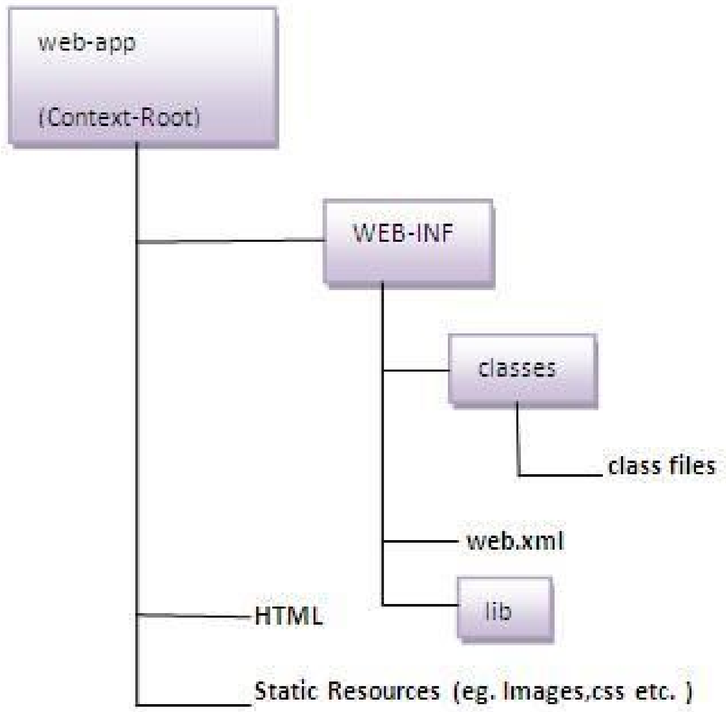
Servlets

Malintha Adikari

Java Servlet

- A **servlet** is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model.
- Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers.

Java Servlet

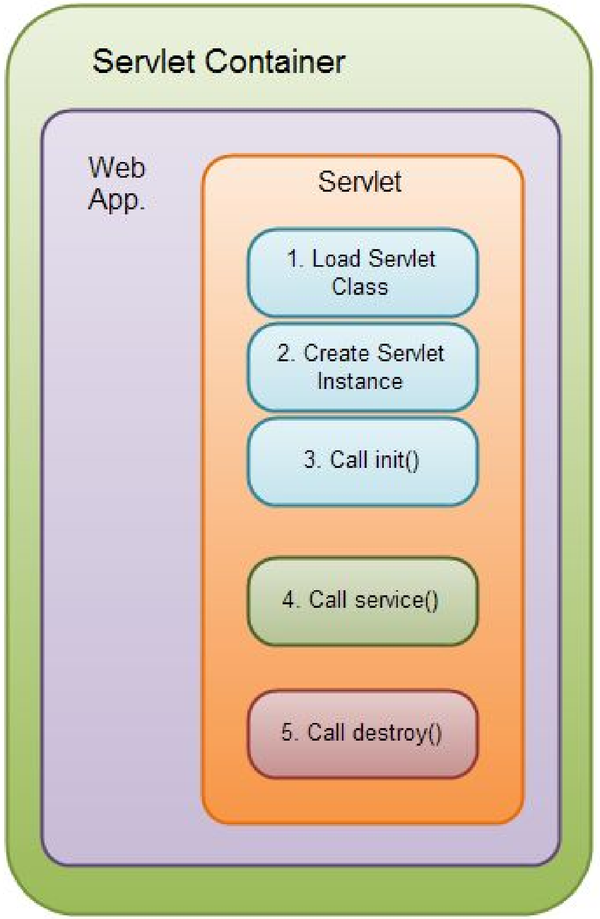


Servlet Lifecycle

A servlet follows a certain life cycle. The servlet life cycle is managed by the servlet container.

The life cycle contains the following steps:

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



- Step 1, 2 and 3 are executed only once, when the servlet is initially loaded. By default the servlet is not loaded until the first request is received for it.
- Step 4 is executed multiple times - once for every HTTP request to the servlet.
- Step 5 is executed when the servlet container unloads the servlet.

Servlet Container

- A **servlet container** is the component of a web server that interacts with Java servlets.
- Responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access-rights.
- **Tomcat** is primarily a Java servlet container.
- Tomcat is able to
  - receive requests from a client
  - dynamically compile a container-managed Java class to handle the request as specified in the relevant application context
  - Return the result to the client.

Servlet Container

- Servlets are expected to handle certain parts of the total data transaction process.
- Ex: Servlet code never listens for requests on a certain port, nor will it communicate directly with a client, nor is it responsible for managing its access to resources.
- These things are managed by the servlet container.
- This allows servlets to be re-used in a wide variety of environments, or for components to be developed asynchronously from one another

Servlet in Container

1. Tomcat receives a request from a client through one of its connectors.
2. Tomcat maps this request to the appropriate Engine for processing. These Engines are contained within other elements, such as Hosts and Servers
3. Once the request has been mapped to the appropriate servlet, Tomcat checks to see if that servlet class has been loaded. If it has not, Tomcat compiles the servlet into Java bytecode and creates an instance of the servlet.
4. Tomcat initializes the servlet by calling its init method. The servlet includes code that is able to read Tomcat configuration files and act accordingly

Servlet in Container

5. Once the servlet has been initialized, Tomcat can call the servlet's service method to process the request, which will be returned as a response.

6. Tomcat and the servlet can communicate through the use of listener classes during the servlet's lifecycle . Tomcat can retrieve and store these state changes in a variety of ways, and allow other servlets access to them

7. Tomcat calls the servlet's destroy method to smoothly remove the servlet. This action is triggered either by a state change to undeploy the servlet's Context or shut down the server.

Servlet Deployment

- Tomcat mainly has a Classloader Hierarchy and a Thread Pool.
- When a web application is deployed into tomcat, tomcat scans the Webapp , reads its deployment descriptor (web.xml or the equivalent) and decides that Servlets (and JSPs) need to be deployed and be made available.
- Servlets and JSPs are class loaded based on a class loader hierarchy

URL Mapping- Method I

Deployment Descriptor (web.xml)

- Define servlets as a part of a Web application in several entries in the J2EE standard Web Application deployment descriptor, web.xml.
- web.xml file is located in the WEB-INF directory of Web application.
- The first entry, under the root servlet element in web.xml, defines a name for the servlet and specifies the compiled class that executes the servlet.
- The second entry in web.xml, under the servlet-mapping element, defines the URL pattern that calls this servlet.

URL Mapping - Method I

Servlet Mapping

Servlet mapping controls how you access a servlet.

```
<servlet>
  <servlet-name>watermelon</servlet-name>
  <servlet-class>myservlets.watermelon</servlet-class>
</servlet>
```

URL Mapping - Method II

Annotation

- Represents the metadata.
- Deployment descriptor (web.xml file) is not required.
- Should have tomcat7 as it will not run in the previous versions of tomcat.
- @WebServlet annotation is used to map the servlet with the specified name.

```
@WebServlet(
        name = "MyOwnServlet",
        description = "This is my first annotated servlet",
        urlPatterns = {"/helloAnnotation", "/helloAnnotationExtension", "/helloWildcard/*"})
```

Thanks!

Any questions?