Note:

GenericSevlet is an best example for "Adapter class design pattern".

init() is overloaded in GenericServlet.

Eg: FourthApp

Note:

By default response type/content type is "text/html"

There is no need to write html head and body tag , by default every thing you write will be in the html body tag.

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns="/test")

public class FourthServlet extends GenericServlet{

public void service(ServletRequest req , ServletResponse res)throws ServletException,IOException{

PrintWriter out = res.getWriter();

out.println("<h1 style = 'color : blue';> writing servlet using generic servlet </h1>");

out.close();

}

}

Behind the scenes

=================

In the above code 2 .class files will be used

a. FourthServlet.class

b. GenericServlet.class

=> Loading:: Container will load FourthServlet.class file for the url pattern ("/disp")

=> Instantiation:: Container will create an Object for FourthServlet.class ( for abstract class object cannot be created GenericServlet (Ac) ).

=> Initialization:: Container will call init(), First it will check in FourthServlet.class if not, it would check in GenericServlet

init() is available inside GenericServlet but it has 2 methods with the name

init(SC config)

init()

Can we override the init logic?

We can override, but it is a good practise to override only init(),but not init(SC config) becoz config is local variable in init(SC config),and the config variable memory would be gone once the control comes out of the init(SC config)

so better override init() but not init(SC config).

Internally implementation, in predefined code even though config is a local variable, it is reused as a instance variable as shown below.

If we try to override int( SC config ) connection w.r.t instance variable config will be broken, which is used again in int() method.

Code in GenericServlet

=====================

public abstract class GenericServlet implements Servlet,ServletConfig,Serializable{

private transient SC config;

public void init(SC config)throws SE{

this.config=config;

init();

}

public void init() throws SE{

;;;;;;

}

}

=> RequestProcessing phase :: Container will call service(req,resp) to provide response to the client.

First it will check in FourthServlet.class if not, it would check in GenericServlet.

service(req,resp) is available inside GenericServlet as abstract and we need to give the body of this method inside FourthServlet as shown in the above program.

=> ServletDelnstantion=> Contianer will call destroy() to perform De-Instantion action.

First it will check in FourthServlet.class if not, it would check in GenericServlet.

destroy() is not avaialble in FourtServlet.class so it would take from GenericServlet.class and it

will execute.

Note:

1. If our servlet class does not contains init() method

a. GenericServlet: init(SC)

b. GS: init()

c. User: service(req,resp)

2. If our servlet class contains init(SC) method

a. US: init(SC)

b. US: service(req,resp)

3. If our servlet class contains init() method

a. GS: init(SC)

b. US: init(SC)

c. US: service(req,resp)

why 2 init(),init(SC config) in GenericServlet?

init(SC config) -> container

init() -> developer

Which init method is best suited for developer?

init() => best suited for writing initialziation logic.