auto-generated method stub

    }

    @Override
    public void service(ServletRequest arg0, ServletResponse arg1) throws ServletException, IOException {
        // TODO Auto-generated method stub

    }

}
```

2. Second way



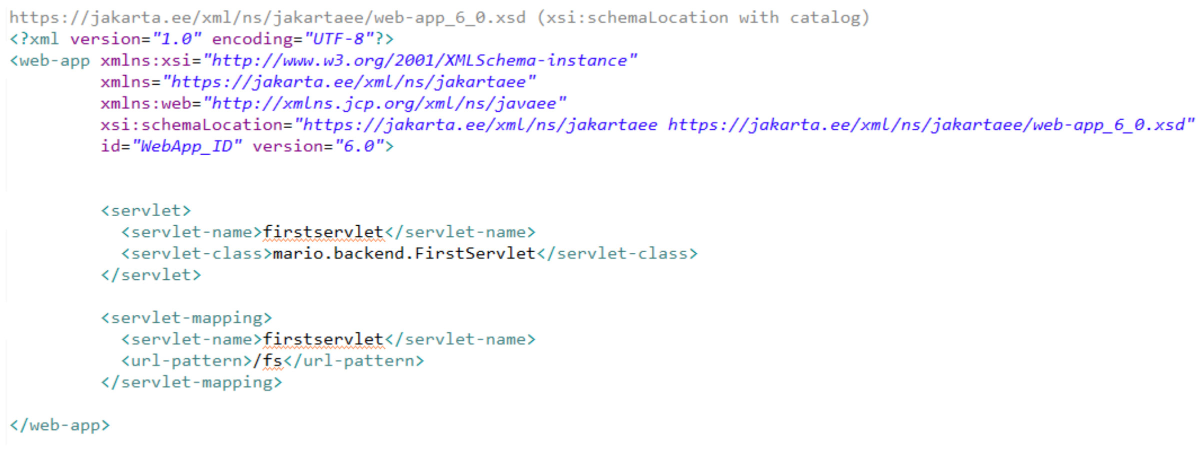
```java
package mario.backend;

import java.io.IOException;

import jakarta.servlet.GenericServlet;
import jakarta.servlet.ServletException;
import jakarta.servlet.ServletRequest;
import jakarta.servlet.ServletResponse;

public class FirstServlet extends GenericServlet{

    @Override
    public void service(ServletRequest arg0, ServletResponse arg1) throws ServletException, IOException {
        // TODO Auto-generated method stub

    }

}
```

3. Third way (*)

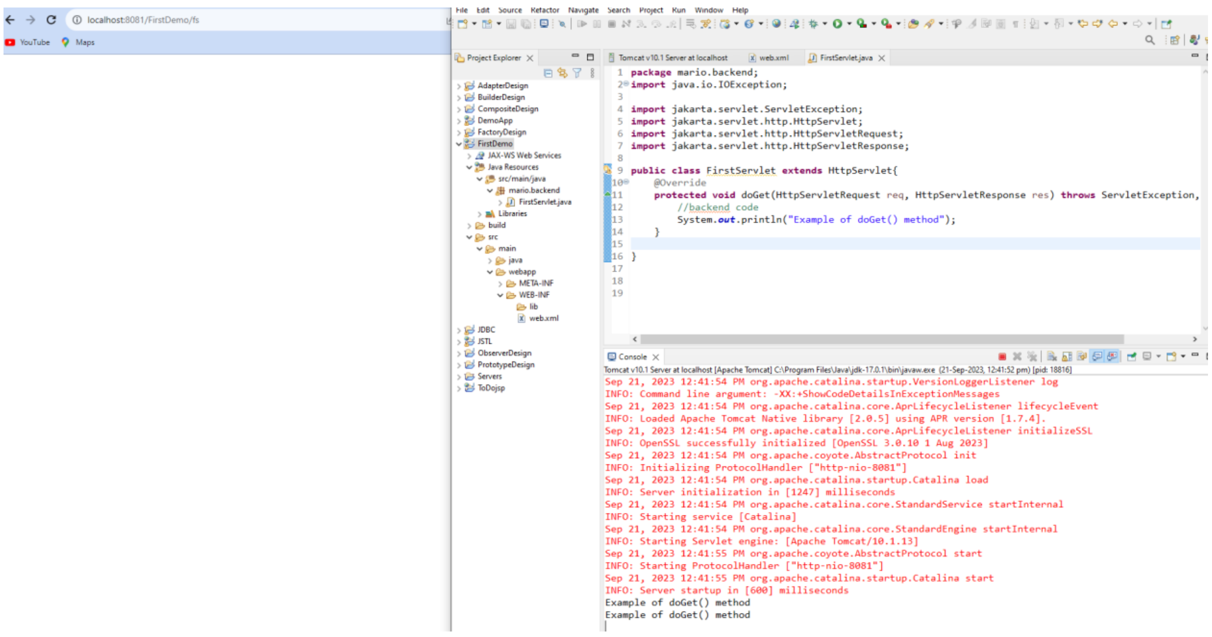


```java
package mario.backend;
import java.io.IOException;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class FirstServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException{
        //backend code
        System.out.println("Example of doGet() method");
    }

}
```

Now we have to configure the web.xml file



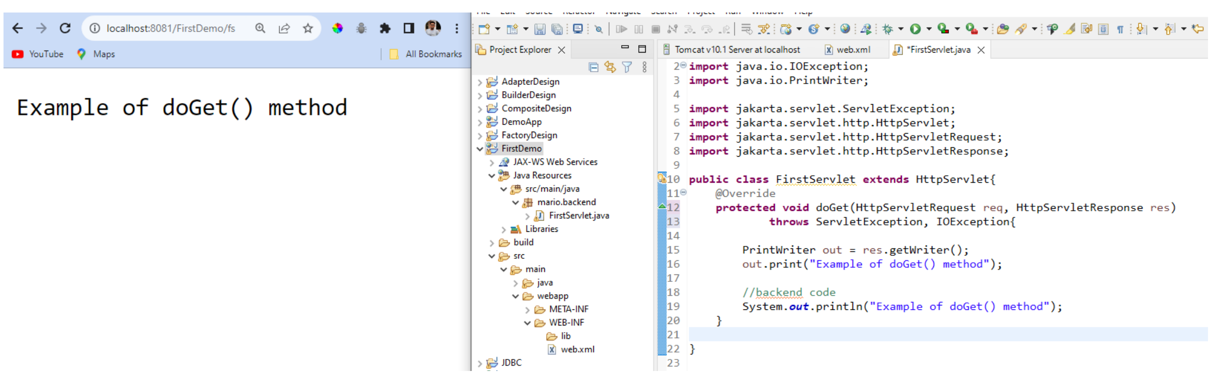
```xml
https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd (xsi:schemaLocation with catalog)
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:web="http://xmlns.jcp.org/xml/ns/javaee"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
         id="WebApp_ID" version="6.0">


         <servlet>
           <servlet-name>firstservlet</servlet-name>
           <servlet-class>mario.backend.FirstServlet</servlet-class>
         </servlet>

         <servlet-mapping>
           <servlet-name>firstservlet</servlet-name>
           <url-pattern>/fs</url-pattern>
         </servlet-mapping>

</web-app>
```



```java
package mario.backend;
import java.io.IOException;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class FirstServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
        //backend code
        System.out.println("Example of doGet() method");
    }

}
```

```
Tomcat v10.1 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (21-Sep-2023, 12:41:52 pm) [pid: 18816]
Sep 21, 2023 12:41:54 PM org.apache.catalina.startup.VersionLoggerListener log
INFO: Command line argument: -XX:+ShowCodeDetailsInExceptionMessages
Sep 21, 2023 12:41:54 PM org.apache.catalina.core.AprLifecycleListener lifecycleEvent
INFO: Loaded Apache Tomcat Native library [2.0.5] using APR version [1.7.4].
Sep 21, 2023 12:41:54 PM org.apache.catalina.core.AprLifecycleListener initializeSSL
INFO: OpenSSL successfully initialized [OpenSSL 3.0.10 1 Aug 2023]
Sep 21, 2023 12:41:54 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-nio-8081"]
Sep 21, 2023 12:41:54 PM org.apache.catalina.startup.Catalina load
INFO: Server initialization in [1247] milliseconds
Sep 21, 2023 12:41:54 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting service [Catalina]
Sep 21, 2023 12:41:54 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet engine: [Apache Tomcat/10.1.13]
Sep 21, 2023 12:41:55 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-8081"]
Sep 21, 2023 12:41:55 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in [600] milliseconds
Example of doGet() method
Example of doGet() method
```

If we want to print in browser



```
Example of doGet() method
```

```java
import java.io.IOException;
import java.io.PrintWriter;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class FirstServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException{

        PrintWriter out = res.getWriter();
        out.print("Example of doGet() method");

        //backend code
        System.out.println("Example of doGet() method");
    }

}
```

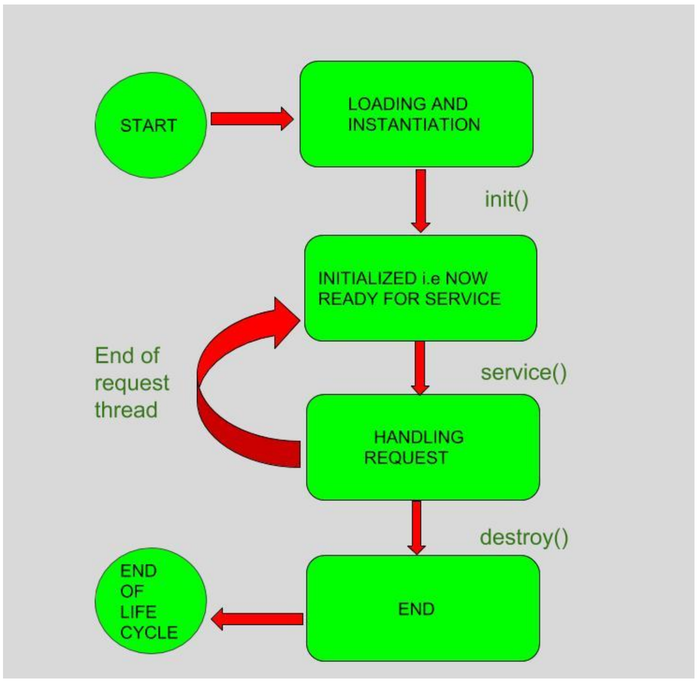
Life Cycle of Servlet :-

1. Loading and instantiation -> When server is started, servlet class is loaded in the memory & servlet object is created.

2. Initialization -> Servlet object will be initialized by invoking init() method.

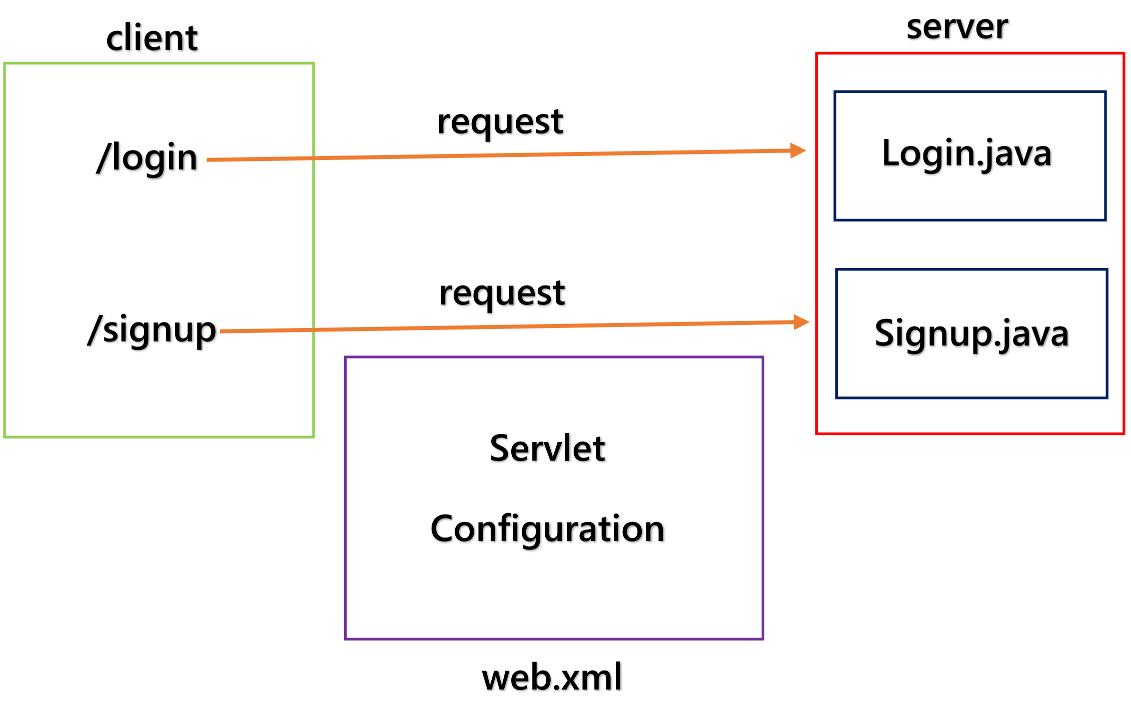
3. Request handling -> It will handle or serve the client request. In this phase service() method will be invoked. The service() method will execute multiple times because it can get multiple request. So, the concept of multi-threading is internally handled.

4. Destroy -> When the server is shut down, destroy() method will be executed and servlet object will be deleted.



web.xml

It is also known as Deployment Descriptor file.



Tags of web.xml file :-

```
<web-app>
         <servlet-mapping>
           <servlet-name>login</servlet-name>
           <url-pattern>/login</url-pattern>
         </servlet-mapping>
         <servlet>
           <servlet-name>login</servlet-name>
           <servlet-class>packageName.Login</servlet-class>
         </servlet>

         <servlet-mapping>
           <servlet-name>signup</servlet-name>
           <url-pattern>/signup</url-pattern>
         </servlet-mapping>
         <servlet>
           <servlet-name> signup </servlet-name>
           <servlet-class>packageName.Signup</servlet-class>
         </servlet>
</web-app>
```

We create web.xml file in WEB-INF folder. But in latest version it is not compulsory to create the web.xml file in particular file location WEB-INF. We should create anywhere using annotations

Different tasks of web.xml file

- Servlet Configuration
- JSP File Configuration
- Filters Configuration
- Listeners Configuration
- Error Page Configuration
- Welcome File Configuration etc

HttpServletRequest

It is an interface. If we want to get the value of the data in request object then the HttpServletRequest is used.

We can get those value of those data with the help of some common methods shown below :-

- getParameter(String name): It will return the value of the request parameter specified by name.
- getCookies(): Returns an array of Cookie objects representing the cookies included in the request
- getSession(): Returns the current session associated with the request or creates a new one if create is true.
- getMethod(): Returns the HTTP method of the request, such as GET, POST, PUT, DELETE etc.
- getAttribute(String name): Returns the value of the named attribute as an object.
- setAttribute(String name, Object value): Binds an object to a given attribute name in the request scope.
- getHeader(String name): Returns the value of the specified HTTP header.

- getHeaderNames(): Returns an enumeration of all the header names sent with the request.

  Etc...

HttpServletResponse

It is an interface. With the help of it we can get the value of the request object's data and write the value it in browser.

Some methods :-

- getWriter(): Returns a PrintWriter object that can be used to send character text to the client.
- setContentType(String type): Sets the MIME(Multipurpose Internet Mail Extensions) type of the response.
- setContentLength(int len): Sets the length of the content being returned in the response.
- sendRedirect(String location): Redirects the client to a different URL.
- sendError(int sc, String msg): Sends an error response to the client with the specified status code and message.
- addCookie(Cookie cookie): Adds a cookie to the response.
- setStatus(int sc): Sets the status code of the response.
- setHeader(String name, String value): Sets the value of the specified response header.
- addHeader(String name, String value): Adds a response header with the given name and value.

  Etc...

HTTP Methods:- HTTP (Hypertext Transfer Protocol) specifies a collection of request methods to specify what action is to be performed on a particular resource. The most commonly used HTTP request methods are GET, POST, PUT, PATCH, and DELETE. These are equivalent to the CRUD operations (create, read, update, and delete).

Some common HTTP methods :-

1	**GET** The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.
2	**HEAD** Same as GET, but transfers the status line and header section only.
3	**POST** A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
4	**PUT** Replaces all current representations of the target resource with the uploaded content.
5	**DELETE** Removes all current representations of the target resource given by a URI.

GET & POST are the HTTP methods those are used in everywhere.

Difference between GET and POST :-

Visibility

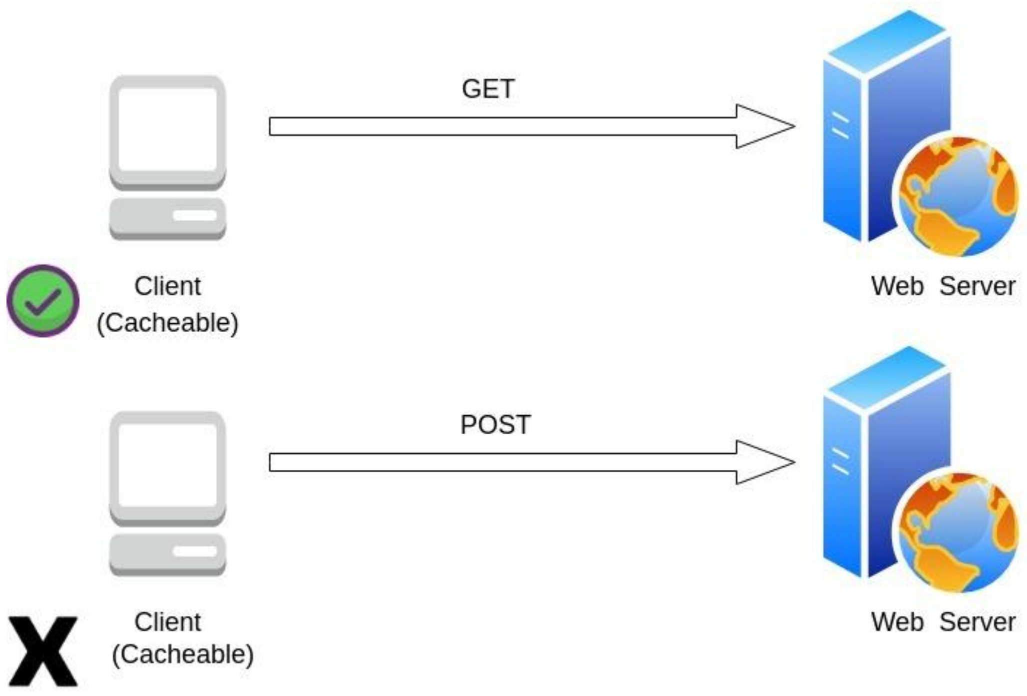
When using GET, data parameters are included in the URL and visible to everyone. However, when using POST, data is not displayed in the URL but in the HTTP message body.

Security

GET is less secure because the URL contains part of the data sent. On the other hand, POST is safer because the parameters are not stored in web server logs or the browser history.

Cache

GET requests can be cached and remain in the browser history, while POST requests cannot. This means GET requests can be bookmarked, shared, and revisited, while POST requests cannot:



Server State

GET requests are intended to retrieve data from a server and do not modify the server's state. On the other hand, POST requests are used to send data to the server for processing and may modify the server's state.

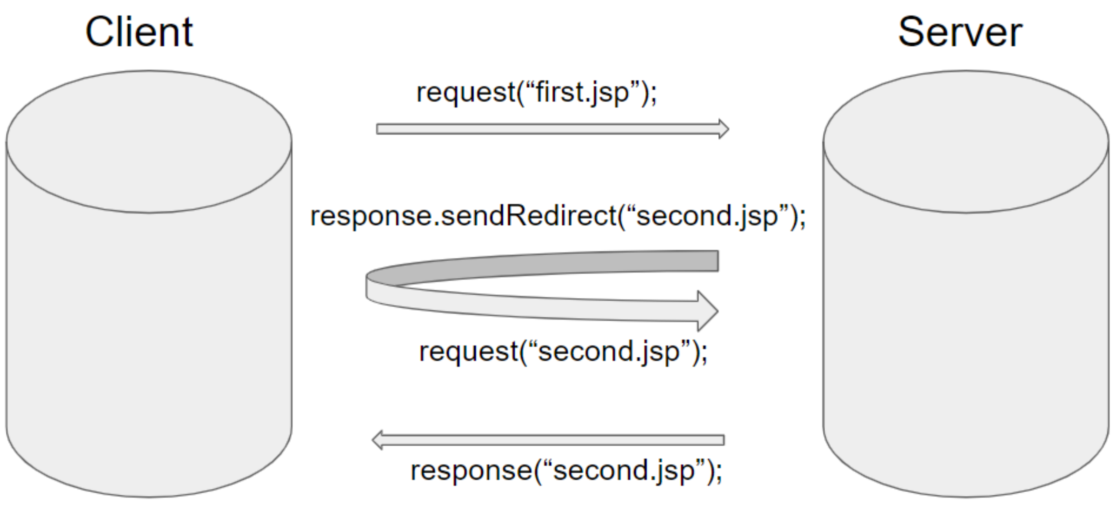
Amount of Data Transmitted

The GET method is limited to a maximum number of characters, while the POST method has no such limitation. This is because the GET method sends data through the resource URL, which is limited in length, while the POST method sends data through the HTTP message body, which has no such limitation.

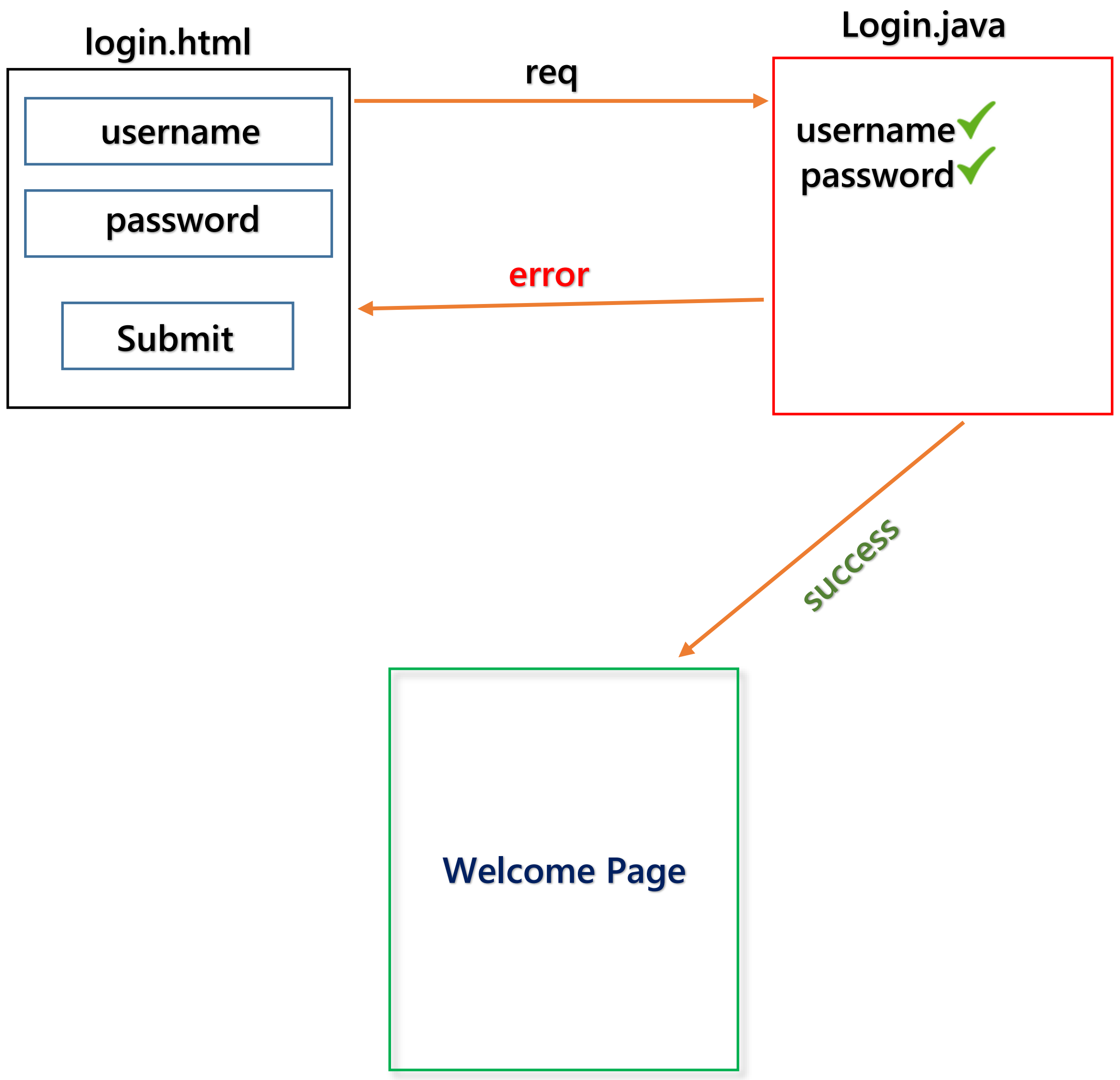
Data Type

The GET method supports only string data types, while the POST method supports different data types such as string, numeric, binary, and so on.

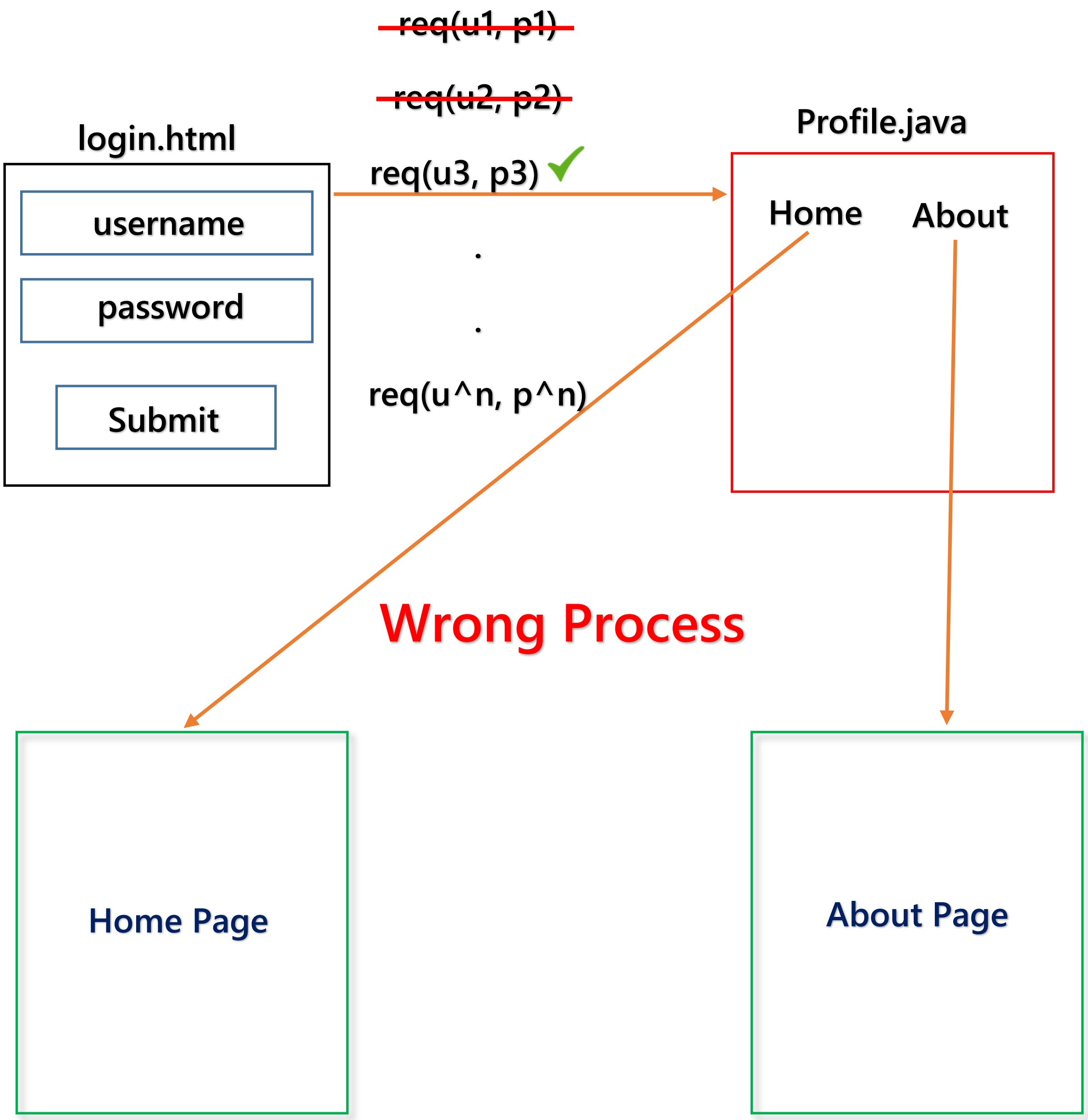
sendRedirect() method redirects the response to another resource, inside or outside the server. It makes the client/browser to create a new request to get to the resource. It sends a temporary redirect response to the client using the specified redirect location URL. It is the method of HttpServletResponse. It will change the URL.

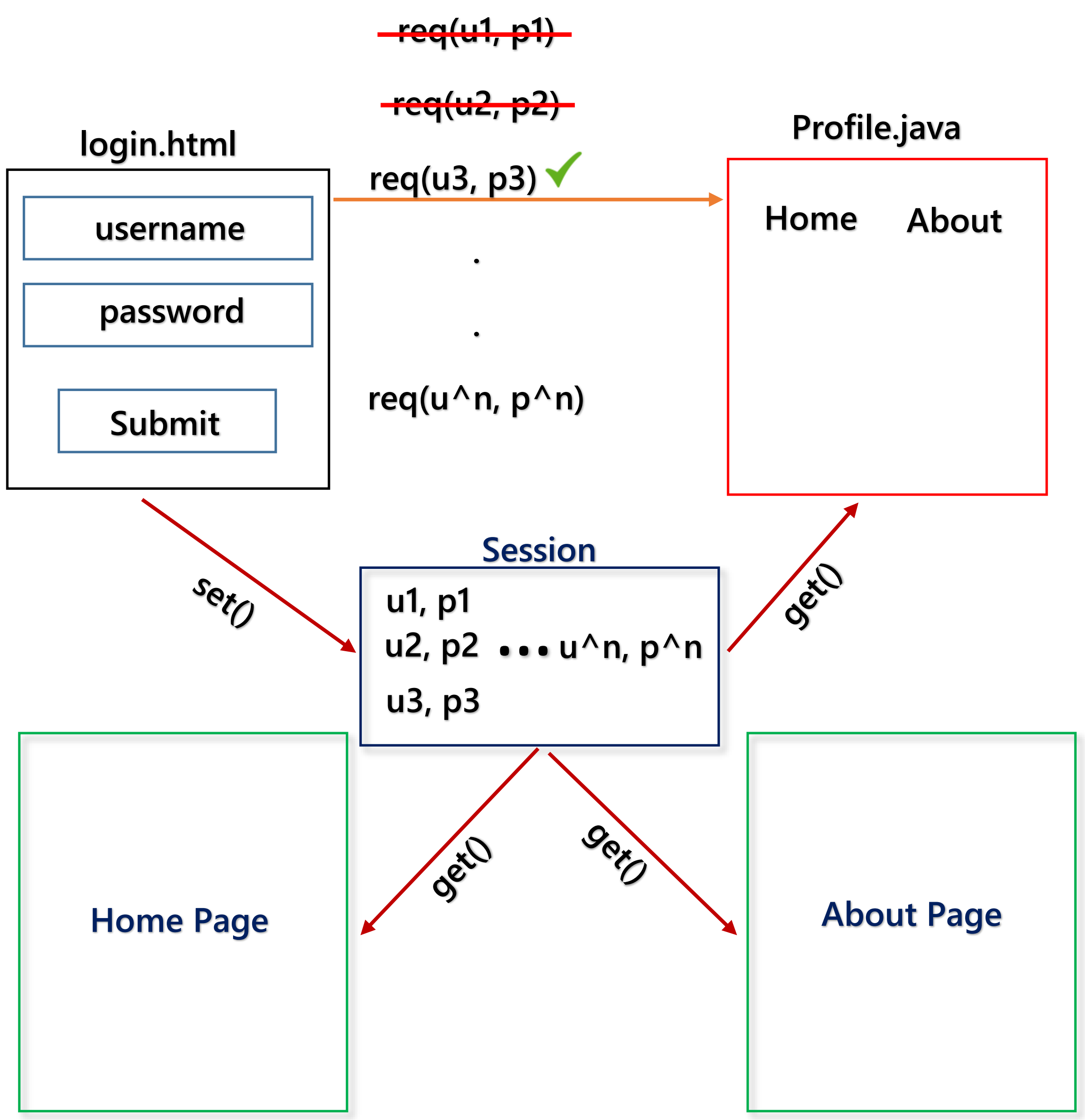


RequestDispatcher is a public interface. Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server. URL is not changed in this case.



Session Management - Servlet Session Management is a mechanism in Java used by Web container to store session information. Session tracking is a way to manage the data of a user, this is known as session management in servlet. Session in Java are managed through different ways, such as, HTTP Session API, Cookies, URL rewriting, etc.





We can create sessions object using HttpServletRequest() interface.

HttpSession session = req.getSession();

Some methods of HttpSession()

- setAttribute(String name, Object value) :- Binds an object name in the session.
- getAttribute(String name) :- Retrieves the object associated with the specified name from the session.
- removeAttribute(String name) :- Removes the object associated with the specified name from the specified name from the session.
- invalidate() :- Invalidates the session , effectively removing all session attributes and ending
- getId() :- Returns the unique identifier assigned to the session.
- isNew() :- Returns a Boolean indicating whether the session is a anew session or an exiting one.
- getLastAccessedTime() :- Returns the time, in milliseconds since midnight January 1, 1970 GMT, when the session was last accessed by the client.
- getMaxInactiveInterval() :- Returns the maximum the interval, in seconds, between client requests before the session is invalidated.
- setMaxInactiveInterval(int interval) :- Sets the maximum time interval, in seconds, between client requests before the session is invalidated.

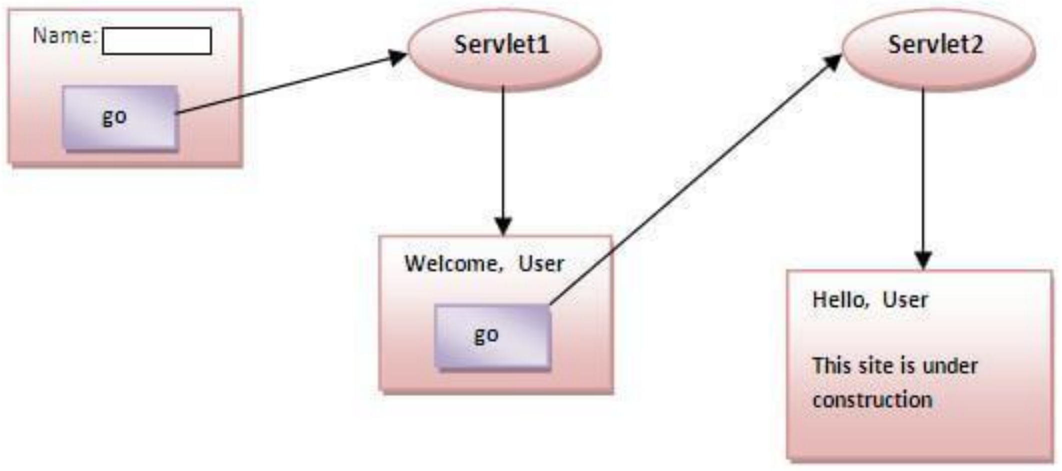
Cookies

Cookies are text files stored on the client computer and they are kept for various information tracking purpose. Java Servlets transparently supports HTTP cookies.

There are three steps involved in identifying returning users −

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.

- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.



ServletConfig Interface

An object of ServletConfig is created by the web container for each servlet. This object can be used to get configuration information from web.xml file.

If the configuration information is modified from the web.xml file, we don't need to change the servlet. So it is easier to manage the web application if any specific content is modified from time to time.

Methods of ServletConfig interface

- public String getInitParameter(String name): Returns the parameter value for the specified parameter name.
- public Enumeration getInitParameterNames(): Returns an enumeration of all the initialization parameter names.
- public String getServletName(): Returns the name of the servlet.
- public ServletContext getServletContext(): Returns an object of ServletContext.

ServletContext Interface

An object of ServletContext is created by the web container at time of deploying the project. This object can be used to get configuration information from web.xml file. There is only one ServletContext object per web application.

If any information is shared to many servlet, it is better to provide it from the web.xml file using the <context-param> element.

Usage of ServletContext Interface

There can be a lot of usage of ServletContext object. Some of them are as follows:

- The object of ServletContext provides an interface between the container and servlet.
- The ServletContext object can be used to get configuration information from the web.xml file.
- The ServletContext object can be used to set, get or remove attribute from the web.xml file.
- The ServletContext object can be used to provide inter-application communication.

Commonly used methods of ServletContext interface

- public String getInitParameter(String name):Returns the parameter value for the specified parameter name.
- public Enumeration getInitParameterNames():Returns the names of the context's initialization parameters.
- public void setAttribute(String name,Object object):sets the given object in the application scope.
- public Object getAttribute(String name):Returns the attribute for the specified name.
- public Enumeration getInitParameterNames():Returns the names of the context's initialization parameters as an Enumeration of String objects.

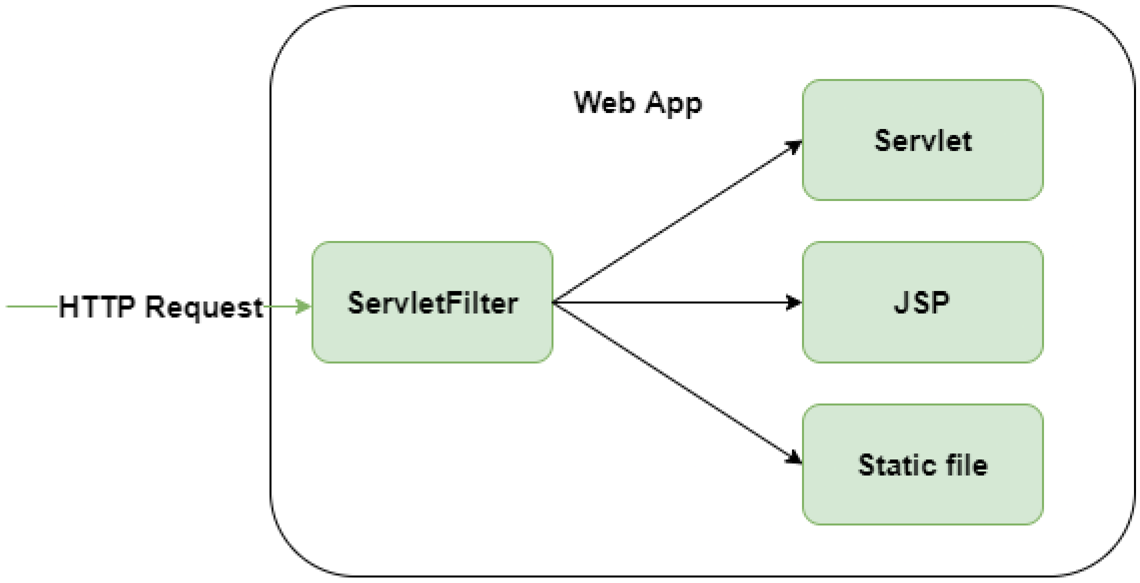
- public void removeAttribute(String name):Removes the attribute with the given name from the servlet context.

Servlet Filter

A filter is an object that is invoked at the preprocessing and postprocessing of a request.

It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc.

The servlet filter is pluggable, i.e. its entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet.



Usage of Filter

- recording all incoming requests
- logs the IP addresses of the computers from which the requests originate
- conversion
- data compression
- encryption and decryption

- input validation etc.

Filter API

Like servlet filter have its own API. The javax.servlet package contains the three interfaces of Filter API.

1. Filter
2. FilterChain
3. FilterConfig

1) Filter interface

For creating any filter, you must implement the Filter interface. Filter interface provides the life cycle methods for a filter.

public void init(FilterConfig config) - init() method is invoked only once. It is used to initialize the filter.

public void doFilter(HttpServletRequest req, HttpServletResponse res, FilterChain chain) – doFilter() method is invoked every time when user request to any resource, to which the filter is mapped. It is used to perform filtering tasks.

public void destroy() – This is invoked only when filter is taken out of the service.

2) FilterChain interface

1. The object of FilterChain is responsible to invoke the next filter or resource in the chain.This object is passed in the doFilter method of Filter interface.The FilterChain interface contains only one method: public void doFilter(HttpServletRequest request, HttpServletResponse response): it passes the control to the next filter or resource.

3) FilterConfig interface

An object of FilterConfig is created by the web container. This object can be used to get the configuration information from the web.xml file.

Methods of FilterConfig interface

There are following 4 methods in the FilterConfig interface.

1. public void init(FilterConfig config): init() method is invoked only once it is used to initialize the filter.
2. public String getInitParameter(String parameterName): Returns the parameter value for the specified parameter name.
3. public java.util.Enumeration getInitParameterNames(): Returns an enumeration containing all the parameter names.
4. public ServletContext getServletContext(): Returns the ServletContext object.