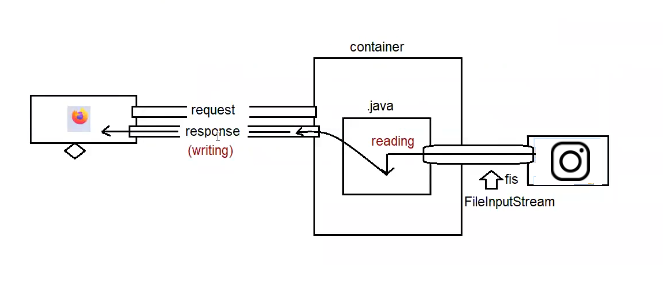
Working with ServletResponse/HttpServletResponse

1. public abstract PrintWriter getWriter() throws IOException;

To send character type of Response

2. To send binary information like videos, audio, images etc response will be through "Stream".

public abstract ServletOutputStream getOutputStream() throws IOException;



Eg: ResponseInfoApp

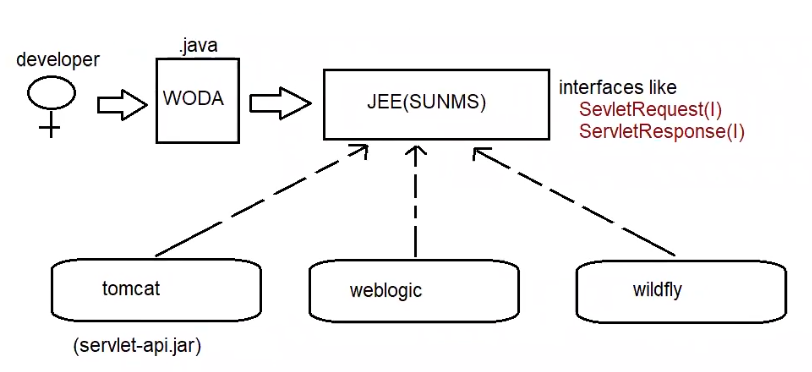
Note:

PrintWriter => only character type of data.

ServletOutputStream => we can send both character type and binary type of data.

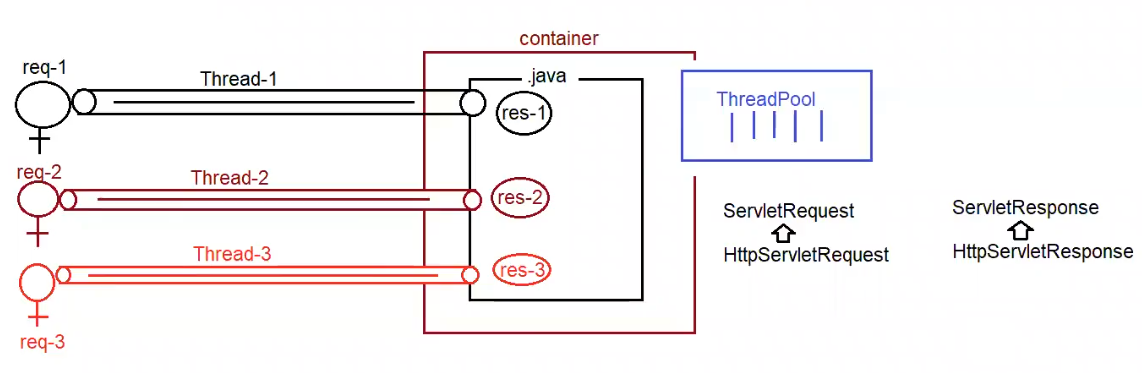
Q> In single application can we use both PrintWriter object and ServletOutputStream to send the response?

Not possible, it would result in "java.lang.IllegalStateException: getWriter() has already been called for this response



Multi-Threading technology used in the servlet:

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



Eg: RequestResponseApp



ServletContext(I) vs ServletConfig(I)

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

1. Loading

2. Instantiation === > Object obj = Class.forName(String urlPattern).newInstance();

Object will be created for the loaded class using Zero parameterized constructor.

3. Initialization ==== > For the created Servlet object, we can initialize the value using init(ServletConfig config)

ServletConfig(I)

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

This object will be used to initialize the values for the loaded servlet.

How to initialize the value of ServletConfig object?

To initalize the value of ServletConfig object, we need to configure the container.

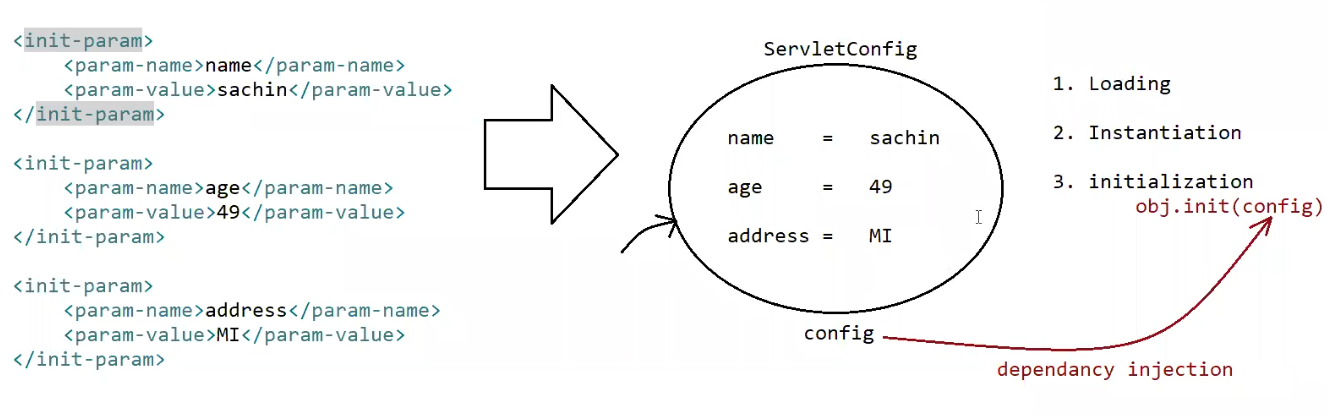
Container can be configured in 2 ways

a. XML

b. Annotation

Eg: ServletConfigApp

//based on xml



Methods

========

public Enumeration<String> getlnitParameterNames();

public String getInitParameter(String key);

Note:

=> ServletConfig object is unique w.r.t every Servlet.

=> ServletConfig object stores the data in the form of Key,Value pair.

=> Assume we want the dbconfiguration details for a servlet like

url => jdbc:oracle:thin:@localhost:1521:orcl

username=> practice

password => 1234

After adding the jar, jars will available to jdks/w configure for eclipse, not for tomcat server.

To make those jars available for tomcat server(catalina container) we need to put in deployment assembly(/WEB-INF/lib)

Eg: servletDBCommunication

1.why there is no main method in Servlets ??

Answer: JVM has be designed to make a call to main method with the following prototype to start the execution

public static void main(String[] args)

If the application is standalone application then jvm will come into picture so we need main method to start the execution.

if the application is webapplication, then container will execute our java code based on its life cycle actions like

a. Loading

b. Instantiation

c. Initalization

d. RequestProcessing

e. De-Instantiation

Since jvm does not play vital role to start the execution we don't need "main method" in Servlets.

2.Eclipse Shortcuts

ctrl+2, L => to generate the return type of method, constructor, ….

Alt+ctrl+arrowup -> to duplciate the lines

Alt+arrowup -> to move the line upwards

Alt+arrowdown-> to move the line downwards

Alt+shift+m -> To move the selected lines to be a part of the method.

Alt+shift+R -> To replace the varaible name in every place of java code.

Alt+shift+s,r => To generate setters and getters

Alt+shift+s,S => To generate toString()

Alt+shift+s,v => To do override for any methods of the class.

cntrl+shift+o => To organize the import statements/import particular class

ctrl+a => To select everything

ctrl+shift+l=> To perform indentation.

ctrl+shift+f => To perform formatting.

ctrl+shift+/=> To perform commenting.

ctrl+shift+\ => To perform uncommenting.

ctrl+shfit+t => To open particular class from the jar.

ctrl+ o => To list all the methods of the class.

ctrl+spacebar => To give assistance for our code.

Ctrl+shift+l => give the shortcut list in eclipse

Ctrl+d => to delete the line

Ctrl+2,f => to generate instance variable and l to generate local variable

Ctrl+shift+x => to make uppercase

Ctrl+shift+y => to make lowercase

3.ServletConfig InitParameters Using

1.Xml

2.Annotations

a. @WebServlet (urlPatterns = {}, loadOnStartup= 10,

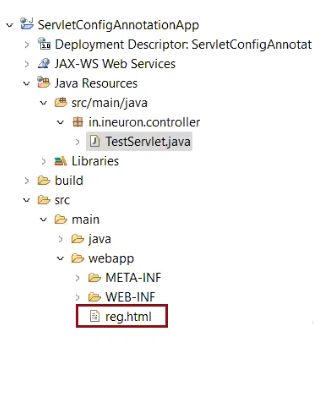
initParams = {

@WeblnitParam(name = "url", value = "jdbc:mysql:///octbatch"),

@WeblnitParam(name = "user", value = "root"),

@WeblnitParam(name = "password", value = "root123")

})



Eg: ServletConfigAnnotationApp

Note:

Servlet initialization parameters are key-value pair where both key and value are of type String.

From the Servlet we can access these parameters but we can't modify.

since we can't modify we just have only getXXXX() but not setXXXX().

So we say Servlet Initialization parameters as "Deploy time constants".

For every Servlet we will have only one ServletConfig object to hold its configuration information.

ServletContext(I)

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

For every webapplication web container will create only one ServletContext object to hold the configuration details.

By using context object we can get configuration information like context paramters, requestdispatcher etc ...

Assume there are 3 servlets and for all the servlets if the configuration details is common can we keep in config object?

Ans. we can keep, but it is not a good practise because if we keep the data inside Context object it will be available to all the servlets of the application.

How to keep the data in ServletContext object?

Ans. We can keep in only 1 way that is through XML.

Annotation support not available because when container gets started, but the ServletContext object is created when no java code is coming into picture i.e is servletcontext object is created when container gets started.

Eg: ServletContextApp

web.xml

<web-app>

<context-param>

<param-name>jdbcUrl</param-name>

<param-value>jdbc:mysql:///octbatch</param-value>

</context-param>

<context-param>

<param-name>user</param-name>

<param-value>root</param-value>

</context-param>

<context-param>

<param-name>password</param-name>

<param-value>root123</param-value>

</context-param>

<servlet>

</servlet>

</web-app>

Inside servlet we can get the ServletContext data in 2 ways

a. ServletConfig config = getServletConfig();

ServletContext context= config.getServletContext();

methods

public String getInitParameter(String name)

public Enumeration getlnitParamterNames()

b. ServletContext context = getServletContext();

methods

public String getInitParameter(String name)

public Enumeration getInitParamterNames()

Note:

when 2 servlets have load-on-startup then

a. lower load-on-startup will get chance first for executeion

when 2 servlets have same load-on-startup then

a. it depends on container(not in the hands of the programmer)

if we give negative value for load-on-startup then the container will not load any of the servlet.

we should give only positive value (zero can also be given)

Difference b/w ServletContext vs ServletConfig object

ServletContext

a. For every webapplication, container will create only one ServletContext object to hold the data at application level.

b. It will be created at the time of application deployment and destroyed at the time of application undeployment.

c. <context-param>

<param-name></param-name>

<param-value></param-value>

</context-param>

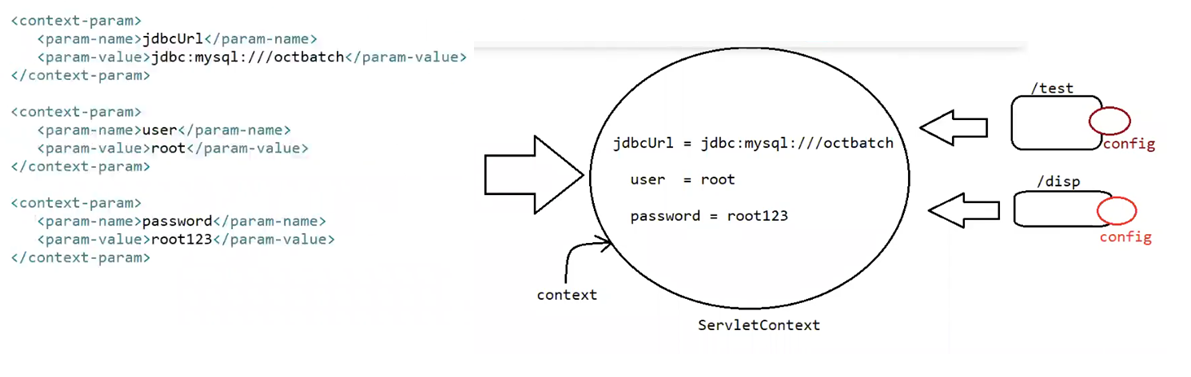
d. 2 ways to get the Context Object

ServletContext context = getServletContext();

ServletConfig config = getServletConfig();

ServletContext context = config.getServletContext();

e. Configuration can be done only in one way through XML



ServletConfig

a. For every Servlet, container will create only seperate ServletConfig object to hold the data at servlet level.

b. It will be created at the time of Servlet object creation and destroyed at the time of Servlet Object Destruction.

c. <init-param>

<param-name></param-name>

<param-value></param-value>

</init-param>

d. Approach to get ServletConfig object

ServletConfig config = getServletConfig()

e. Configuration can be done in 2 ways

a. XML

b. Annotation