commonly used packages in servlet coding
javax.servlet.*
javax.servlet.http.*
==========
using these interfaces abstraction is promoted through which WODA is achieved
ServletRequest(I)
ServletResponse(I)
HttpServletRequest(I)
HttpServletResponse(I)
ServletContext(I)
ServletConfig(I)

```
void init(ServletConfig config) throws SE,IOE
{

}
void doXXXX(HttpServletRequest request, HttpServletResponse response)throws SE,IOE
{
      //request => QueryString data in the form of key-value pair
      //response=> PrintWriter object will be there with empty response
}
```

w.r.t tomcat server the implementation classnames are as shown below
==========
Implementation class of config is :: org.apache.catalina.core.StandardWrapperFacade
Implementation class of context is ::
org.apache.catalina.core.ApplicationContextFacade
Implementation class of request is :: org.apache.catalina.connector.RequestFacade
Implementation class of response is :: org.apache.catalina.connector.ResponseFacade

Different types of scope and attributes in Servlet
==========
scope refers to the accessability of a variable.
      a. local scope => restricted inside method
      b. global scope=> avaialble in all the methods of a particular task.

There are 3 types of parameters(k,v) possible in servlet
      a. Form parameters(QueryString[k,v])
      b. ServletInitializationParameters(ServletConfig[k,v])
      c. ContextInitializationParamters(ServletContext[k,v])

The above 3 parameters type are read-only.from the servlet we can perform only read operation, we cannot modify remove
values based on our requirement.so we say parameter type of data is not best suited for sharing the data between
component of the webapplication.
parameter data => both key and value should be String.

To resolve this problem we should go for "attributes" type of data.based on our requirement we can add the data,
remove the data and we can share the data between the components of the application.
attribute data => key should be String, value can be any Object.

Based on our requirement we need to store the attributes in particular scope.
In Servlet we have 3 types of scope
      1. request
      2. session(HttpSessionTracking)
      3. application/context .

1.request
	=> This scope is maintained by ServletRequest /HttpServletRequest object.
	=> This scope will start at the time of request object creation(before calling service())
	=> This scope will destroy at the time of request object destruction(after calling service())
	=> The data stored in the request object will be available for all components which are procesing that request.

3. application/context
	=> This scope is maintained by ServletContext object.
	=> This scope will start at the time of context objet creation(during deployment)
	=> This scope will destroy at the time of context object destruction(during undeployment)
	=> The data stored in the context object will be available for all the components of the application,irrespetive of
		request and the user.

write a program to display hit count(number of requests) of a webapplication?
	firstreq = > hitcount = 1

	refer: ScopeApp

Write a program to display all the attribute information present in application scope?
	Note: In ApplicationScope container will add some attributes for internal purpose.

Getting information from the url
==========
	1. getRequestURI()
	2. getQueryString()
	3. getServletPath()
	4. getPathInfo()
	5. getContextPath()

```
package in.ineuron.controller;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


@WebServlet("/test/iNeuron/*")
public class TestServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

	@Override
	public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		PrintWriter out = response.getWriter();
```

```
            out.println("<h1>Request URI  :: "+request.getRequestURI()+"</h1>");
            out.println("<h1>Context Path ::"+request.getContextPath()+"</h1>");
            out.println("<h1>Servlet Path ::"+request.getServletPath()+"</h1>");
            out.println("<h1>Path Info    ::"+request.getPathInfo()+"</h1>");
            out.println("<h1>Query String ::"+request.getQueryString()+"</h1>");
            out.close();
      }
}

request
      http://localhost:9999/RequestAppInfo/test/iNeuron/hyder/java?
name=sachin&password=tendulkar
response
      Request URI :: /RequestAppInfo/test/iNeuron/hyder/java
      Context Path ::/RequestAppInfo
      Servlet Path ::/test/iNeuron
      Path Info ::/hyder/java
      Query String ::name=sachin&password=tendulkar

Deployment
==========
      Harddeployment
            Creating an webapplication inside webapps folder of tomcat and starting
the server manually is called
            "hard-deployment".

      Smoothdeployment
            Creating an application outside the webapps folder of tomcat and
starting the server through some additional
            set up is called "smooth-deployment".

Note:
webapps(deployment folder)
      In case of eclipse integration with tomcat, internally eclipse uses "smooth"
deployment through which it clones
      our project and perfoms deployment as shown in the following path
                  *\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps
```