Saturday, December 25, 2021 10:49 AM

Servlet Life Cycle ---------- Is managed by Tomcat servlet web container

Servlet Object is Created -- constructor call

|

|

public void init () call -------write code for initializing data

|

|

public void service() call //generate dynamic html, process data , session management

OR

public void doGet() call

OR

public void doPost() call

|

|

public void destroy() call ---------- write code for cleanup activity, saving data ,writing log files

|

|

Servlet is destroyed (made available to GC by making unreachable)

_____

1.

Servlet MUST MUST MUST have NO-ARGS constructor (Either u don't give any constructor , then u get default , else if u give parameterized constructor then make sure that u give NO-parameter/NO-args constructor ---- OTHERWISE container cannot create object of SERVLET !!!!!!!

2. The object of the servlet is created when
   a. FIRST request arrives = LAZY initialization = BY DEFAULT
      PRO = memory is allocated only when object is needed , space saving when object not needed
      CON = first request will have slow response

   b. Servlet is DEPLOYED = EAGER initialization = <load-on-startup>1</load-on-startup>
      PRO = first request will be fast , Some initializations can be done before requests come
      CON = even though object may not be needed it will occupy RAM space .

3. ONLY one servlet object is created by the container !!!! The same object is shared for all the REQUESTS . Mostly servlets don't have user specific properties !!!

4. Init service destroy methods are called by the container on certain events -
   a. Just after the servlet object is created ---------- init is called--ONCE , when started or deployed
   b. Just before the servlet object is destroyed ----- destroy is called --ONCE, when stopped or undeployed
   c. For every request ----------service is called -- 0 to n times
   d. If we want to distinguish INCOMING request as GET and POST Http Request then instead of service we must override doGet or doPost--- for every get request doGet is called and for every post request doPost is called

---

Main???? ---- tomcat is a java program having main !!

---

web.xml PURPOSE ??? Deployment Descriptor =
programmer configurations are
communicated to container
through this descriptor

Writing DD using ANNOTATIONS !!!!
Annotations ----------

Annotation is a Sticker
It can be applied to class, methods, properties, constructors, interfaces !!!

One who applies the sticker is the programmer
One who reads the stickers and behaves in a different way with the methos or properties etc is the CONTAINER !!!

Example of Annotation ----------

```
class XYZ
{
    @Override
      Public boolean equals(Object o)
      {
        …
      }

}
```

Sticker = Override
Target = method
Applied by = programmer
Read by = CONTAINER ----- Java Compiler (if the method is annotated by Override , then it checks if the signature is same as super class method else give error)

```
@FunctionalInterface
interface OneMethodInterface
    {
        Public void m1();
    }
```

Sticker = FuntionalInterface
Target = interface
Applied by = programmer
Read by = CONTAINER ----- Java Compiler

(if the interface is annotated by FunctionalInterface then the compiler would check if the interface has only one method , if more or less then it would give error)

Example

```
@WebServlet(loadOnStartup = 1 , urlPatterns = "/TestServlet")
```

```
public class TestServlet extends HttpServlet {


	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
	ServletException, IOException {

		response.getWriter().append("Served at: ").append(request.getContextPath());
	}

}
```

Sticker = WebServlet
Target = Type = class/interface
Applied by = programmer
Read by = CONTAINER ----- Tomcat container during DEPLOYMENT

HW 1---

Modify the LoginServlet such that in the init method a hashmap of hardcoded users is created - as discussed in class
Use that map in the doPost method to check if login info is correct or wrong
|
|
Again modify the init method such that the usernames and passwords are read from a DB table user_info and added to hashmap (populate the hashmap from DB)

HW 2 ----

Write a static html
Enter book id : tf
Enter book name : tf
Enter cost : tf

Save ----------submit ------>AddBookServlet ---doPost ----write code to insert one record in the DB table

HW 3 ----

Write a Servlet SearchInput
Select an Id : dropdownlist of all the book ids from book table
Show-details ---submit ---------> FetchDetailsServlet -----fire query get all the details of the book with selected id and show the details on browser .

HW 4 --- Try out the life cycle examples done in class
Not adding default constructor
Lazy and eager
Init , destroy ,service
Init doGet destroy