1. Servlet(I) ⇦ javax.servlet
   init(),service(sreq,sresp), destroy()===> life cycle methods
   getServletInfo(), getServletConfig()

   ⇧ implements

2. GenericServlet(AC) ⇦ javax.servlet
   All 4 methods of Servlet(I) implementation is available
   abstract public void service(Sreq,Sresp) throws SE,IOE

   ⇧ extends

3. HttpServlet(AC) ⇦ javax.servlet.http

GenericServlet====> meant for any protocol interaction
                    like http, smtp,.....

For a webapplication, the protocol used in "Http",so to work with
specific protocol SUNMS had come up with SRS called "HttpServlet".

SUNMS had come up with special package to promote
        http protocol interaction
        interface
        =========
        1.HttpServletRequest(I)
        2.HttpServletResponse(I)
        3.HttpSession(I)

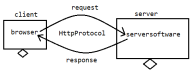
        class
        =====
        1.HttpServlet
        2.Cookie

public void init(ServletConfig config) throws ServletException;
        ⇧
This is used internally by webcontainer

1. loading
2. instantiation
3. initialization
        init(ServletConfig config)
        {
            init()   ⇧ local variable
        }
4. service
    public void service(Sreq,Sresp)throws SE,IOE
    {
        ServletConfig config =getServletConfig();
        System.out.println(config); // null
    }
5. deInstantiation

public void init() throws ServletException;
        ⇧
This is used by devleoper in his code

1. loading 2. instantiation
3. initialization
        init(ServletConfig config)
        init()
4. service
5. destroy



Structurue of HttpProtocol Request
==================================

RequestLine
RequestHeaders
RequestBody

RequestLine: Request Type | Requested resource | protocol version
GET | FirstApp/test | HTTP/1.1
FirstApp/test ⇧ configured in the server env

RequestHeaders ⇦ internally used to give a proper response
        |
Browser details and its supported language format and so on.....

RequestBody ⇦ "user supplied inputs will be available"
        |
It contains user information

Structure of HttpProtocol Response
==================================

StatusLine
ResponseHeaders
ResponseBody

StatusLine: Protocol version | status code | Description status code
HTTP/1.1 | 200 | OK
information

1XX => informational
2XX => Succesfull
3XX => Redirection
4XX => ClientError
5XX => ServerError

ResponseHeader ⇦ MIME(MultiPurpose Internet Mail Exchange)
        |
response information from the server
    a. content-type
    b. content-length
    c. modified date etc.....

ResponseBody ⇦ "Acutal information"
        |
Actual information sent by the server,this
information will be displayed to the user.

Type of HttpRequest Methods
===========================

1. GET
2. POST
3. HEAD
        ⇘ introduced in HTTP/1.0V
4. PUT
5. DELETE
6. OPTIONS
7. TRACE
        ⇗ HTTP/1.1V

RequestType used in Webapplications are
    a. GET
    b. POST

GET (it is also called as "idempotent request/safe request")
  => It is used to get the information from server
  => This request data will be visible in the address bar when we send the request,
     so it is not secured.
  => It can be book marked,and it also supports caching of data
  => since the data is visible in address bar, only limitied data can be sent.

eg:: http://localhost:9999/FirstApp/test ?username = sachin & password = tendulkar
                                        ⇧ QueryString

  1. Type url and hit enter key
  2. clicking on hyperlink
  3. submitting html form with method attribute as "GET"

RequestLine ⇦ GET(Query String is sent)
RequestHeaders ⇦ browser details would be available
RequestBody ⇦ empty

POST(It is not a safe request/idempotent request)
  => It is used for putting the data sent from client to server
  => This request data would not be visible in the address bar when we send the request,
     so it is secured.
  => It can't be book marked,and it won't supports caching of data
  => since the data is not visible in address bar, large volumen of data can be sent.

eg: http://localhost:9999/FirstApp/test ? username = sachin & password = tendulkar
                                        ⇧ QueryString

1. submitting html form with method attribute as "POST"

RequestLine ⇦ POST
RequestHeaders ⇦ browser details would be available
RequestBody ⇦ (Query String is sent)

HttpServlet(AC)

```
public void service(SReq req, SRes resp) throws SE,IOE
{
        HttpServletRequest request = (HSR) req; HttpServletResponse response = (HSR) resp;
        service(request,response);

}
protected void service(HSR req,HSR resp) throws SE,IOE
{
    String method = req.getMethod();

    if(method.equalsIgnoreCase("POST"))
      doPost(req,resp)

    else if(method.equalsIgnoreCase("GET"))
      doGet(req,resp);

    ;;;;;
    ;;;;;
    ;;;;;
    else
        return 501 status code saying http method not implemented

}
```

HttpServlet(AC)

```
protected void doPost(HSR req, HSR resp) throws SE,IOE
{
    return 405|400 status code saying HTTP Method POST not s upported by this URL
}

protected void  doGet(HSR req, HSR resp) throws SE,IOE
{
    return 405|400 status code saying HTTP Method GET not s upported by this URL
}
```

1. loading
      static block

2. instantiation
      constructor(public zero argument)

3. initialization
      init(ServletConfig config)
      init() ⇦ meant for devleoper

4. request processing
      public void service(SReq req,SRes) throws SE,IOE
      protected void service(HSR req,HSR res) throws SE,IOE
      ⇧
      user defined FirstServlet Object doXXX(HSR req,HSR res) throws SE,IOE should be called

5.deinstantiation
      public void destroy()

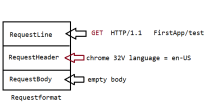
Structure of Project
====================

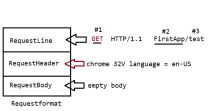
ProjectName
|
|====> WEB-INF
| |=======> classes
| |=======> lib *.class   private-area
| |=======> web.xml
|
|=====> .java
|=====> .html files   public area

⇧ The region/files which can't be accessed by the clients from address bar(sending the request)
This is solely meant for webcontainer only.

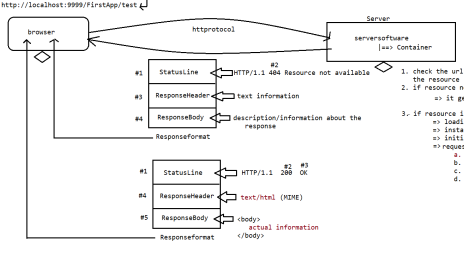
The region/files which can be accessed directly by the clients from address bar(sending the request)

http://localhost:9999/FirstApp/test
⇧ it is configured in annoatation style for container



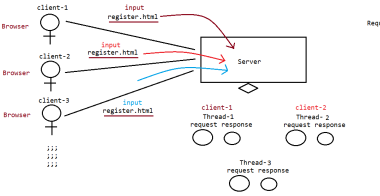


http://localhost:9999/FirstApp/test



1. check the url pattern and find the resource
2. if resource not available
        => it generates clienterror and send it server

3. if resource is available
        => loading
        => instantiation
        => initialization
        => request processing
            a. content-type
            b. use response entry
            c. write the response
            d. close the writer




```
RequestApp
    |-> src
        |=> in.ineuron.controller
                => RequestServlet.java

    |-> webapp
        |====> register.html

    |=> WEB-INF
        |=> classes
            |=> RequestServlet.class
        |=> lib
            |=> *.jar

        |=> web.xml
            |=> mapping details in xml format
```