PENTAGON SPACE™
Mastering The Future

SERVLETS AND JSP

PUNITH B

Desktop Applications/ Standalone applications

- A desktop application has to be downloaded, installed in our computer, then we can able to use it.
- If a new version of a desktop application is released, then you have to update or reinstall the application again in your system.
- For example, IDE software's like Eclipse, IntelliJ, STS and Anti-virus software's like McAfee, Kaspersky are desktop applications.

Web applications

- A web application is a software application which executes on a web server and users can access through browsers over internet.
- A web application can provide the service to multiple users at a time over internet.
- If any changes are made to a web application then users doesn't require to make any changes in their computer.
- Web Applications are of two types:
    - ➢ Static Web Application.
    - ➢ Dynamic Web Application.

Static Web Applications

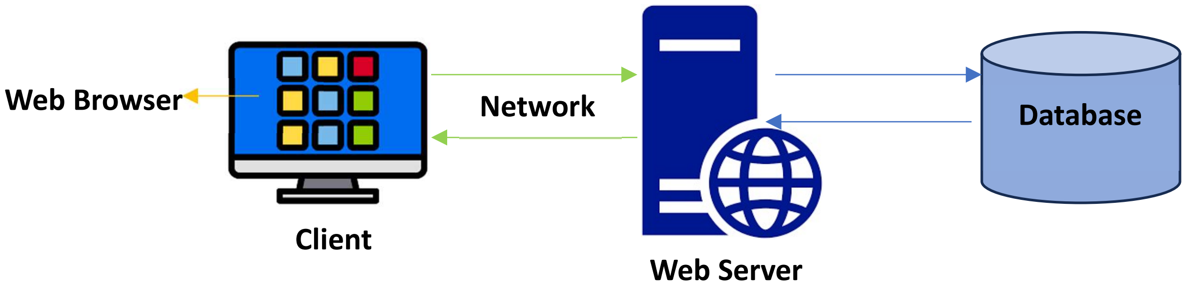
- A static web application will provide the same content to all the users without any server-side processing.
- Static web applications can be developed using HTML, CSS and JavaScript.
- Ex: personal blogs, online tutorials are static web applications.

Dynamic Web Applications

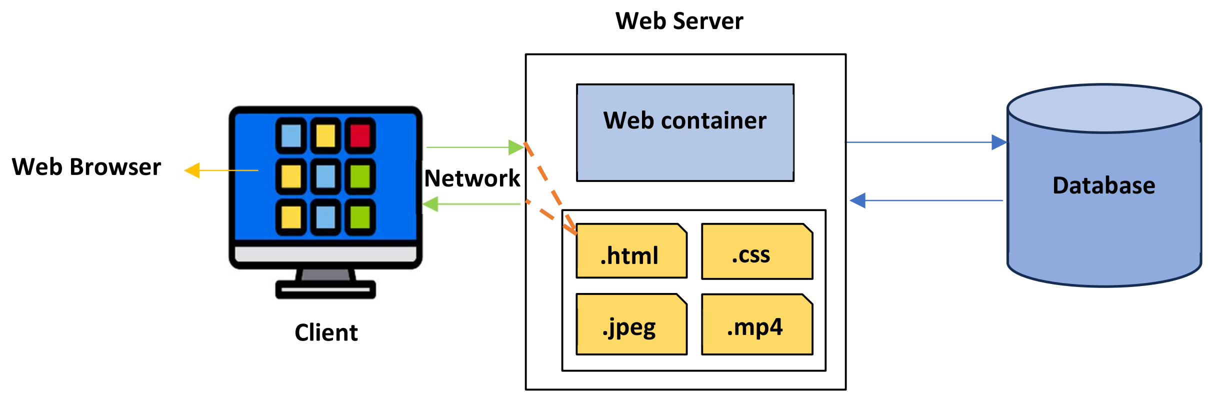
- A dynamic web application will take user input, process it at server-side and generates dynamic content to the users.
- Dynamic web applications can be developed using server-side web technologies like servlets, asp.net, JSP, PHP, Node.js, ..
- To develop dynamic web applications, we need both client-side and server-side web technologies.

Ex: facebook, amazon, irctc are dynamic web applications.

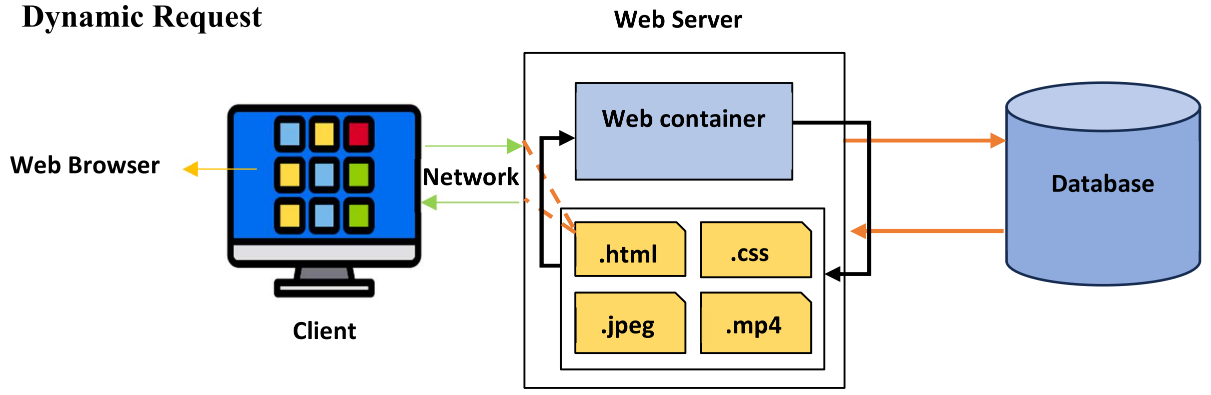
- A dynamic web application also requires html, css and java script files. It also contains servlet classes, JSP files to do request processing at server.



Static Request



Dynamic Request



- Here, the client will do a request using web browser and the request is sent to server via network.
- In case of static request, the web server will accept the client request and checks whether the request is for static resource or dynamic resource.
- If it is a static request, web server alone handles the request and provide the response to the client.
- If it is a dynamic request like servlet class or a JSP file, then web server will forward the request to the web container.
- The web container will execute the servlet classes, and provides the dynamic response generated by the servlet classes and send it to the web server.
- Finally, the web server will forward the dynamic response to the client.

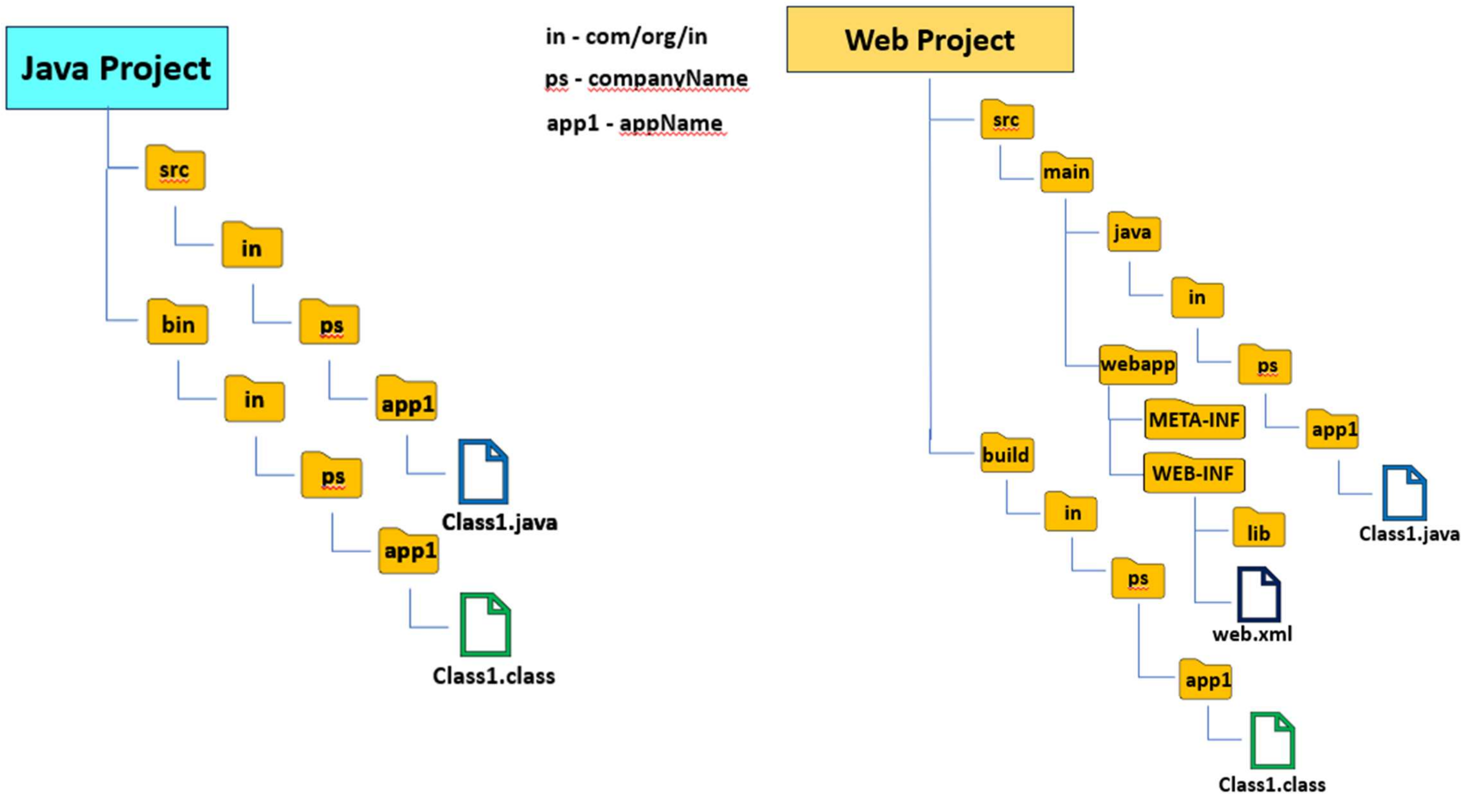
Web Container

- A web container is responsible for executing dynamic resources like servlet classes and JSP files.
- It is responsible for handling the dynamic request and dynamic response.

Deployment Descriptor Stub(web.xml)

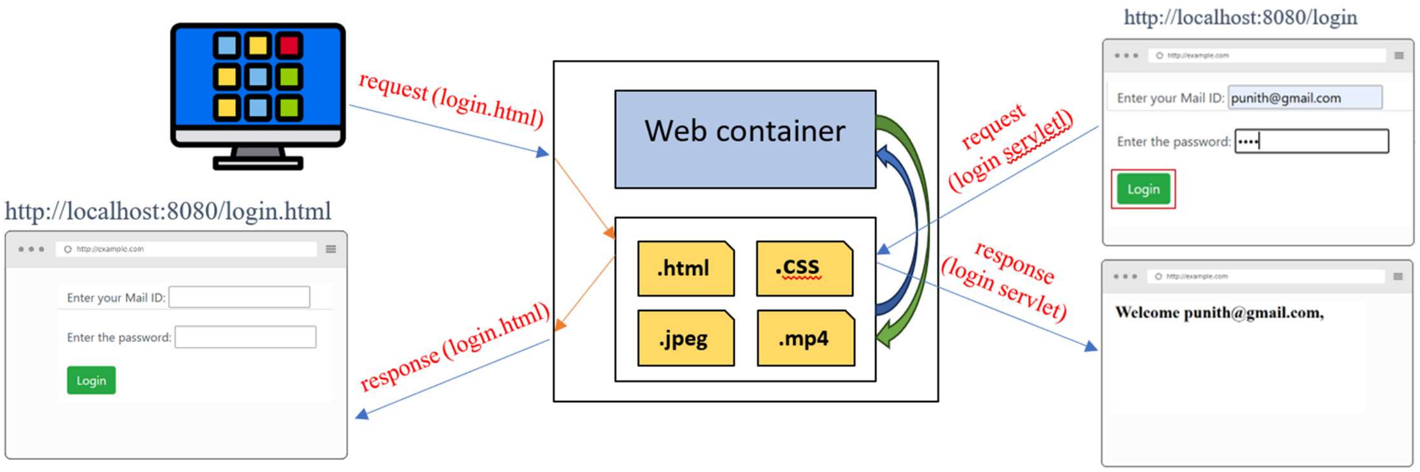
- Web.xml or deployment descriptor stub is used to write the configuration of the application. It is an Extensible Markup Language file.
- In this file, a developer has to write some information about the servlet classes, which is needed by the container.
- A **Dynamic Descriptor Stub** in the context of Java web applications refers to a mechanism to allow **dynamic registration of servlets.**

Structure of a Java project and Dynamic Web Project

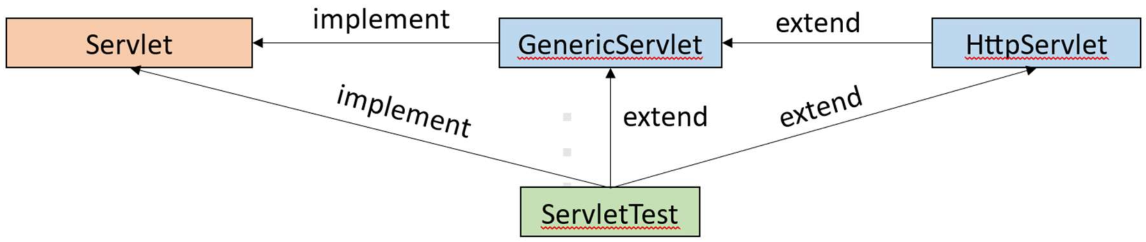


Servlets

- **Servlets** are nothing but the **server side(Dynamic) web technologies**.
- They are the **server-side** Java Programs.
- Servlet follows all the types of Logics. i.e.
    - ➢ Presentation Logic (Front End)
    - ➢ Persistence Logic (Database)
    - ➢ Business Logic (Backend)



Servlet Hirarchy



GenericServlet

- **GenericServlet** is an abstract class present in **jakarta.servlet package.**
- GenericServlet is a **protocol independent**.
- It does not support session.
- GenericServlet consist of both abstract method and concrete method.
    - ➢ Abstract Method – service()
    - ➢ Concrete Method – init(), destroy().

service()

- service method is a abstract method present in GenericServlet.
- service method takes ServletRequest object and ServletResponse Object as a argument.
- Whenever service() is called, it throws ServletExpection, IOException.

HttpServlet

- class is an abstract class present in **jakarta.servlet.http** package. Here the class does not contain any abstract methods.
- In Java, we can create a class as an abstract class, without any abstract methods, for the following two reasons.
  - To convey a message that a class has partial implementation/some dummy implementation for the functionalities.
  - To convey a message that a class has partial implementation, so we can't create object to a class.
- HttpServlet class extends **GenericServlet** class.
- HttpServlet class is given as an abstract class, to indicate the developers that it is an incomplete class.
- HttpServlet class has defined 7 doXXX() methods with some dummy implementation/logic, which doesn't fit for any application. [doGet(), doPost(), doPut(), doDelete(), doTrace(), doOptions(), doHead()]
- Hence, developers have to override the required doXXX() methods into their servlet classes.
- doXXX() takes HttpServletRequest object and HttpServletResponse object as a argument.

Parameters : Parameters are the data which is provided in the key and a value pair. Here, both the key and the value pair are String.

```
<form action= "url" method="post">
      <input type="text" name="key">
</form>
```

Note : The value entered by the user in the UI or form will be stored inside the key specified in name attribute.

req.getParameter()

- This method is used to read the form data or collect the data from the end user.
- getParameter() is present in the req ref object.
- getParameter() takes key as argument which will be assigned in name attribute in html file.
- It returns the value which is stored inside the key.

Syntax:

```
 String req.getParameter("key");
```

PrintWriter class

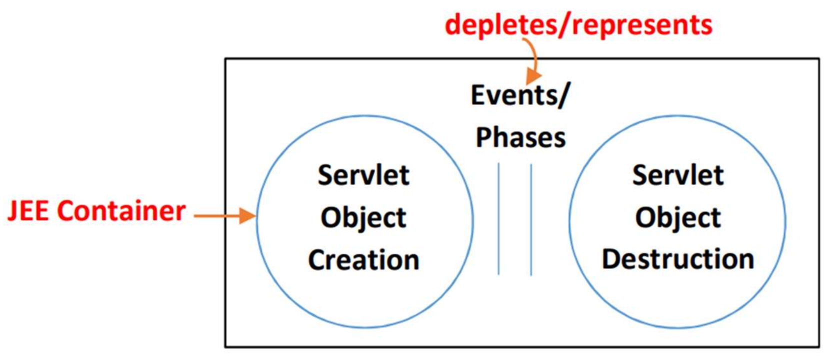
- PrintWriter is an abstract class present in java.io package.
- It is used to write any content in the web browser.
- As it is an abstract class, we can create a reference object by using getWriter() which is present in respone object.

Syntax:

```
java.io.PrintWriter out=resp.getWriter();
```

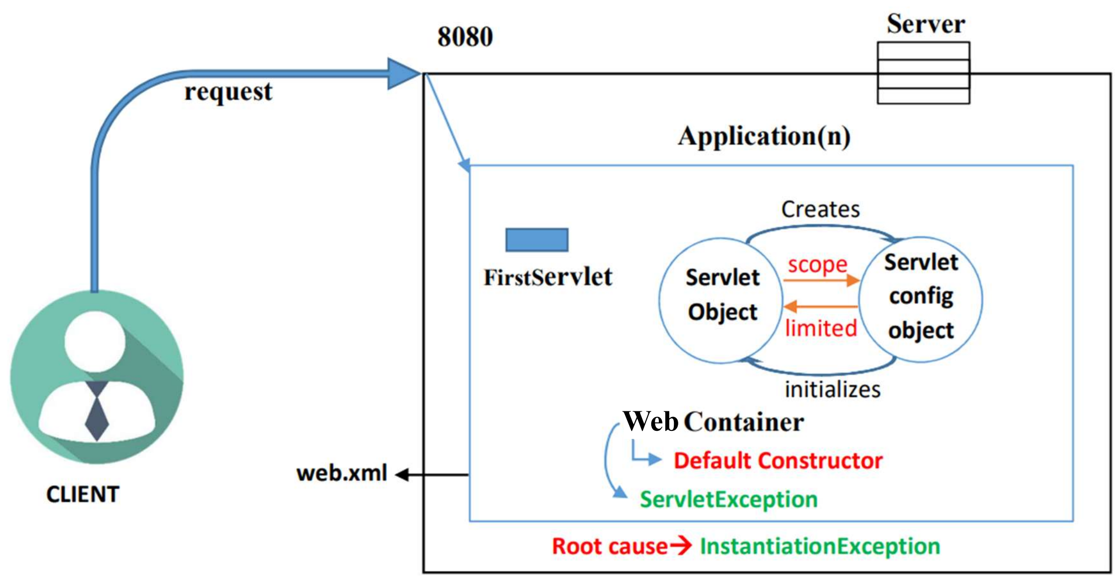
Servlet Life Cycle

- Servlet Life cycle starts/begin only after the creation of Servlet object.
- Servlet Life Cycle depletes or represents the events or phases which takes place from the servlet object creation until the servlet object destruction.
- The entire servlet lifecycle is managed by Web container or one of the responsibility of Web container is servlet life cycle management.



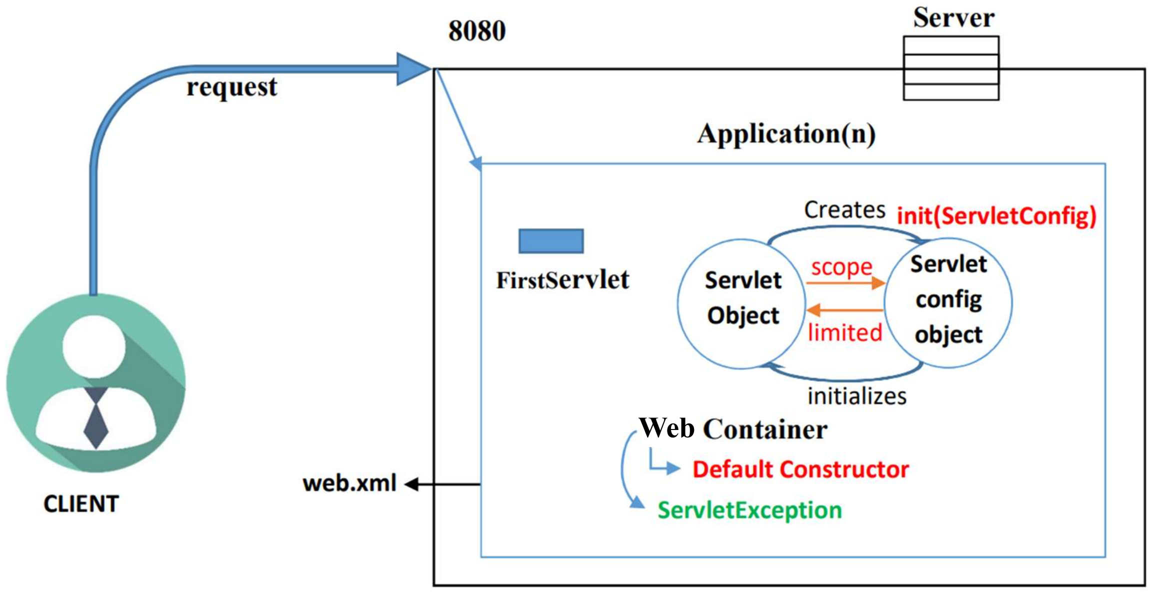
- There are different phases of Servlet Life Cycle. They are:
  - Instantiation Phase/ Object creation phase
  - Initialization Phase
  - Service Phase
  - Destruction Phase Event

Instantiation Phase



- In this phase, Servlet object has to be created.
- Whenever the client makes his first request to a servlet, one servlet object is created by the Web container by calling a default constructor of servlet and now the Servlet lifecycle begins.
- If the JEE container does not find the default constructor of servlet, then the Web container throws an exception called as ServletException and its root cause is InstantiationException (Object not created).
- Immediately after the servlet object creation, one servlet config object is created by the Web container which is used to initialize the resource of Servlet object.
- Hence the scope of the servlet config object is limited to that particular servlet object.

Initialization Phase

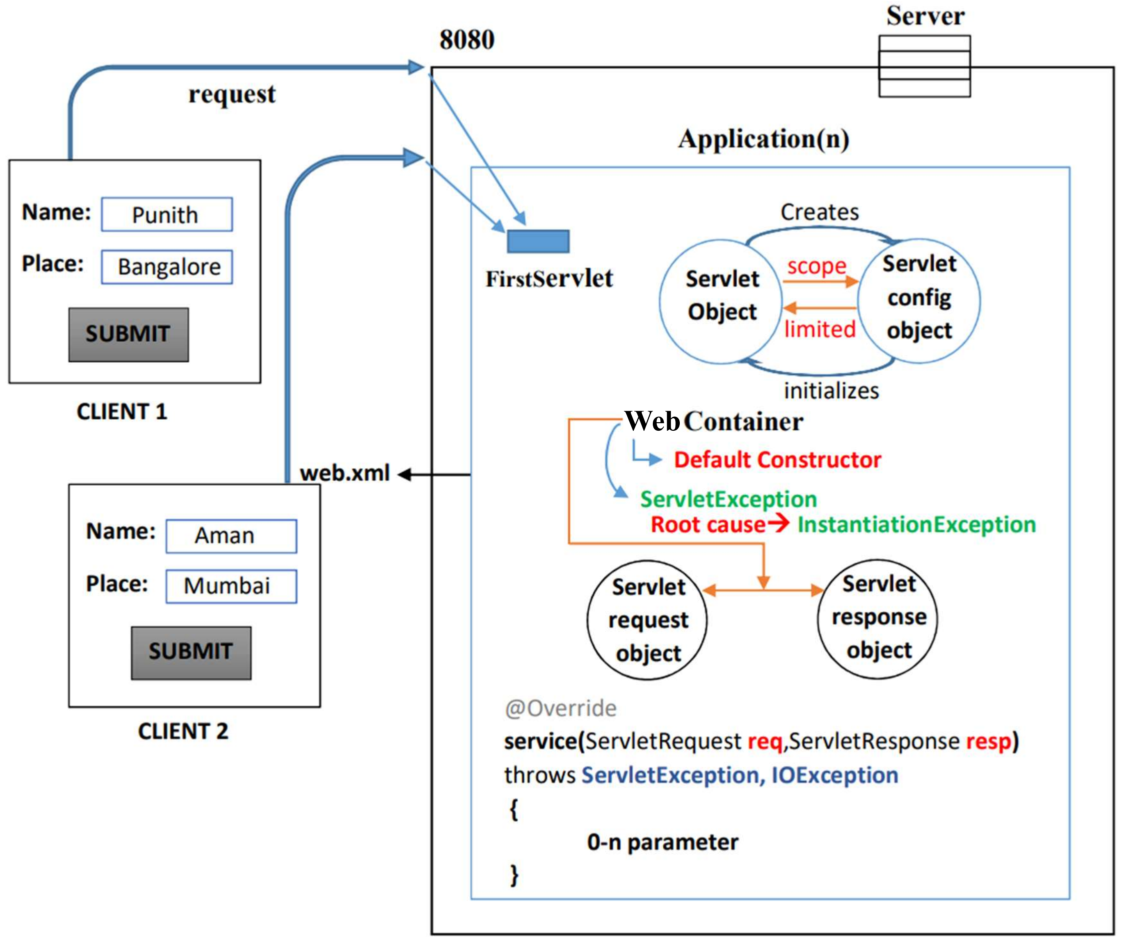


- In this phase, Servlet object has to be initialized.
- Servlet object has to be initialized by using init(ServletConfig) which takes ServletConfig as a parameter which is used to initialize the resource of servlet object.
- init(ServletConfig) is always called by Web container only once.
- If this phase fails, then the Web container throws an exception called as ServletException.

Service Phase

- In this phase, service() is called which is responsible for processing each and every client request.
- In this phase, the Web container creates one request and one response object for each and every client request including the first client request.
- By default, Servlet is multithreaded but can be made single threaded in two different ways namely:
  1. By writing a servlet class which implements marker interface by name called as Single Threaded model.
  2. By making service() as synchronize.
- By default, service() is also multithreaded i.e., one thread is created for each and every client request and service() will be executed.
- Whenever a client makes a second request or subsequent client request to the same servlet once again, only service() will be executed but servlet object will not be created once again.
- service() called by Web container for multiple times.

- If this phase fails, then Web container throws an Exception called as ServletException.



```
@Override
service(ServletRequest req,ServletResponse resp)
throws ServletException, IOException
{
        0-n parameter
}
```

Destruction Phase

- In this phase, the destroy() is called by Web container to close all costly resources.
- destroy() is called by Web container only once.
- If this phase fails, then the performance of application decreases.