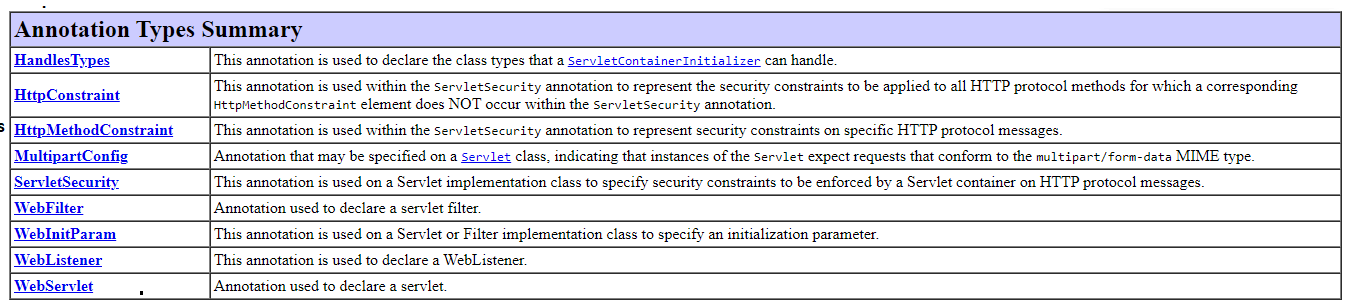
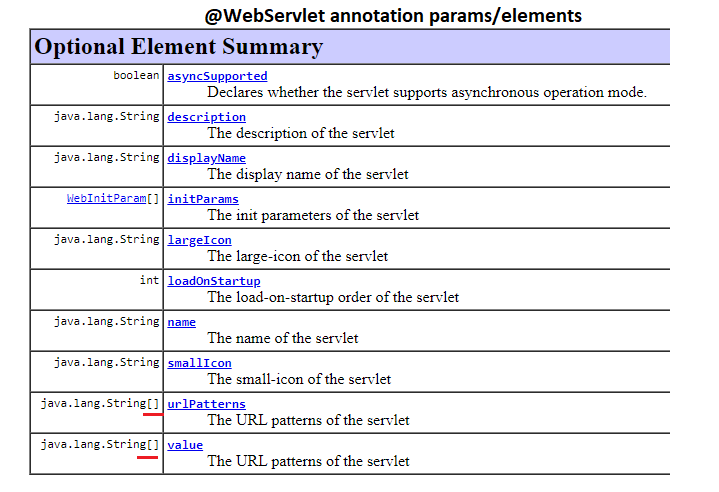
All the annotations of servlet API is given in jakarta.servlet.annotation package. This package have total 9 annotations.



[**1.@WebServlet:-**](mailto:1.@WebServlet:-) Instead of performing servlet mapping configuration using separate web.xml file, we can do same configuration using **@WebServlet** annotation. We should write the servlet configuration on top of the servlet component class.

Syntax:

@WebServlet(paramname=value,…etc)



* In any annotation, data given with out parameter name directly will be assigned to value parameter.

@WebServlet(“/test”)

@webServlet(value=”/test”)

Both are same.

* Value and urlPatterns are string[] type parameters which can be used to provide the multiple url patterns to single servlet.

@WebServlet({“/test”,”/test-1”,..etc})

@WebServlet(value={“/test”,”/test-1”,..etc})

@WebServlet(urlPatterns={“/test”,”/test-1”,..etc})

All annotations are same.

* We set logical name to servlet component using “name” parameter.

@WebServlet(name=”rock”)

* We supply a values to ServletConfig object by “initparams” parameter.

This parameter is WebInitParam[] type array .

@WebServlet(initparam={@WebInitParam(name=”p1”,value=”sv”),

@WebInitParam(name=”p2”,value=”sv1”),…etc})

* We specify the “early-loading” by the “loadOnStartUp” parameter. It is integer type parameter.

Non-zero represents early-loading.

Example:- This program demonstrates the @WebServlet annotations.

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** jakarta.servlet.ServletConfig;

**import** jakarta.servlet.annotation.WebInitParam;

**import** jakarta.servlet.annotation.WebServlet;

**import** jakarta.servlet.http.HttpServlet;

**import** jakarta.servlet.http.HttpServletRequest;

**import** jakarta.servlet.http.HttpServletResponse;

**@WebServlet(name="FirstAnnotation" , urlPatterns= {"/one","/two"},initParams= {@WebInitParam(name="ename",value="sv"),@WebInitParam(name="salary",value="7000")},loadOnStartup=1)**

**public** **class** FirstAnnotation **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** **void** init(ServletConfig cfg) {

System.***out***.println("EmpName:"+cfg.getInitParameter("ename"));

System.***out***.println("salary:"+cfg.getInitParameter("salary"));

}

**public** **void** service(HttpServletRequest req,HttpServletResponse res) **throws** IOException {

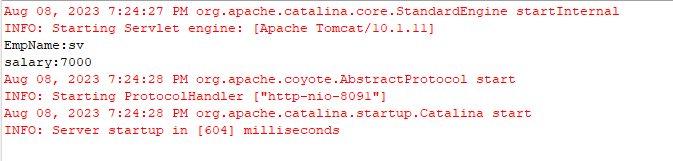
PrintWriter pw=res.getWriter();

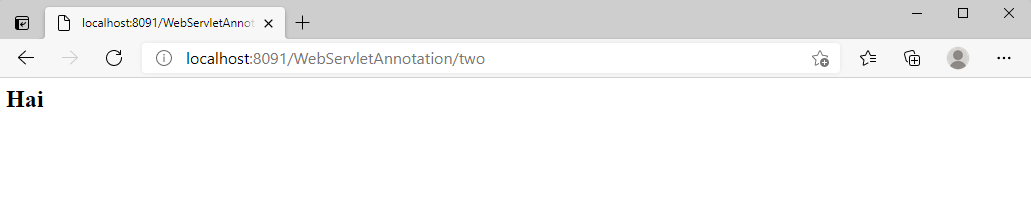
pw.write("<h2>Hai</h2>");

}

}

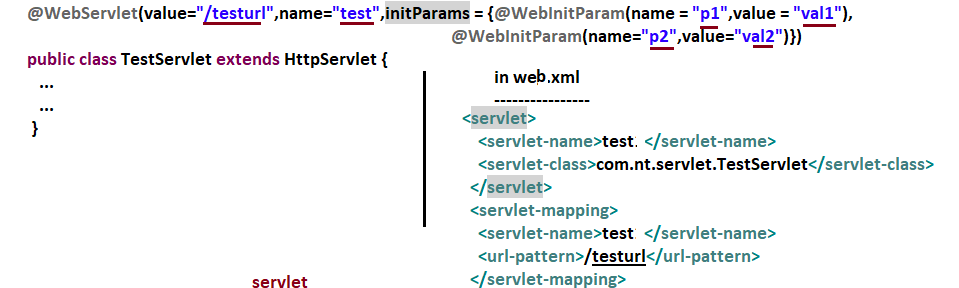
Output:-





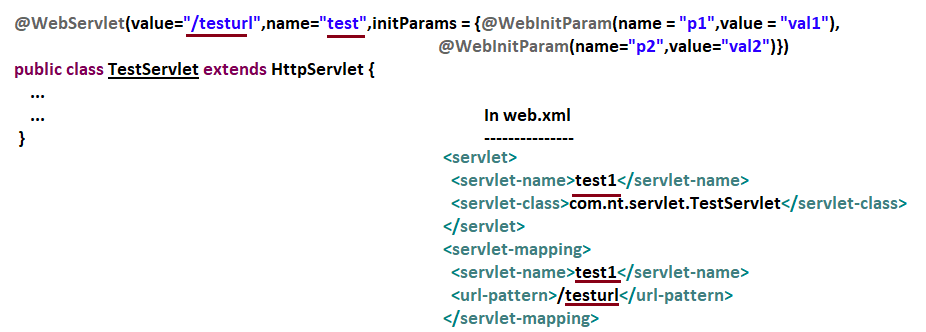
Q.What happens if we cfg the same servlet component using annotations and web.xml which have same logical name and same url pattern?

A. The Servlet Component executes smoothly.



Q. what happens if we cfg same servlet component using annotations and web.xml which have same url patterns and different logical name?

A. Java.lang.illegalArgumentException will be thrown.



Q. what happens if the same servlet component is cfg using annotations and web.xml having different logical names and different url patterns?



Note:-In realtime, if the servlet comp is cfg using annotations then it will not cfg using web.xml and vice-versa.

**Thumb Rules:-**

1. As of now all configurations are not possible using annotations. So the cfgs that are possible with annotations do it using annotations and remaining cfgs do it using web.xml file cfgs.

Cfg using web.xml file in annotation driven cfgs

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

1. Welcome files cfg.
2. Security cfgs
3. Context params.
4. Session management cfgs.
5. …etc

Note: current trend of servlet configurations are xml cfgs+annotations.