Servlet Technology

Servlet technology used to create dynamic Web applications.

Before Servlet Technology there was a common Gateway interface technology to develop dynamic web applications.
In the early days of web application we had all static web applications and to make web applications as dynamic there was a first Technology introduced was CGI.

With CGI Technologies there are two major drawbacks
1.CGI engine creates a new process and a new object of peril program for each request
It increases the burden on the server.
2.CGI engine is not a capable to authenticate user it means CGI engine Canal power security
Intu overcome above drawbacks send Microsystem released Servlet technology to develop dynamic web applications;

A Servlet container creates only one process for web applications and one object for Servlet and creates a separate thread for each request, so it reduces burden on the server.

A Servlet container is capable of authenticating and authorizing user it means Servlet container provide security.

CGI Technology	Servlet Technology
1.CGI technology follows process model	1.Servlet technology follows Thread model
2.Due to the process model, the burden on the server will increase.	2.Due to this thread model, the burden on the server will be reduced.
3.CGI is not a secured technology	3.Servlet is Secure technology
4.CGI is not a scalable technology	4,Servlet is Scalable technology

Servlet:

1. Servlet is a platform independent java class,which runs in a server and provides responses to multiple clients over the internet.
2. We can also define a servlet as It is a server side program which increases functionality of the server.

How to create a Servlet?

Servet Api provided three different ways to create servlet class

1.by implementing our class from **Servlet** interface.

2.by extending our class from the **GenericServlet** class.

3.by extending our class from the class.**HttpServlet** class**.**

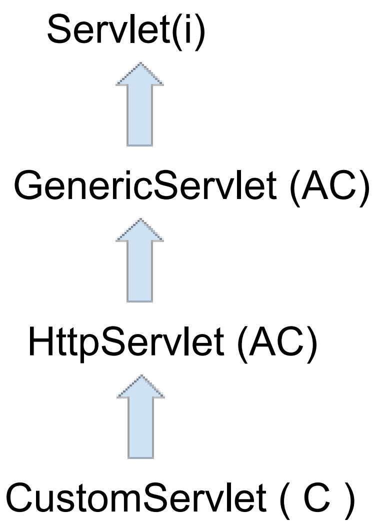
Servlet API provides three packages ,mainly

1.javax.servlet.*

2.javax.servlet.http.*

3.javax.servlet.annotation.

Each package contains classes and interfaces. Except Servlet interface all remaining interfaces,implementation classes will be provided by server vendors.



Servlet(i)

GenericServlet (AC)

HttpServlet (AC)

CustomServlet (C)

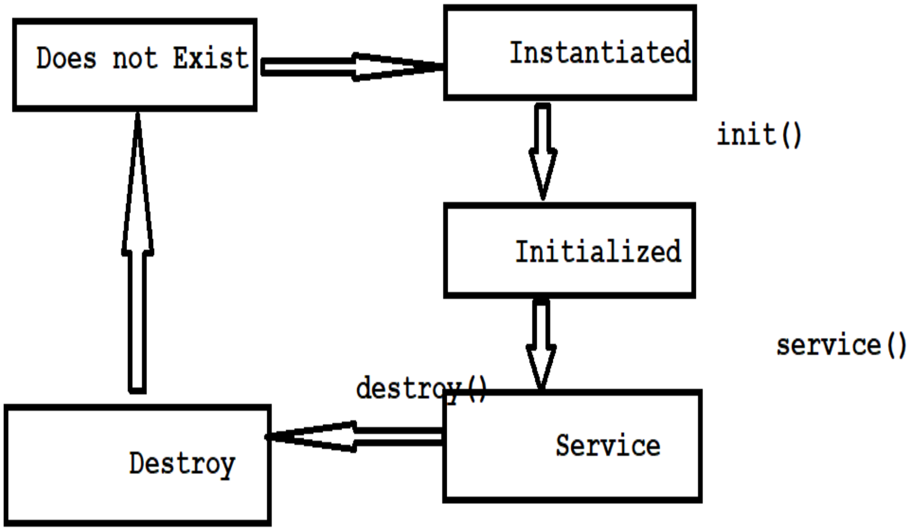
Servlet interface and GenericServlet both are given in javax.servlet package but HttpServlet is given under javax.servlet.http package.

1. Generally we don’t implement Servlet interface because it has five abstract methods and if we implement that interface then we must override all five abstract methods.
2. Generic Servlet is an abstract class which contains one abstract method,so if we extend GenericServlet we should override one abstract method only.

3. HttpServlet is also an abstract class but it does not contain any abstract methods. In Java an abstract class may or may not contain abstract methods,but if a class contains at least one abstract method then that must be declared as abstract class.

Servlet LifeCycle:

- Servlet is a java class,its lifecycle is managed by ServletContainer.
- Servlet container waits for First Request,to create an object of a Servlet class.before an object is created a servlet lifecycle state us **DOES NOT EXIST.**
- When First Request is arrived then the container creates an object of servlet,it means a Servlet is **INSTANTIATED**.
- The servlet container calls init() ,which means Servlet is **INITIALIZED.**
- **For** every request container calls service() method ,it means servlet providing service to the client.
- When a Servlet object is not more required in the servlet the container calls destroy() method ,here the servlet object is **destroyed**.
- When a servlet is destroyed then it comes back to a state that **DOES NOT EXIST** state.



Servlet Interface:

1. Servlet interface has five methods (3 lifecycle methods + 2 Non-life cycle methods)

Life cycle methods:

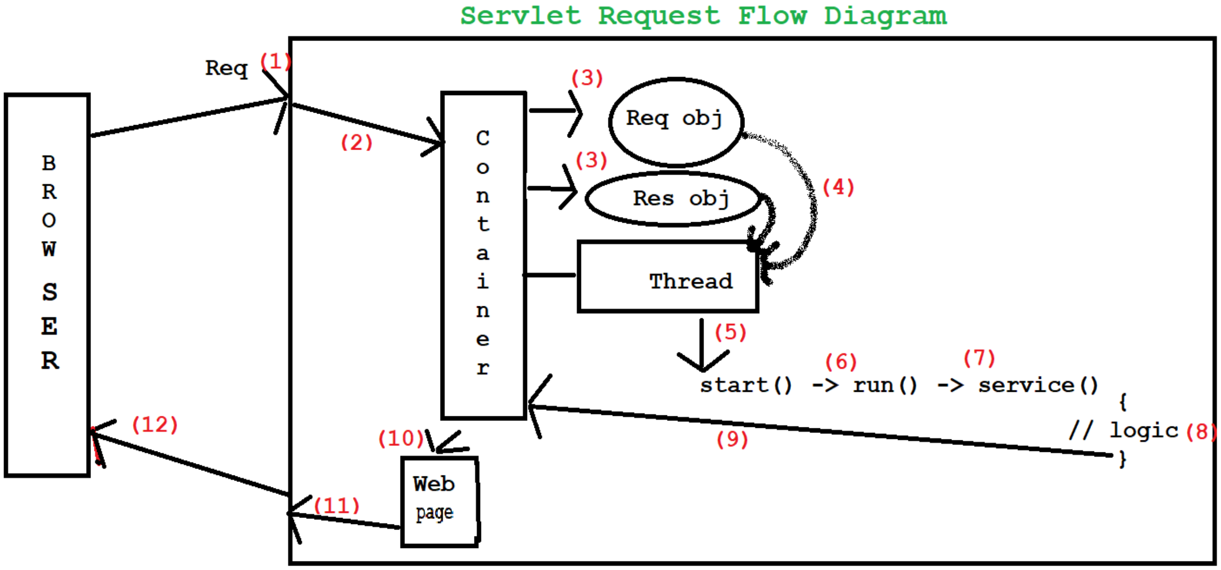
- public void init(ServletConfig config) throws ServletException
- public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException;
- public void destroy();

Non-lifecycle methods.

- public String getServletInfo();
- public ServletConfig getServletConfig();

2. Servlet container calls init() and destroy() method for only once and service() calls for each client request.
3. Lifecycle methods' servlet interface are provided in javax.servlet interface.

Servlet Request flow DIagram:



1.When a request is sent to a servlet from the browser it is first stopped at the server.
2. Server forward a request to the container.
3. Containers create Request and Response objects and a thread.
4. Containers assign request and response objects to the thread.
5,6.Container start thread and start() method calls run() method.
7.run method calls service() method
8.service() process request logic and write output into response object.

9.Container takes the response object from the service method.
10. The response will convert to a webpage.
11.Conatiner transfers the web page to the browser as a response.
12.brower displays   the response as a page.

NOTE: When response is sent back to the browser then immediately the container removes request,response and thread objects.

ServletConfig Object:

- ServletConfig is an interface of javax.servlet package.
- ServletConfig Object means its implementation class Object.
- The implementation class for interfaces of servlet technology will be provided by server vendor(Tomcat,jboss…etc)
- Based on our requirement we have to take instance variables in our class.to initialize data members with dynamic values we need parameterized constructor.
- In Servlet ,if we define a parameterized constructor then there is no use because we cannot directly call a parameterized container and Container also doesn’t call a parameterized constructor.
- To Initialized data members with dynamic values a lifecycle method has been provided called init().
- To pass dynamic values to data members Servlet technology has provided **<init-param>** tag in web.xml.
- An init parameters contains a name and value ,it can be configured under in **<servlet>** tag

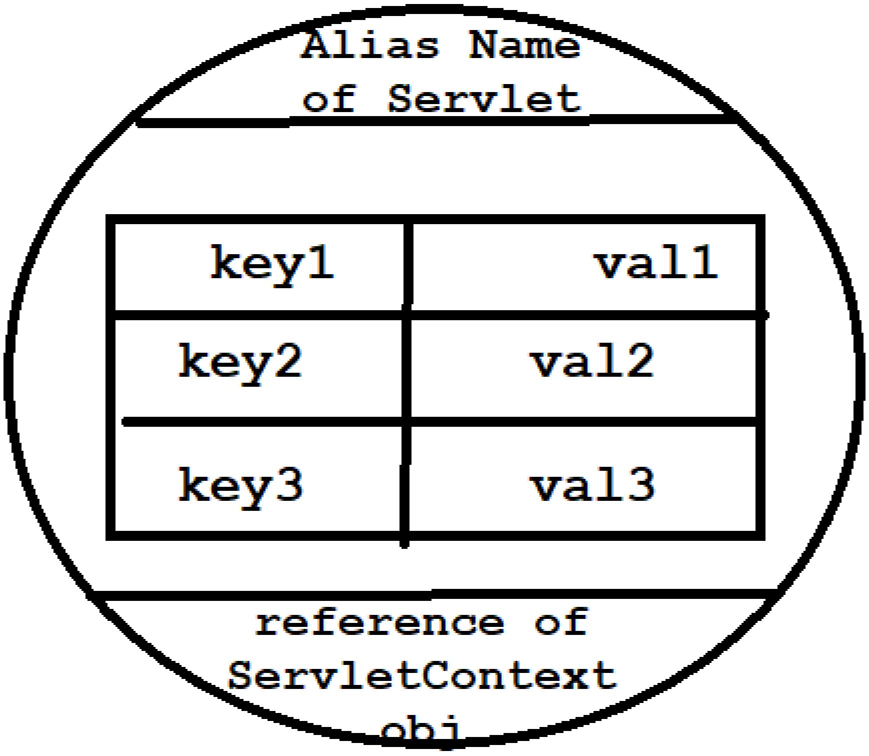
```
<servlet>
   <servlet-name>aliasName</servlet-name>
   <servlet-class>FullyQualifiedClassName</servlet-class>
   <init-param>
```

```
        <param-name></param-name>
       <param-value></param-value>
 </servlet>
```

- Before calling the lifecycle init() method of servlet,Servletcontainer creates config objects and stores all init parameters to config objects.

- Config object internally maintains a Map object for storing init-parameters.
- Servlet Container will also store a Servlet alias name and reference of **ServletContext** in Config Object.

ServletConfig Obj



key1	val1
key2	val2
key3	val3

- ServletContainer creates a config object after the constructor is executed and before lifecycle init() is called.so Config Object is not accessible in the constructor, but accessible from init() method.
- In init(),we can read init parameters values from the config object and we can assign to the variables.
- To init parameters from the config object we need to call the below method.

  ```
  config.getInitParameter("key1");
  ```

- To read Servlet Alias Name of Servlet from Config Object,we need to call the method below.

  ```
  String alias= config.getServletNames();
  ```

- To get ServletContext Object from config we need to call the below method.

  ```
  ServletContext ctx= config.getServletContext();
  ```

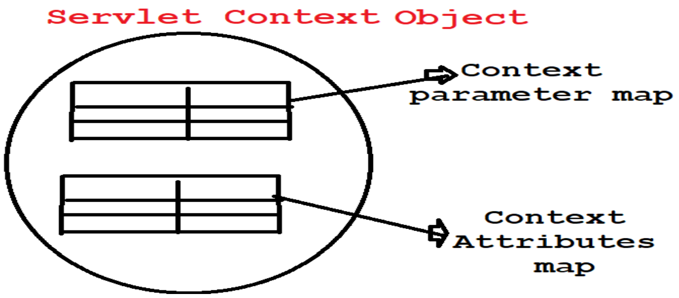
ServletContext Object:

- servletContext is an interface of javax.servlet package.
- servletContext object means it is an implementation class object.
- When a web application is deployed in a server successfully then servletContainer first creates a servletContext object.
- For one web application one context object is created it means the ratio between web application and context object in a server is 1:1.
- A context object is a global object for multiple servlets in a web application. The ratio between a context object and servlets in web-application is 1:n.
- context object is mainly used for sharing data across multiple servlets of a web-application.
- In a web-application we can have multiple servlets and sometimes we want to initialize the data members of multiple servlets with the same values.
- we can configure init parameters in web.xml under(servlet) tag. for each servlet separately.
- If we configure init parameters separately then the size of the web.xml file will be increased.
- A solution provided by servlet technology is, configure context parameters in web.xml.

- context parameters are configured directly under root tag (web-app) but not under(servlet) tag.so, context parameter will become global to all servlets.
- servlet container stores the context parameters of web.xml under context object. and multiple servlets can read the values of context parameters through context object.
- context parameters can be configured in web.xml like the following.

```
  <context-parameter>
                  <parameter-name>xxx</parameter-name>
                  <parameter-value>xxx</parameter-value>
     <context-parameter>
```

- A context can share its value with remaining servlets in a web application by storing that value in a context object.
- A servlet can store value in a context object,by converting it into an attribute. It means a key is required to identify the values.
- context object has a method setAttribute(), to store an attribute in context object.
- context object internally maintains two map objects,one for storing the context parameters stored in web.xml other for storing attributes.
- A servlet can read an attribute from context object by calling getAttribute() method similarly a servlet can remove attribute from context object by calling removeAttribute() method.



Exception Handling in Servlet:

1. When a servlet is processing a request of a client, if any exception occurs then by default the exception will be displayed as a web page on the browser.

2. It is not a good way to show an exception on browsers to an end user because the end user doesn't know of the technology and cannot understand the exception.
3. In the pace of displaying exceptions,we need to display an understandable message to the end user.
4. In servlet technology,exception handling can be done in two ways.
   A.Programmatic Approach
   B. Declarative Approach

Programmatic Approach:

- In a servlet, we need to put the source code in try-catch block.
- If an exception occurred then in the catch block, we need to redirect the request to the html page which displays a message in the place of exception on the browser.
  Ex:-

```
 public class MyServelet extends GenericServlet {
        public void service(req,res) throws SE,IOE
         {
               try{

                   // code

               }catch(Exception e){
                         res.sendRedirect("sorry.html");
                }
         }
   }
```

- With a programmatic approach there are two drawbacks.
  I. If the html file name changed from sorry.html to error.html then we need to modify the multiple servlets of applications ,then again recompile ,test,redeploy of application is required.
  II. If applications have multiple servlets then we need to add try-catch blocks for each servlet.

Declarative Approach:

1.In declarative approach we can configure <error-page> tag in web.xml.

2. If an exception occurs in a servlet then the container will search for the <error-page> tag for that exception.

Ex:

```
<error-page>
        <exception-type>java.lang.NullPointerException</exception-type>
        <location>/sorry.html</location>
  </error-page>
```

3. We can also configure more than one <error-page> tags in web.xml.

Ex:

```
<error-page>
        <exception-type>java.lang.Exception</exception-type>
        <location>/error.html</location>
  </error-page>
```

4. If NullPointerException occurs then container forwards request to sorry.html and if any exception occurs then request is forwarded to error.html.

5. A Servlet can also send an error code as response for a request, by calling send error() method.

6. The Servlet container creates a web page for that error code and the container sends that page to the browser.

7. Suppose if we want to show some other html page on the browser instead of error code then we can configure <error-page> tag in web.xml.

Ex:

```
<error-page>
        <error-code>404</error-code>
        <location>/NotFound.html</location>
  </error-page>
```

8. We can find the below differences between programmatic and declarative.

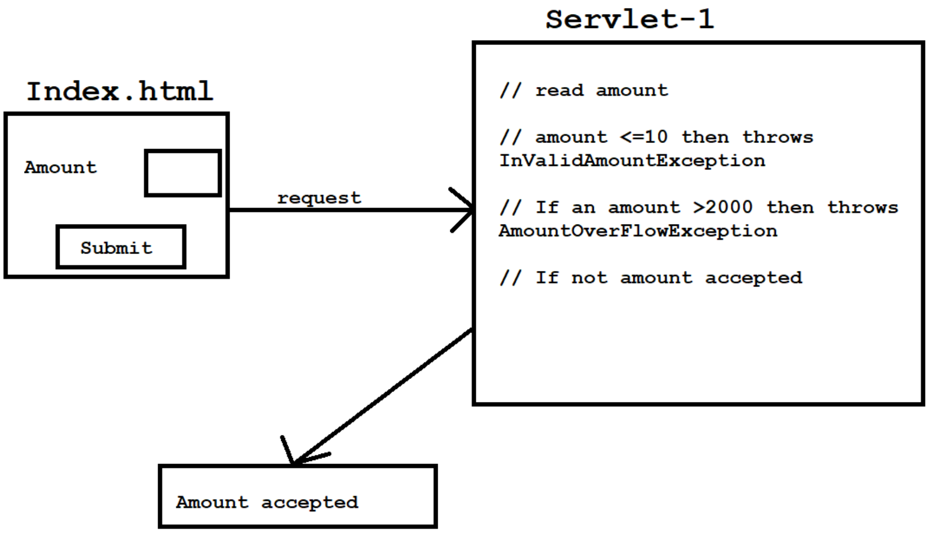
Programmatic	Declarative
1. In programmatic we must try and catch blocks in every servlet	1. In declarative we add <error-page> tag in web.xml

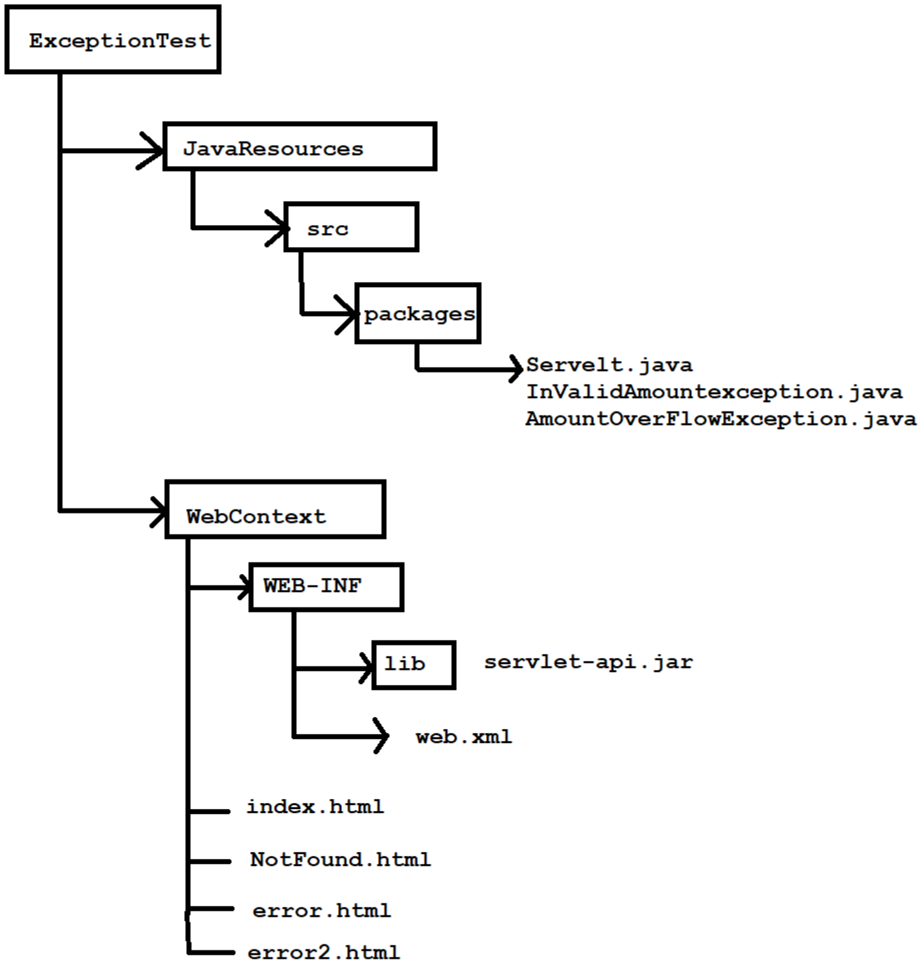
2. If any changes are made in exception handling then we need to recompile source code, reload and restart the server.	2. No need to recompile, reload and restart
3. In the Programmatic approach we can only handle exceptions.	3. In declarative we can handle both exceptions and error codes.

9. In a web.xml, we can make one servlet as error servlet for the remaining servlet of the application. If any exception occurs then the container will forward the request to error servlet.

Ex:

```
 <error-page>
         <exception-type>java.lang.Exception</exception-type>
         <location>/errorservlet</location>
   </error-page>
```


Servlet-1
Index.html
Amount
Submit
request
// read amount
// amount <=10 then throws InValidAmountException
// If an amount >2000 then throws AmountOverFlowException
// If not amount accepted
Amount accepted



Http Status Codes:

These are called response codes ,which tells about the status of the request.This status codes are divided into 5 Types.

1XX - Informational
2XX - Success
3XX - Redirection
4XX - Client SIde Error
5XX - Server side Error

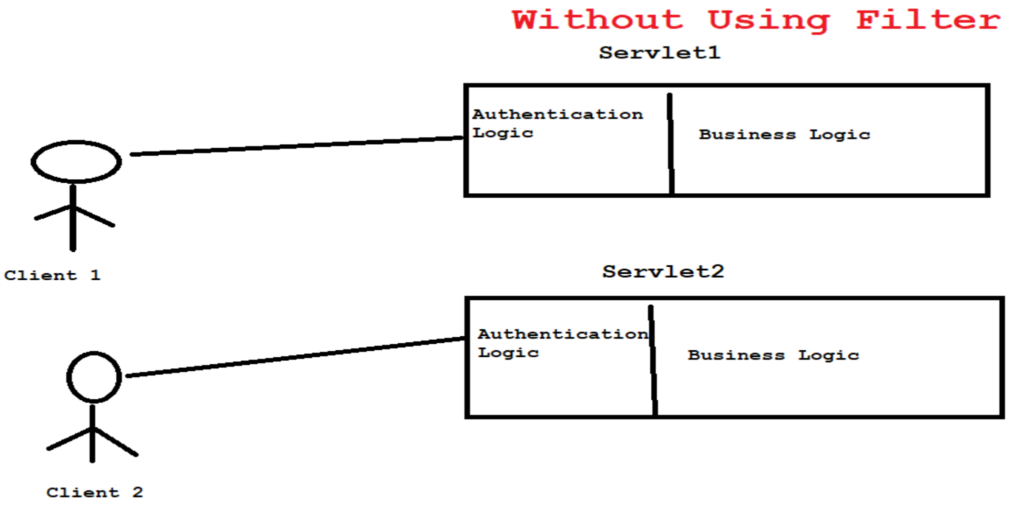
Status Code	Description
200 (OK)	Indicates that request was successfully completed and response also generated.
202 (ACCEPTED)	Indicates that request has been accepted for processing and processing is not completed
204(No Content)	Request Processed successfully but not sending any content.
400 (Bad Request)	Indicates that the server cannot process the request due to invalid request syntax.
401 (Unauthorized)	Credentials of users are not valid,Authentication failed.
404(Not Found)	Request resource is not available.
405(Method not allowed)	The request was sent with a method not supported by the servlet.
500(Internal Server Error)	Generic Error Message An unexpected condition occurs in the server.
505(HttpVersonNotSupported)	The server does not support the http version.

Difference Between Http GET and Http POST Request types?

GET	**POST**
1. Get method transforms approximately 1 KB of data at a time to server.if limit crossed then remaining data is lost.	1.Post method can transfer any amount of data at a time.there is no limit.
2. get method does not support uploading a file	2.post method support uploading file
3. Get method shows form data in address bar , so it is not a secure.	3. post method does not show a form data in address path.so its a secured method.
4. Get method is an idempotent method,it means it is designed to only read the data from the server.	4.Post() is a non-idempotent method.this means it is designed to send data to the server.

Filters:

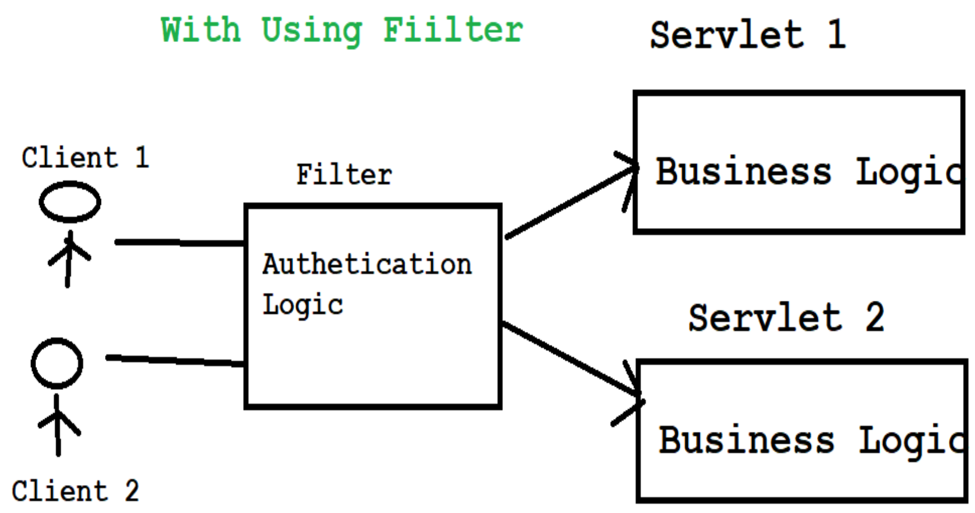
- A web application can have multiple servlets and multiple services can have common services
- If Multiple Servlets have common services and that common service is defined in multiple servlets then it's a duplicate logic.
- In order to avoid duplication of logic and also to reuse a service for multiple servlets, filters are introduced in servlets 2.3 version.
- We can define a common services in a filter ,we can add that filter to multiple servlets.
- When client send a request to a servlets and servlet container calls a filter then next transfer request to servlets.



A filter is executed in two cases.

1. when a request is going to servlet (Pre processing logic)
2. when the response is going back to servlet (Post processing logic).

- In a filter we can define pre-processing and post processing logic.
- Pre-processing logic of the filter is executed when a request is going towards servlets.
- Post processing logic is executed while the response is going towards the browser.
- In Web applications we can define multiple filters for multiple services(functionalities).
- For example,if we want to add authentication logic and timer logic for one more servlets then we can define two filters one is for Authentication and other is for timer.
- While response is coming back, first post logic of filter is executed and then post loi if filter is executed.



- A filter can be created by implementing java.servlet.Filter.
- In Filter Interface there are three abstract methods which are called life cycle methods of filter.
- When we are creating a filter class we must override all three abstract methods of filter interface.

Ex:

```
  class AuthenticatioFilter implements FIlter
   {
      publc void init(FilterConfig config) throws ServlerException
       {
          // Initialization logi if any
      }
       public void doFilter(ServletRequest req, ServletResponse res, FilterChain fc) throws
        {                                                                                  SE,IOE
              //   fc.doFIlter(req,res);
        }

       public void destroy(){
          //destory logic if any
       }
 }
```

- Like a servlet, lifecycle methods of a filter are also called by container.
- init() and destroy() methods are called once and the doFIlter() method is called for each request.
- In the doFilter() method we can define both pre and post processing logic of the filter.
- In the middle of pre and post processing logics we called doFIlter() of FilterChain interface.
- doFilter() method of FilterChain interface accepts only request and response objects and it calls either the next filter or servlet.
- doFilter() method of FIlter interface is used to define pre and post processing logics .
- doFilter() method of FilterChain interface is used to call next FIlter or Servlet.

Filter Configuration in web.xml:
—---------------------------------------

Like servlet, a FIlter is also a container managed object,so we need to configure a filter in web.xml
A Filter Configuration is almost similar as servlet configuration.
There are two parent tag are <filter> and <filter-mapping>
To add a filter to a servlet, we need to write a filter url pattern same as servlet url pattern.
If we want to add same filter to multiple servlets then we need to add multiple url patterns to the filter.

```
  <filter>
        <filter-name>fone</filter-name>
        <filter-class>pack1.MyFilter</filer-class>
  </filter>
 <filter-mapping>
       <filter-name>fone</filter-name>
       <url-mapping>/srv1</url-mapping>
<url-mapping>/srv1</url-mapping>
```

```
   <url-mapping>/srv1</url-mapping>
</filter-mapping>
```

- A Servlet Container creates an object of filter at deployment only.
- In filter configuration there is no need to write <load-on-startup> tag,suppose if we write <load-on-startup> tag with negative or positive integer then it will not be considered by container.
- Servlet Container executes filters according to the configuration order.

Servlet Thread Safety:

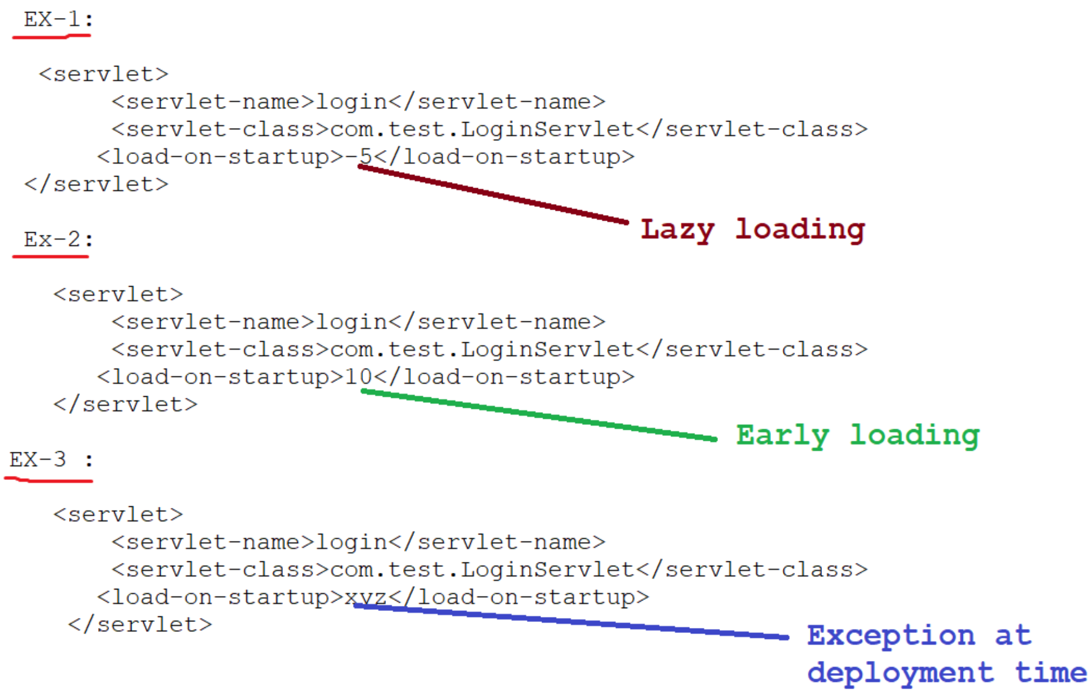
- In java, if an object is shared by multiple threads simultaneously and if the changes made by one thread affect the remaining threads then that object is not a thread safe object.
- If the changes made by one thread are not affected by another thread then that object is a ThreadSafe object.
- In the case of servlet,the servlet object is thread safe or not by default based on whether it has an instance variable or static variable or not.
- If a servlet has an instance variable or static variable then that servlet object is not a thread safe object.
- If a servlet does not have any instance or static variable then that servlet is thread safe.
- In servlet class,if we avoid instance or static variables if we create only local variables in service() method then that servlet object is thread safe.
- If we want use instance variables also in servlet class if we want to make a servlet object as a Thread safe object the we can make it three ways
    1. By implementing a servlet class from javax.servlet.SingleThreadModel interface.
    2. Define service() method in servlet as synchronized .
    3. Define synchronized block in service method.
- SingleThreadModel is a marker interface,it does not have any abstract methods,so we do not need to override any methods in servlet.
- SingleThreadModel interface tells ServletContainer that it allows one thread at a time to access a servlet object. So a servlet object becomes ThreadSafe.
- Making the service() method as synchronized is the same as implementing SingleThreadModel.
- If a class implements SingleThreadModel or if we make service() method as synchronized method in both cases a new thread has to wait until its previous thread comes out of service method.the waiting time of next thread be increased.

- So a better solution is to define a synchronized block in service method with the code that causes the ThreadSafety problem.

```
          Ex: public  class MyServlet extends GenericServlet{

                   //instance block
                   // static block
                 public void service(req,res) throws SE,IOE
                 {
                      // few lines of code
                      synchronized(this){
                        {
                             // code which required thread safety.
                        }

                    /// other code
                }

          }
}
```

Lazy loading and Early loading in a servlet:

- By default a servlet is lazy loaded because the container creates an object of the servlet class at first request time.
- Early loading of a servlet means creating a servlet object at deployment time.
- A Servlet container either lazy loads a servlet, based on the value <load-on-startup> tag configured in web.xml
- <load-on-startup> tag is a child tag of <servlet> tag.
- If we don't write <load-on-startup> tag then its default value -1
- <load-on-startup> tag value must be an integer.if it is negative then a servlet is lazy loaded if it is positive a servlet is early loaded.
- If we pass a double or string type of value to <load-on-startup> tag the exception will be thrown at deployment time by servlet container.



EX-1:

```
 <servlet>
      <servlet-name>login</servlet-name>
      <servlet-class>com.test.LoginServlet</servlet-class>
     <load-on-startup>-5</load-on-startup>
</servlet>
```

Lazy loading

Ex-2:

```
  <servlet>
      <servlet-name>login</servlet-name>
      <servlet-class>com.test.LoginServlet</servlet-class>
     <load-on-startup>10</load-on-startup>
  </servlet>
```

Early loading

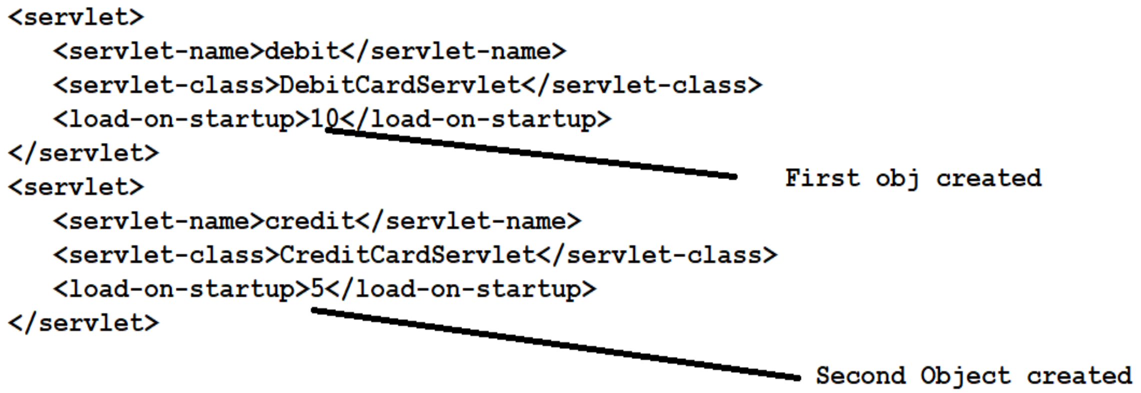
EX-3 :

```
  <servlet>
      <servlet-name>login</servlet-name>
      <servlet-class>com.test.LoginServlet</servlet-class>
     <load-on-startup>xyz</load-on-startup>
   </servlet>
```

Exception at deployment time

- In web.xml,if multiple servlets are configured with <load-on-startup> tag then a servlet container creates an object according to priority.
- Low values will get high priority and high values get low priority.

Ex:1



```
<servlet>
   <servlet-name>debit</servlet-name>
   <servlet-class>DebitCardServlet</servlet-class>
   <load-on-startup>10</load-on-startup>
</servlet>
<servlet>
   <servlet-name>credit</servlet-name>
   <servlet-class>CreditCardServlet</servlet-class>
   <load-on-startup>5</load-on-startup>
</servlet>
```

First obj created

Second Object created

Ex :2

ServletName	<load-on-startup>	Priority
Servlet1	15	3rd
Servlet2	8	2nd

Servlet3	0	1st
Servlet4	No tag provided	NA
Servlet5	-12	NA

Welcome File in Servlet Application:

- For Web applications we can set a welcome file.This file will be sent to the browser,when a request is sent from browser to web-application.
- The default welcome file of every java web application is index.html.
- If we set another html file as a welcome file by writing configuration in web.xml with <welcome-file-list> tag.
- For example if we want to set login.html as welcome file the in web.xml we need to write configuration like following

```
 <web-app>
     <welcome-file-list>
         <welcome-file>login.html</welcome-file>
     </welcome-file-list>
</web-app>
```

- We can write more than one welcome file in web.xml

```
<web-app>
     <welcome-file-list>
         <welcome-file>login.html</welcome-file>
         <welcome-file>home.html</welcome-file>
     </welcome-file-list>
 </web-app>
```

- Container will search for the first welcome file and if it is not present then the container will search for the next welcome file.

Mime Type(Multipurpose Internet Mail extension):

- A servlet can generate responses in different formats.

- A browser can open a response directly in html or plain or xml or image, for other types a browser will take support of respective software to open a response on browser.
- A servlet should intimate the browser about what type of response it is sending by setting MIME TYPE to the response.
- In order to set the MIME type, we have to call the setContentType() method of response Object.

  response.setContextTupe(“text/html”);
- If we don’t set mime type to response then default mime type will be set as **text/html**.
- Different types of MIME TYPES are

  1. text/html 2. text/plain 3.application/pdf 4. application/vnd.ms-excel
  5.text/xml 6.application/msword 7.image/gif 8.image/jpeg
  9.video/mp4 …………

FileUploading in Servlet Example:

- File uploading means transferring a file from the client system to the server.
- When a file is reached to the server then at the server side can be saved in either harddisk or database.
- To Allow the user for select a file in the form tag we need to use the below tag.

  ```
  <input type=”file” name=”file”>
  ```

  This tag will display the browse button on the browser to select a file.
- GET method of http protocol will not support file uploading from the browser,so we need to change the method of http as POST.
- When uploading a file ,a browser should not encoded its content , so we need to set browser our form encoding as “multipart/form-data”
- Finally in the form design of html page,we need to the following changes.

  ```
   <form action=”XXX”  method=”post” enctyoe=”multipart/form-data”>

       <input type=”file” name=”file”/>
  </form>
  ```

- When a file is uploaded from servlet to browser,IN Servlet api we don't have predefined methods or classes for reading the uploaded file from the request.so in a servlet class we need to take the support of a third party API. Ex: java zoom api ,Apache common file upload api ……

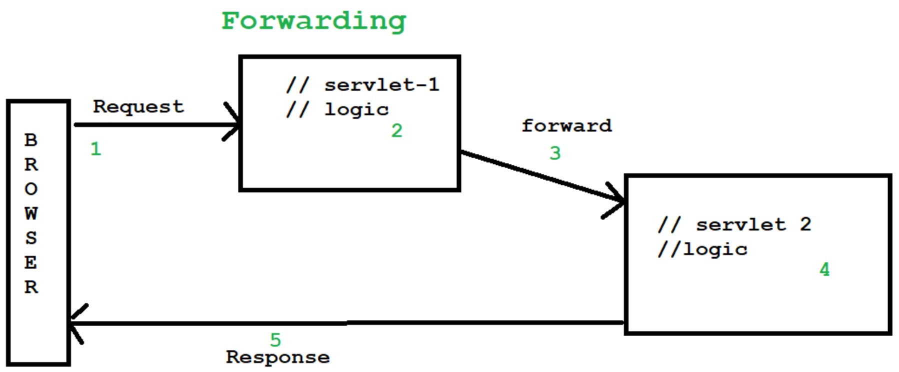
- Mostly apache common file upload api will be used for reading and uploading file request.
- In common fileupload api , a class given called ServletFileUpload ,it has parseRequest() to read upload the from the request object.
- parseRequest() reads each value of request object,creates file item object for each value and stores file itrm object in arraylist and returns that list.
- TO create file item objects servlet fileupload class takes the DiskFIleItemFactory class.
- Because of we are using third party api,we need to download commons file upload.jar and commonsio.jar and we need to add this jar file to the lib folder of our application.

Servlet Collaboration / Servlet Chaining:

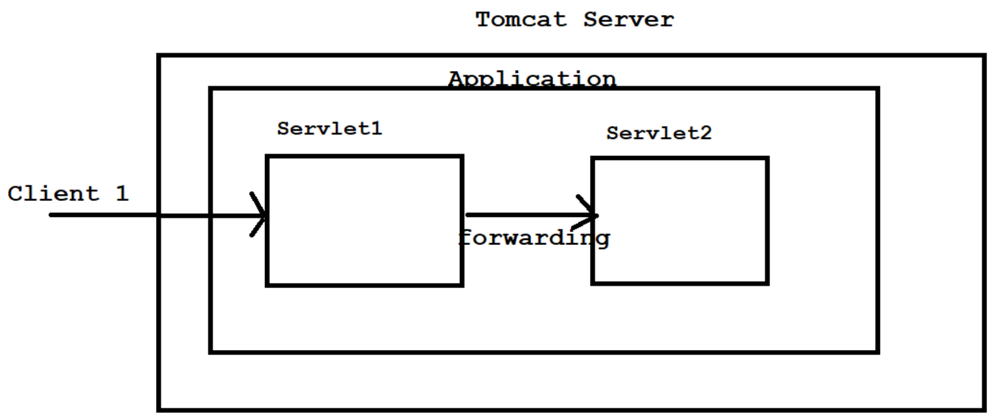
- In development of web applications,it is not possible to define complete logic to process a request in a single servlet.
- In such cases we divide the request processing logic into multiple servlets.
- When logic is developed in multiple servlets a servlet communicates with another,to provide a response to a request.
- For servlet collaboration,servlet technology is given two mechanisms.
  1. Dispatching mechanism

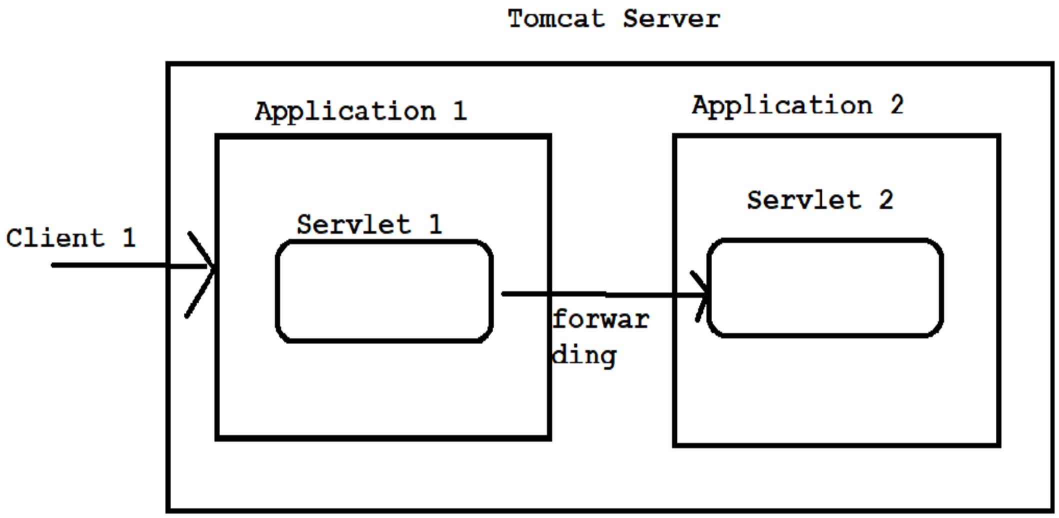
     This dispatching mechanism is divided into two types.
     a. Forwarding
     b. including
  2. Redirecting mechanism.
- **Forwarding**
  - ❖ When a request processing logic is divided into multiple servlets then a request is forwarded from one servlet to another.

- In forwarding a request processing is finished in destination servlet and response is also generated in destination servlet.
- If any response in source servlet then it will not be considered.
- When a servlet is forwarding a request in another servlet,information will not be given to the client about forwarding.



- When request forwarded then the input values submitted by client from browser are also automatically forwarded. So we can read the input values from the request object in the destination servlet also.
- To forward a request, both source and destination resources must be running in the same server only, if they are in two different servers then forwarding is not possible.

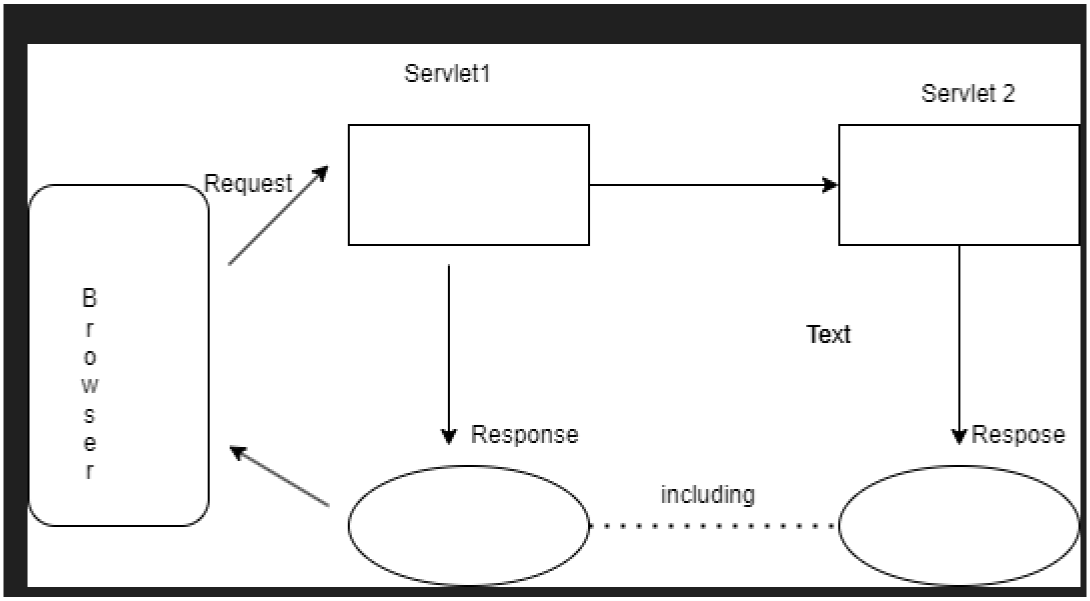




- Forwarding is possible only if the destination resource is servlet ,jsp,html.

including :

- In a web-applications, in multiple servlets response contains any common output then for re-usability multiple servlets include that out form another servlet.
- Including is also servlet collaboration because a servlet is tak=lking with another servlet yo include its output.
- Like forwarding , including also a servlet will not give any intimation to the client.
- in Including the response will be partially generated by source and partially generated by destination servlet.



- Like forwarding .even in including also both servlets must be running in the same server.
- In including also, the destination resource must be a servlet or jsp or html.

- Servlet-api has provided RequestDispatcher interface with two abstract methods, for forwarding a request and for including a response.
  forward(request,response)
  include(request,response)

We can get RequestDispatcher Object in three Approaches.

Approach-1:

We can call getRequestDispatcher() method of servletRequest interface

RequestDispatcher rd= request.getRequestDispatcher("url path");

Approach-2:

We can get the RequestDispatcher object,by calling getRequestDispatcher(), method of ServletContext interface.

RequestDispatcher rd = context.getRequestDispatcher("url");

Approach-3:

We can obtain a RequestDispatcher object by calling getNamedDispatcher() method ServletContext interface.

To the getNamedDispatcher() method, we need to pass alias name of destination servlet as a parameter.

RequestDispatcher r= context.getNamedDispatcher("alias name");

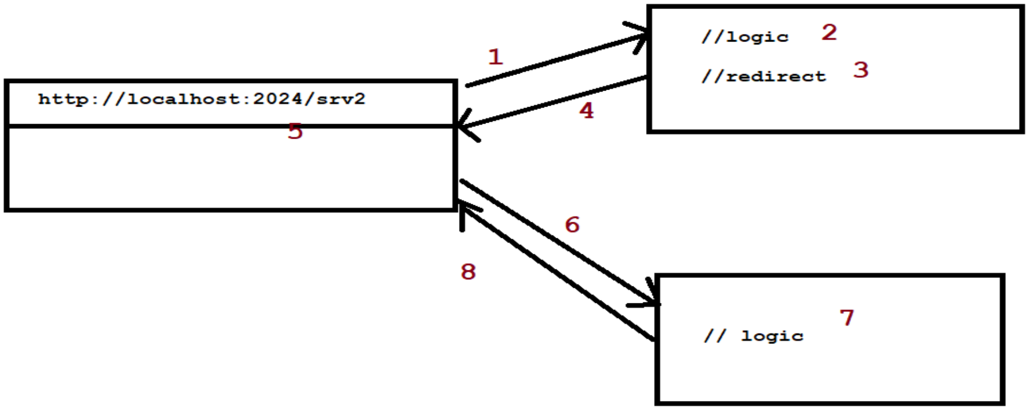
request.getRequestDispatcher() vs context.getRequestDispatcher()

request.getRequestDispatcher()	**context.getRequestDispatcher()**
Works for source and destination servlets in the same web application.	Works for source and destination servlets in the same web application.
Source and destination servlets are in two different web applications this approach will not works	Source and destination servlets are in two different web applications this approach will works.
We need to pass request relative path as input	We need to pass Context relative path As input.

Redirection Mechanism.

- When a request processing logic is divided into multiple servlets then a servlet redirect request to another servlet.
- Redirection means, telling a browser about send a request to destination servlet immediately.
- In Redirection, control comes back from servlet1 to browser,then URL, brar is changed and then another request will be sent from browser to the destination servlet.

Redirection mechanishm



- A servlet can redirect a request to another servlet by calling sendRedirect() method of HttpServletResponseInterface.
- If a Servlet extends GenericServlet then we cannot redirect request from one servlet to another servlet because service() method has parameter of type ServletRequest and ServletResponse.

Forwarding vs Re-direction:

Forwarding	Re-Direction
1. While forwarding a request a servlet will not give intimation to the server	1.Servlet will provides intimation to the browser by changing url of address bar
2. Forwarding means, some request is transfer to the the destination , so one pair of request ,response object are shared source and destination	2.redirection means, a browser will send new request to the destination, so source and destination uses different pairs of request and response object
3. In forwarding source and destination	3.In redirection,source and destination can

must be on the same server.	be within the same server or different server also.
4. in, forwarding,with request it's input values are also forwarded.	4.in direction, input values of request are not redirected.
5. In forwarding, destination resource should be either servlet,jsp,html	5.in direction, destination can be either java or non-java type of resource also.

SESSION MANAGEMENT:-

—-----------------------------------

- Http is a protocol which transfers data between a browser and server.
- If a user has send a request then the http protocol transfers the data to the server, but it does not remember the data.
- If the same user from the same browser has sent another request then the http protocol only transfers the data of the current request, but it cannot transfer previous data of that user because of that reason http protocol is called a stateless protocol.
- In web application user has send a request then that request data stored in request object.a request object is deleted from server after sending thet response.
- If the same user from the same browser has sent another request to the server then the previous data of the client does not exists. It means by default web applications are also stateless.
- When developing some web application,it should remember a client data from some period of time.
- For developing such web application session management mechanism is used.
- For example in online shopping cant applications, items selected by a user are remembered by the server until a user completes shopping,so online shopping application uses session management mechanism.

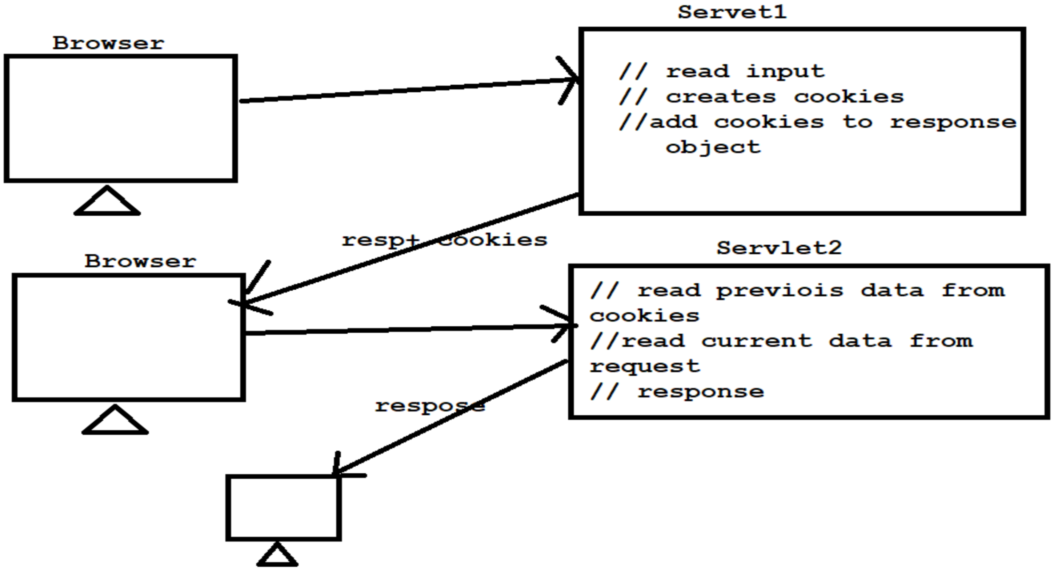
- ❖ Servlet technology has provided four ways to add session management to the web applications.
  1. Session management with cookies
  2. Session management with hidden field.
  3. Session management with http session and cookies
  4. Session management with http session and url rewriting.

With cookies:-

—---------------

- ❖ It is a session mangement techniques, to get a ststeful behaviour in web applications.
- ❖ This cookies technique works like the following.
  1. A client, browser sends a request to a servlet with data.
  2. A servlets stores the data in one or more cookies
  3. A servlet adds cookies to response object.so with response cookies are transfer to browser.
  4. When user submits one more request then with that request,browser sends cookies to servlet
  5. A servlet reads previous data of client then cookies and current data of client from request.
  6. Finally the servlet generates response based on both previous data and current data of the client.

Cookies flow


Browser
Servet1
// read input
// creates cookies
//add cookies to response object
resp+ cookies
Browser
Servlet2
// read previois data from cookies
//read current data from request
// response
respose

- A cookie is an object which stores data in a key- value pair format, creates a server and sends to the browser and stores on the browser.
- For creating cookies servlet technology provided “Javax.servlet.http.cookie” class.

Creating cookie:-

—--------------------

- Cookie c1= new cookie(string key, string value);
- Every cookie must contain key and value and both must be string format.
- We cannot create an empty cookie and we cannot create a cookie with a key.

  Ex:

  ```
  String name = req.getparameter(“uname”);
  Cookie c1 = new Cookie (“un”, uname);
  Cookie c2 = new cookie();
  Cookie c3= new Cookie (“un”);
  ```

- A cookie can be added to the response object by calling add cookie() method of http servlet response interface.

  ```
  response.addcookie(c1);
  ```

- To read cookies that are sent along with the request, we need to call getcookie( ) method of httpservletrequest interface. This method returns an array containing one or more cookie objects.

  ```
  Cookie cookies( )= request.getcookies( );
  ```

Session management:-

Apply stateful behaviour in a web application is called session management.

Types of cookies:

1. Non- persistent cookie(in memory cookie)
2. Persistent cookie.

- A non- persistent cookie will be stored temporarily in ram by a web browser. When the browser is closed then the non persistent cookies are also removed from ram.
- If we open the browser again and if we send the result then cookies are not transferred to the server.
- Persistent Cookies are stored in the form of deleted hard-disk by browser. They are deleted when a browser is closed.
- A browser deletes cookies from hard-disk when a browser is closed.
- A browser deletes cookies from hard- disk when they expire time is reduced.
- By default a cookie in a servlet is a non-persistent cookie. We can make cookie is persistent cookie by setting “expire time for cookie”.

```
 Cookie c1= new Cookie(key,value);
  c1.setMaxAge(30);
```

- When cookie object is created by default its expire time will be set as -1 seconds.so it is an in-memory cookie.
- setMaxAge() method will set expire time for a cookie object ,so its is Persistent cookie.

Methods inside Cookie class:

1.setMaxAge(seconds) : it will set the expiration time for a cookie.

```
c1.setMaxAge(300); -> 5 mins expiry time
```

2.**getMaxAge()** : it will read the expiry time of the cookie.

```
Int seconds= c1.getMaxAge();
```

3.**setValue()**:

It will change the value stored in the cookie.

```
c1.setValue(“java”);
```

Value of a cookie can be changed but the key cannot be changed.

4.**getName**():

Read the key and value of a cookie.

```
String key= c1.getName();
```

5.**getValue**():

```
String value=c1.getValue();
```

Drawbacks of Cookies:

1. If a browser disables cookies then session management does not work.
2. If a large number of cookies are transferred between browser to server then network traffic will be increased.
3. we can set only a string type of data to a cookie ,cannot set other types of data.
4. A Browser can store approximately 300 cookies on a website,more than that cannot be stored.

Session Management With Hidden Fields:

1. This session management technique is almost similar to cookies.
2. In this technique input values of a user are stored in hidden fields,then they are added to the form.
3. Along with the form hidden fields are also transferred to the browser but hidden fields are invisible on the browser.
4. When a user submits a next request from that browser then values of hidden fields are also sent to the server.
5. One Advantages of hidden fields over cookies is, in the case of cookies a browser can block them but hidden fields a browser cannot block the hidden fields.

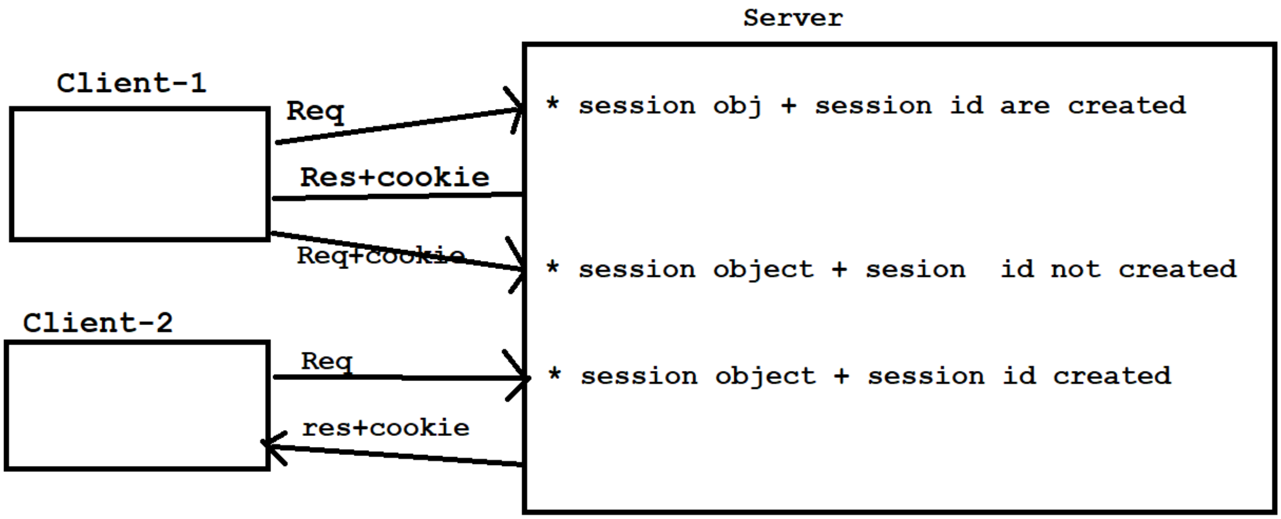
Drawbacks of Hidden fields:

1. hidden fields can also store string type of data, we cannot store any other data type.
2. Transferring a huge number of hidden fields over the network increases network traffic.

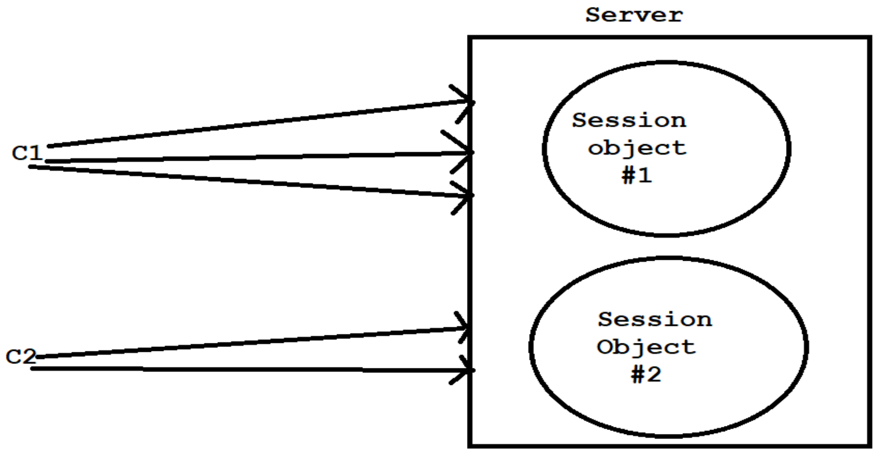
HttpSession with Cookie:

- HttpSession is an interface and its implementation will provided by vendor.
- In this session tracking technique for each new user a new session object will be created in server and a session id will be set to the session object.

- In session object,there is internally a map object to store a data of client which require for future object.
- A session id is a very large string,its used to identify the session object.
- Creation of a session object and session id will be done by servlet container.
- This session tracking technique works like the following.
    1. When a new user send request to servlet then a session object and sessioni-d are created.
    2. Container transfers session id in the form of a cookie along with response to the browser.
    3. When the user sends the next request then with that request cookie also transferred to the server.
    4. With the session id stored in the cookie server identifies that client as an existing client,so the server will not create a session object again.


Server
Client-1
Req
Res+cookie
Req+cookie
Client-2
Req
res+cookie
* session obj + session id are created
* session object + sesion id not created
* session object + session id created

- In the server,session objects are created for each user but not for each request.
- The no.of session objects in the server is always equals to the noof clients connected to the server.



How to get a session Object?

- ➢ A session object and session id are automatically created by Servlet container.
- ➢ If a httpServletRequest interface, a method getSession() is provided which internally calls a container and finally returns a session object.
- ➢ We can call getSession() method without parameter and boolean parameter.

HttpSession session=request.getSession();

a. getSession() method internally calls getSession(true). It returns either a new session object or existing session object.

b. A container internally check weather a client is new or existing based on session id and then create either new session object or returns existing session object.

HttpSession session=request.getSession(true);

getSession() and getSession(true) works like same, get session() internally calls the getSession(true);

HttpSession session=request.getSession(false);

a. getSession(false) returns either existing session object or null.

b. If a client is a new client then getSession(false) returns null.

c. If a client is an existing client then getSession(false) returns a session object.

Storing Data to Session:

In servlet technology, data can be shared by storing in request or session or context objects.

1. The data stored in the request object is removed when the response of the request object is sent to the browser.

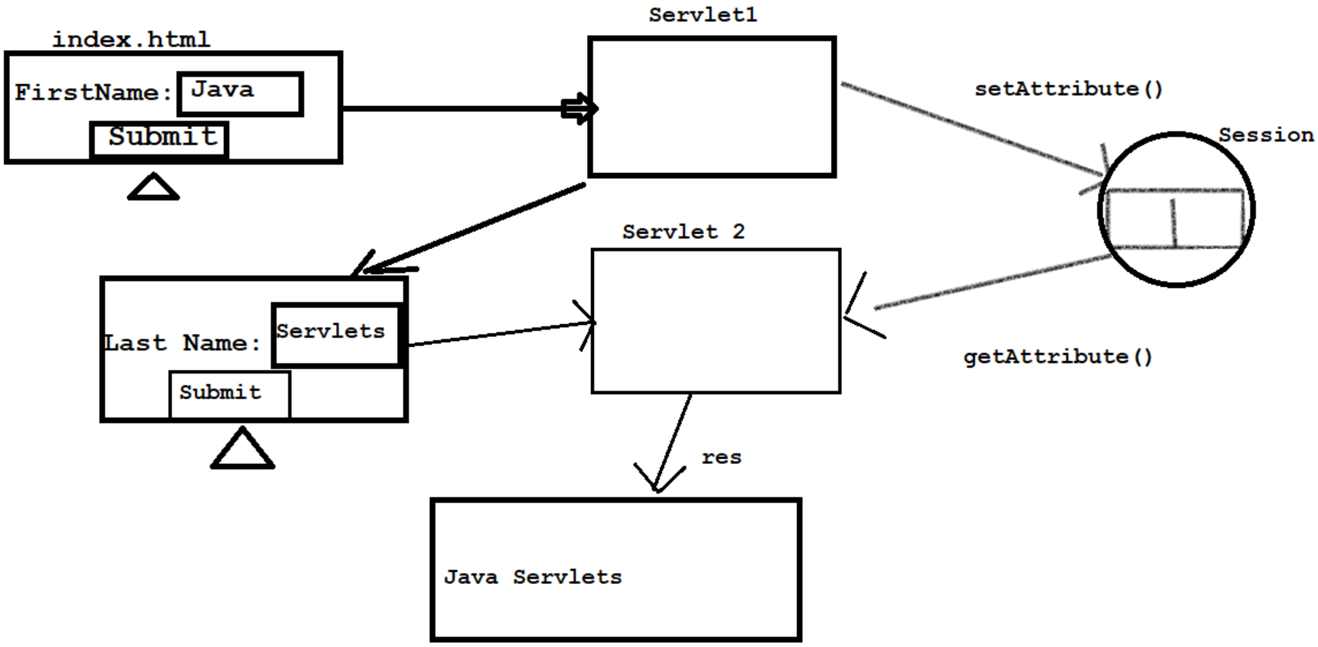
2. The data stored in session object can be shared in multiple request of the same client, data is removed when a session is closed.

3. The data stored in context object is shared for all the client throughout the life time of application,the data removed when context object is closed.

The data can be stored or read or remove by calling following methods.

i. setAttribute(key,value)
ii. getAttribute(key)
iii. removeAttribute(key)

Example:



HttpSession with URL-REWRITING:

- In the session with cookie technique, a session is is transfer from server to browser in the form of cookie but the browser may reject cookie.

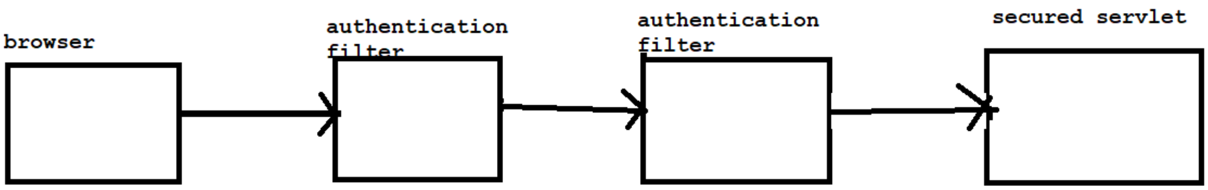
- If the cookies are rejected by the browser session management does not work.
- To overcome the above problem url rewriting technique is provided.
- In this url rewriting technique session is appended to the url pattern, that is sent from server to browser..
- When another request is submitted from the browser then with url pattern the appended session is also transformed to the server.
- This url rewriting technique works in both cases when a cookie is accepted by the browser and cookie rejected by the browser.
- For appending session to url pattern, we need call encode url( ) method of http servlet response interface.
- **Example:**
  response.encodeurl("srv2");
  The above statement converts url pattern as srv2;sessionid=4256fdr2hdgjj

WebApplication Security :

- A web application is a group of resources and it may contain some static resources and some dynamic resources.
- A web application resources can be either secure or not secure.
- A non secure resource only accessible to all the clients ,it is public.
- A secure resource is only accessible to authorized clients. they are called protected resources.
- Providing security to a web resources is nothing but we authenticating and authorizing a client before accessing a particular resources.
- Authentication is nothing but verifying user name and password and authorization nothing but checking the access permission of authenticated clients.
- If username and password are valid then it is not possible to access every resource access permission also required.
- For example if we show a flight ticket and id proof then we will get a boarding pass it means a passenger is authenticated.
- With that boarding pass, we don't have any permission to sit in any flight,we are authorized to sit only in a particular flight.
- We web applications, to provide security for a servlet or jsp, we have two ways

1. Programmatic security
2. Declarative security.

- In programmatic security, we can define two filters manually for authentication and authorisation.
- When a request is sent to a secure servlet the authentication filter verifies username and password them authentication filter checks for permission.if authorized success then that request is allowed and send to servlet.



-
- Every authentication user may not be an authorized user. Every authorized user is an authenticated user.
- A problem in programmatic security is a program has to manually define logic in filters and a program has to configure filters manually in web.xml.
- In declarative security a programmer needs to configure security tags in web.xml automatically and the servlet contains will authenticate and authorize a user so declarative security is better than programmatic security.
- In web.xml, we need to configure security tags in the following order to get declarative security.
- To provide declarative security to web resources, we need to select one of the four authentication models.
  1. Basic
  2. Digest
  3. Form based
  4. Client certificate(ssl).

Basic authentication model:-

- This web application authentication model works like the following.
  1. A client(end user) will send request to a secure servlet from browser
  2. A servlet container will tell the browser that authentication is required, basically the authentication model.

3. A browser opens a dialog box to enter the username and password. Browser will send the username and password to the container.
4. Container verifies username and password if not valid then again 2,3,4 steps are repeated.
5. If valid then the client is allowed to access that servlet.
6. A best example for a basic authentication model is tomcat manager.

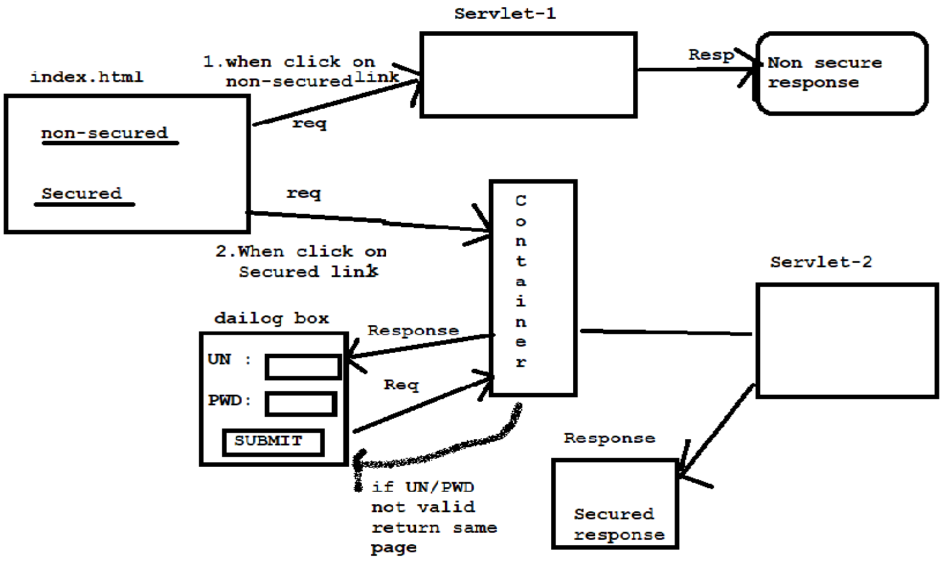
- We click an manager app button in tomcat home page then a request is sent to the manager servlet but it is a secured servlet.
- Tomcat container tells the browser that authentication is required, so the browser opens a dialog box to enter the username and password.
- We need to configure the role name with one or more usernames and passwords in tomcat/users.xml of tomcat/conf directory.

```
<user username="chenchu" password="star" roles="employee"/>
<user username="raja" password="super" roles="admin"/>
<role rolename="manager-gui"/>
<user username="tomcat" password="tomcat" roles="manager-gui"/>
```

web.xml:

Add below lines.

```
   <security-constraint>
        <web-resource-collection>
            <web-resource-name>abc</web-resource-name>
            <url-pattern>/srv2</url-pattern>
            <http-method>GET</http-method>
        </web-resource-collection>
        <auth-constraint>
            <role-name>employee</role-name>
        </auth-constraint>
    </security-constraint>
    <login-config>
        <auth-method>BASIC</auth-method>
    </login-config>
```



DIGEST Authentication:

Just change the value of <auth-method> element inside the <login-config> element from BASIC to DIGEST. Everything else is the same for both BASIC and DIGEST authentication.

Form Based Authentication:

- In this authentication model, servlet containers will send a login page to the browser, when a user sends a request to the secured resource.
- If the username and password is not valid then the container will send an error page to the browser.
- The login page and error page should be defined by the application developer and the pages must be configured in web.xml like the following.

```
<security-constraint>
        <web-resource-collection>
                <web-resource-name>abc</web-resource-name>
                <url-pattern>/srv2</url-pattern>
                <http-method>GET</http-method>
        </web-resource-collection>
        <auth-constraint>
                <role-name>employee</role-name>
        </auth-constraint>
 </security-constraint>
 <login-config>
        <auth-method>FORM</auth-method>
        <form-login-config>
                <form-login-page>/login.html</form-login-page>
                <form-error-page>/login_failed.html</form-error-page>
        </form-login-config>
</login-config>
```

- In the form based authentication, when a login page is submitted then a predefined servlet will check the username and password, so while designing the form, form action and username and password text box names are pre-defined.
    1. Form action must be j_security_check
    2. Name of username of text of text box must be j_username
    3. Name of the password textbox must be j_password.

How to Configure Database Authentication for Basic/Digest/Form Based Authentication ?

- For authenticating a user, tomcat container reads usernames,passwords and roles from tomcat- user.xml. It is called memory authentication.
- A problem with in-memory authentication is that if a new user is added or an existing user is deleted then everytime we need to restart the server. Because, the container can read tomcat-users.xml of startup-time only.
- We can solve the above problem, by using the database, for maintaining username, passwords and the roles, this is called database authentication.
- In database authentication, a new user can be added and an existing user can be removed without restarting the server.
- To add database authentication to a web-application security do the following changes.
    1. Create the following tables and insert sample records in Mysql.

```
create table users( user_name varchar(15) not null, user_pass      varchar(15) not null);
insert into users values('chechu','java');
insert into users values('raja','jdbc');
insert into users values('tomcat','tomcat');
create table user_roles( user_name varchar(15) not null, role_name varchar(15) not null,
primary key (user_name,role_name));
insert into user_roles values('chenchu','employee');
insert into user_roles values('raja','admin');
```

DIGEST Authentication:

Just change the value of <auth-method> element inside the <login-config> element from BASIC to DIGEST. Everything else is the same for both BASIC and DIGEST authentication.insert into user_roles values('tomcat','manager-gui');

2. Open server.xml of tomcat and add the following <realm> tag.

```
 <Realm className="org.apache.catalina.realm.JDBCRealm"
                    driverName="com.mysql.jdbc.Driver"
                    connectionURL="jdbc:mysql://localhost:3306/startech"
                   connectionName="root"   connectionPassword="root"
                   userTable="users" userNameCol="user_name"
                   userCredCol="user_pass"  userRoleTable="user_roles"
roleNameCol="role_name"/>
```

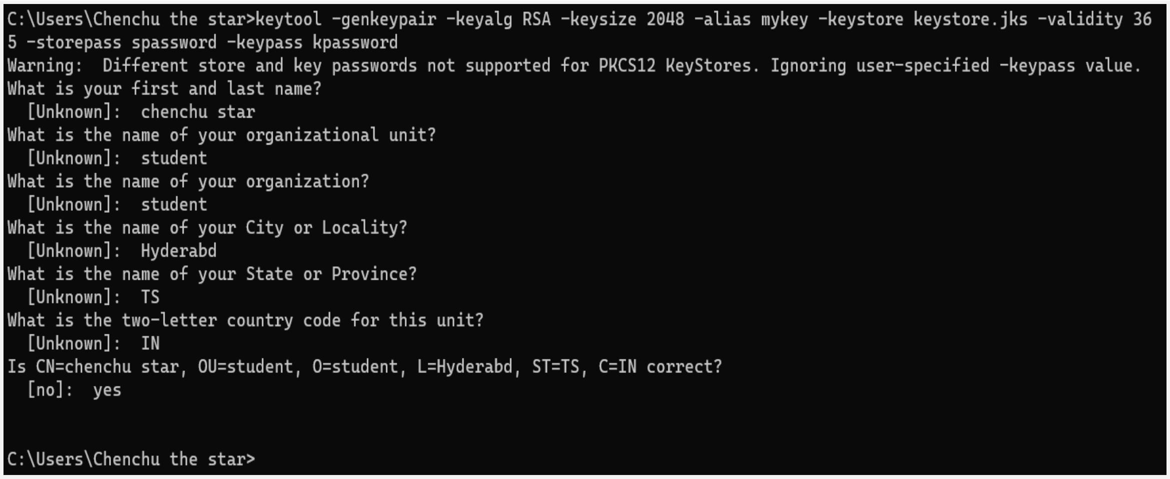
3. Copy MYSQL driver file to the lib folder of tomcat.
4. Now Start server and test BASIC/DIGEST/FORM BASED applications.

SSL Authentication/ Client Certificate Authentication:

- Ssl authentication is not for verifying the username and password and role. It is for strongly encrypting data and transferring between browser and server.
- Ssl authentication use RSA algorithm for encrypting and decrypting the data.
- In ssl authentication, first a certificate will be send to the browser by server.
- A browser verifies whether the server to whom it wants to connect is the same or not, after that a browser accepts a server certificate.
- If certificate is accepted by the browser the only browser and server can communicate.
- If a certificate is accepted, the server will send encrypted data to the browser and at the browser side, it is decrypted with the help of keys available in the certificate.
- A browser will send encrypted data to the server and at the server side it will be decrypted.
- The data transfer between browser and server its https protocol, so it is not possible to hack the data in the middle.
- If we want authentication and authorization of the user along with ssl https then we can use ssl with basic ssl formbased.
- For real time applications, a trusted certificate authority generates a security certificate for an organization.
- Some trusted third party certificate authorities are verisign,B2B,thawte…etc.
- For java applications we can generate a security certificate(untrusted), by using the key tool command of jdk.
- In a web application,if web want to enable ssl authentication, the following changes are required.

1. We need to generate a certificate with keytool like the following command.

```
c:\>keytool -genkeypair -keyalg RSA -keysize 2048 -alias mykey -keystore
keystore.jks -validity 365 -storepass spassword -keypass kpassword
```



```
C:\Users\Chenchu the star>keytool -genkeypair -keyalg RSA -keysize 2048 -alias mykey -keystore keystore.jks -validity 36
5 -storepass spassword -keypass kpassword
Warning:  Different store and key passwords not supported for PKCS12 KeyStores. Ignoring user-specified -keypass value.
What is your first and last name?
  [Unknown]:  chenchu star
What is the name of your organizational unit?
  [Unknown]:  student
What is the name of your organization?
  [Unknown]:  student
What is the name of your City or Locality?
  [Unknown]:  Hyderabd
What is the name of your State or Province?
  [Unknown]:  TS
What is the two-letter country code for this unit?
  [Unknown]:  IN
Is CN=chenchu star, OU=student, O=student, L=Hyderabd, ST=TS, C=IN correct?
  [no]:  yes


C:\Users\Chenchu the star>
```

2. Open server.xml in tomcat conf directory and remove the comments for <connector tag with port 8443 and add the connector tag like below.

```
<Connector port="8443"
protocol="org.apache.coyote.http11.Http11NioProtocol"
           maxThreads="150" SSLEnabled="true">
    <SSLHostConfig>
        <Certificate certificateKeystoreFile="D:\securitykey\keystore.jks"
                     certificateKeystorePassword="spassword"
                     type="RSA" />
    </SSLHostConfig>
</Connector>
```

3. Open web.xml of any application and the following tag after closing authorization constraint.

```
   <user-data-constraint>
         <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
```

4. Start the tomcat server and then send the request to the application from browser.

5. Click on the secured link then request is redirected to https protocol,server will send a certificate to browser.
6. If the certificate is accepted then server will send the login page back to browser.

Servlet Annotations:

1. In servlet 3.0, annotations are provided, to reduce the size of web.xml
2. In servlet 3.0, a new package was added in the servlet api called javax.servlet.annotation.
3. Annotations are provided for class level configuration but not for application level configuration.
4. For example,<servlet> end <servlet mapping> tags are used to configure a servlet class.it is a class level configuration. But <context paren> tag is to configure context parameters and they are common for all servlets of applications so it is application level configuration.
5. The following are the list of annotations,
   - @Webservlet
   - @Webfilter
   - @WebListener
   - @WebInitParam
   - @ServletSecurity
   - @Httpconstrain
   - @HttpMethodConstrain

6.@web servlet annotation removes <servlet> and <servlet mapping> toss from web.xml'

7.@web init param removes <init param> toss from web.xml.

EX:-

```
@webservlet (value="/srvl", name="sone",loadonstartup=1,
initparams={@webinitparam(name="p1",value="100"),@webinitparam(name="p2",value="2
00")}
Public class myservlet extends servlet
}
—----
—---
```

}

- ❖ @Webfilter annotation removes <filter>and <filter mapping> tags from web.xml
- ❖ Ex:
    - ❖ @webfilter(value=”/srvl”, name=”filter”)
      Public class myservlet extends httpservlet
      }
      —-------
      —-------
      }

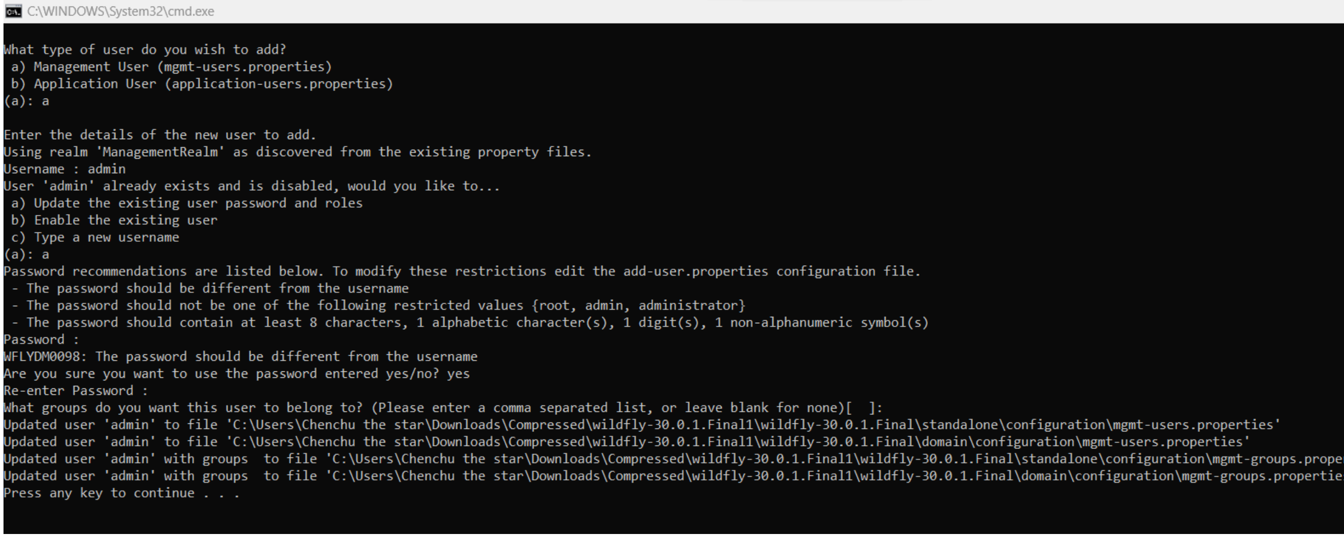
If we want to pass multiple url patterns to filter then in the place of value we should use urlPattern element.

```
 @webFilter(urlPattern={“/srv1”,”srv2} name=”filter1)
  Public class MyFilter implements Filter{

   }
```

How to Setup WildFly Server in WIndows?

1. Download the latest wildfly server from below link
   https://github.com/wildfly/wildfly/releases/download/30.0.0.Final/wildfly-30.0.0.Final.zip

2. Extract the zip file.
3. By default wildfly server port is 8080, if we want to change , open the file below and change the port 8080 to a new port number.
   wildfly-30.0.1.Final\standalone\configuration\standalone.xml
   <socket-binding name="http" port="${jboss.http.port:8080}"/>
4. Wildfly by default comes with an admin user but it's disabled, to enable admin user run **wildfly-30.0.1.Final\bin\add-user.bat** with administrator mode and follow the instructions.



```
What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): a

Enter the details of the new user to add.
Using realm 'ManagementRealm' as discovered from the existing property files.
Username : admin
User 'admin' already exists and is disabled, would you like to...
 a) Update the existing user password and roles
 b) Enable the existing user
 c) Type a new username
(a): a
Password recommendations are listed below. To modify these restrictions edit the add-user.properties configuration file.
 - The password should be different from the username
 - The password should not be one of the following restricted values {root, admin, administrator}
 - The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
Password :
WFLYDM0098: The password should be different from the username
Are you sure you want to use the password entered yes/no? yes
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[  ]:
Updated user 'admin' to file 'C:\Users\Chenchu the star\Downloads\Compressed\wildfly-30.0.1.Final1\wildfly-30.0.1.Final\standalone\configuration\mgmt-users.properties'
Updated user 'admin' to file 'C:\Users\Chenchu the star\Downloads\Compressed\wildfly-30.0.1.Final1\wildfly-30.0.1.Final\domain\configuration\mgmt-users.properties'
Updated user 'admin' with groups  to file 'C:\Users\Chenchu the star\Downloads\Compressed\wildfly-30.0.1.Final1\wildfly-30.0.1.Final\standalone\configuration\mgmt-groups.prope
Updated user 'admin' with groups  to file 'C:\Users\Chenchu the star\Downloads\Compressed\wildfly-30.0.1.Final1\wildfly-30.0.1.Final\domain\configuration\mgmt-groups.propertie
Press any key to continue . . .
```

Note : remember the password what you given here.

5. Start wildfly server in standalone mode by below file.
   wildfly-30.0.1.Final\bin\standalone.bat

6. open below the management console url.
   http://127.0.0.1:9990/console/index.html
   Enter credentials

7. Click on deployments then add war file to deploy.