Dt : 7/2/2025

faq:

Servlet Life-Cycle:

=>Servlet Life-Cycle demonstrates different states or stages of Servlet-Program from Starting to ending.

=>The following are some important stages of Servlet-Program:

1.Loading process

2.Instantiation process

3.Initialization process

4.Request Handling Process

5.Destroying Process

1.Loading process:

=>The process of identifying the Servlet-program based on url-pattern and loading for execution is known as Loading Process.

2.Instantiation process:

=>The process in which the Servlet-program automatically instantiated is known as Instantiation process.

Note:

=>After Instantiation process,execution-controls will identify the following LifeCycle methods:

(i)init()

(ii)service()

(iii)destroy()

3.Initialization process:

=>The process of making the programming components(Bean Objects,DAO Objects and Services) ready for service()-method is known as Initialization process.

=>we use init()-method to perform Initialization process.

4.Request Handling Process:

=>The process of accepting the request and providing the response,is known as Request Handling Process.

=>we use service()-method to perform Request Handling Process.

5.Destroying Process:

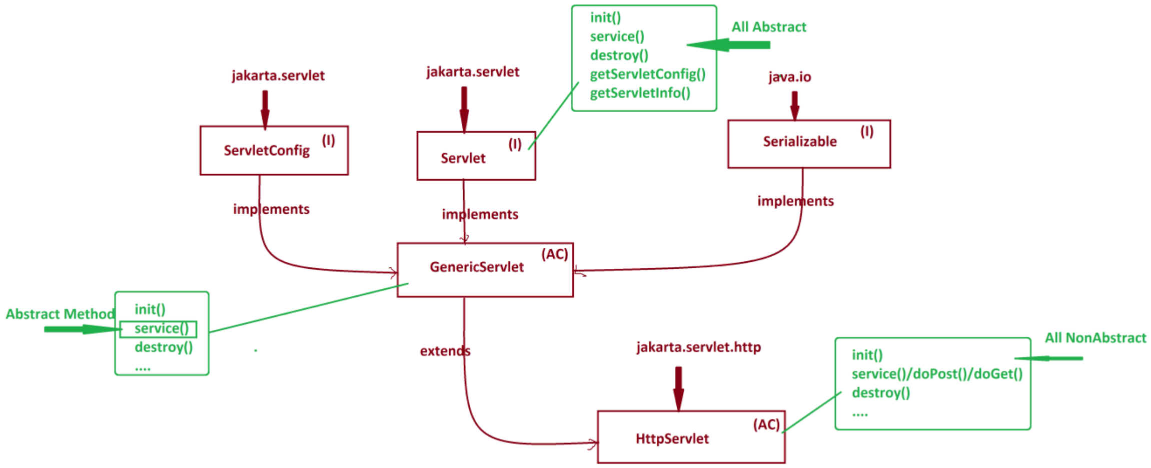
=>The process of closing the opening Components and services is known as Destroying process.

=>we use destroy()-method to perform Destroying process.

--

****imp***

Hierarchy of Servlet-API:



=>In the process of constructing Servlet-Program,we use any one of the following:

Model-1 : Implementing from 'Servlet-Interface'

Model-2 : Extending from 'GenericServlet-AbstractClass'

Model-3 : Extending from 'HttpServlet-AbstractClass'

Model-1 : Implementing from 'Servlet-Interface'

=>when the Servlet-program implemented from 'Servlet-Interface',then we have to construct body for all abstract methods.

Model-2 : Extending from 'GenericServlet-AbstractClass'

=>when the Servlet-Program extended from 'GenericServlet-AbstractClass',then we must construct body for only service()-method and 'init() and destroy()' are optional

Model-3 : Extending from 'HttpServlet-AbstractClass':

=>when the Servlet-program extended from 'HttpServlet-AbstractClass',then all methods are optional methods.

init()

service()/doPost()/doGet()

destroy()

Note:

service() : method will accept both POST and GET requests

doPost() : method will accept only POST request

doGet()   : method will accept only GET request

Note:

=>The Object generated from Servlet-Interface is NonSerializable Object

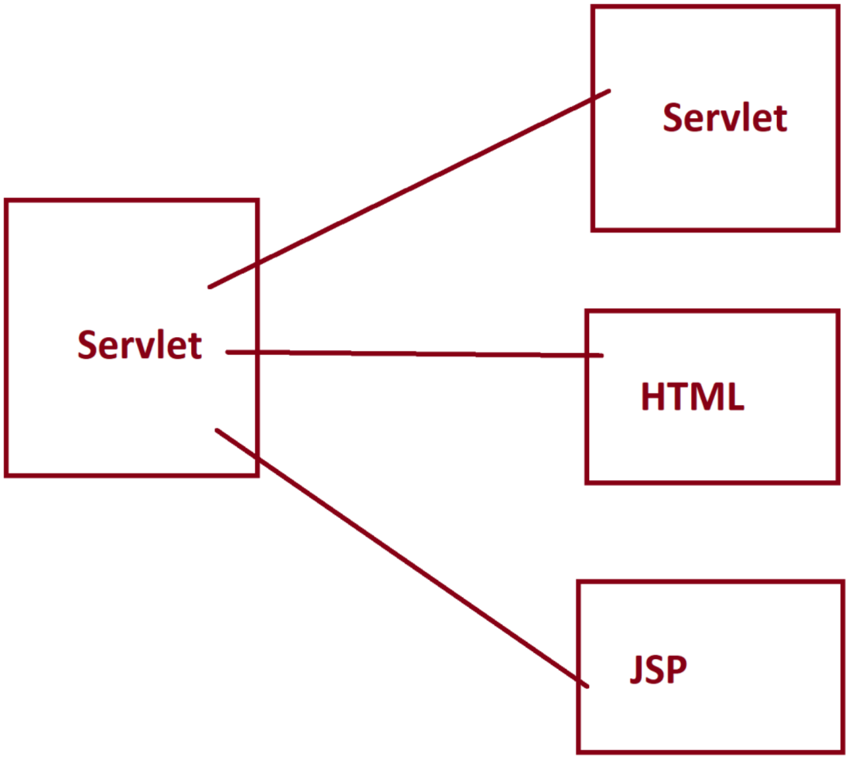
=>The Objects generated from GenericServlet and HttpServlet are Serializable Objects

===

****imp***

'RequestDispatcher' in Servlet Programming:

=>RequestDispatcher is an interface from jakarta.servlet package and which is used for Servlet Communication like Servlet-Servlet Communication,Servlet-HTML Communication and Servlet-JSP Communication.



=>Servlet Communications are categorized into two types:

1.Forward Communication Process

2.Include Communication Process

1.Forward Communication Process:

=>In Forward Communication Process,the Servlet-1 will take the request and forwards the the request Servlet-2,in this process Servlet-2 will provide the response.

=>This Servlet-2 can be replaced with HTML/JSP.

=>we use forward()-method from 'RequestDispatcher' to perform Forward Communication

process.

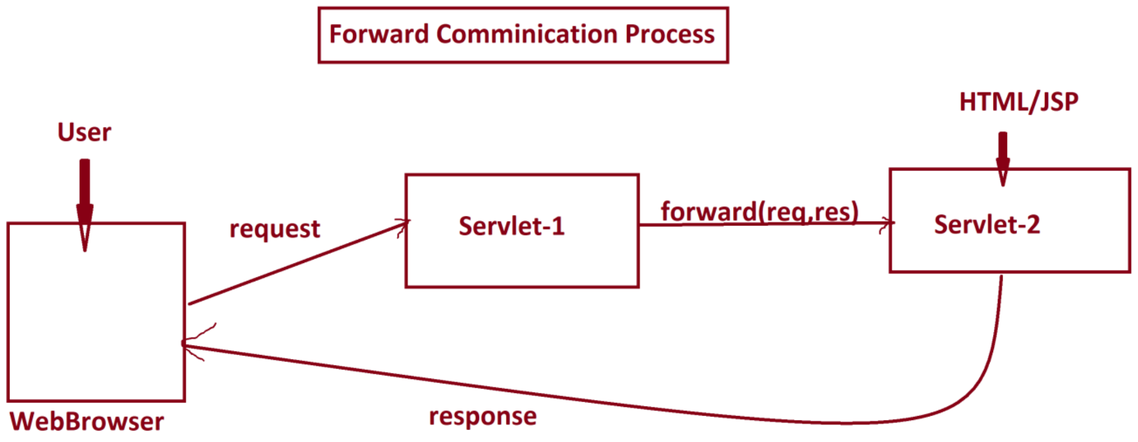
Method Signature:

public abstract void forward(jakarta.servlet.ServletRequest,

jakarta.servlet.ServletResponse) throws jakarta.servlet.ServletException,

java.io.IOException;

Diagram:



2.Include Communication Process:

=>In Include Communication Process,Servlet-1 will take the request and generate the response,but the response is included with the response of Servlet-2

=>This Servlet-2 can be replaced with HTML/JSP

=>we use include()-method from 'RequestDispatcher' to perform Include Communication process.

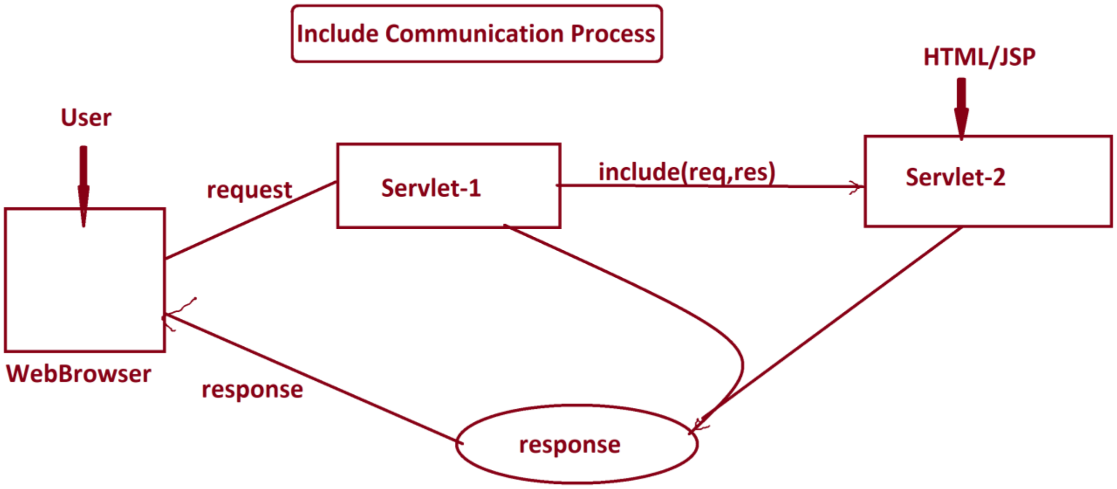
Method Signature:

public abstract void include(jakarta.servlet.ServletRequest,

jakarta.servlet.ServletResponse) throws jakarta.servlet.ServletException,

java.io.IOException;

Diagram:



=>we use getRequestDispatcher()-method from 'ServletRequest' to create implementation

Object for 'RequestDispatcher-Interface'

Method Signature:

public abstract jakarta.servlet.RequestDispatcher getRequestDispatcher(java.lang.String);

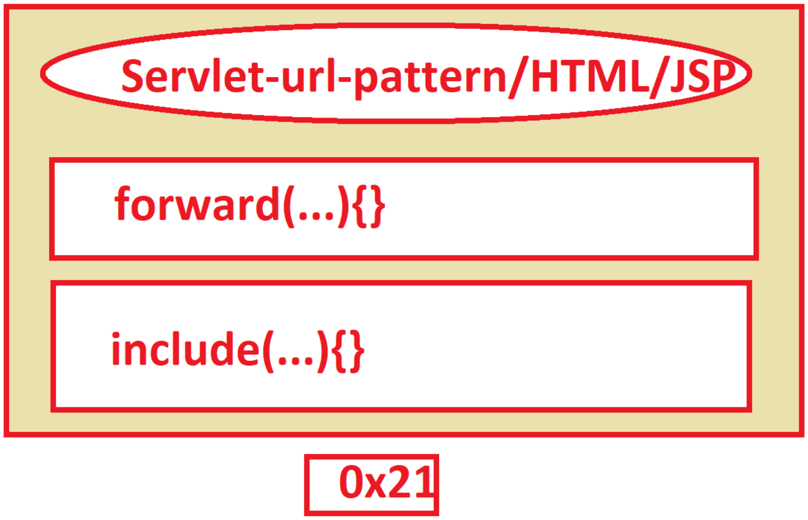
syntax:

RequestDispatcher rd = req.getRequestDispatcher("Servlet-url-pattern/HTML/JSP");

rd.forward(req,res);

rd.include(req,res);

Diagram:



```
RequestDispatcher rd = req.getRequestDispatcher
("Servlet-url-pattern/HTML/JSP");
```