|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Servlet API  |  | | --- | | The javax.servlet and javax.servlet.http packages represent interfaces and classes for servlet api. |  Interfaces in javax.servlet package  |  | | --- | | There are many interfaces in javax.servlet package. They are as follows:   1. Servlet 2. ServletRequest 3. ServletResponse 4. RequestDispatcher 5. ServletConfig 6. ServletContext 7. SingleThreadModel |  Classes in javax.servlet package  |  | | --- | | There are many classes in javax.servlet package. They are as follows:   1. GenericServlet 2. ServletInputStream 3. ServletOutputStream 4. ServletRequestWrapper 5. ServletException 6. UnavailableException |  Interfaces in javax.servlet.http package  |  | | --- | | There are many interfaces in javax.servlet.http package. They are as follows:   1. HttpServletRequest 2. HttpServletResponse 3. HttpSession |  Classes in javax.servlet.http package  |  | | --- | | There are many classes in javax.servlet.http package. They are as follows:   1. HttpServlet 2. Cookie 3. HttpServletRequestWrapper | |
| Servlet Interface Servlet interface provides common behaviour to all the servlets.  Servlet interface needs to be implemented for creating any servlet (either directly or indirectly). It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods. Methods of Servlet interface  1. **init()** 2. **service()** 3. **destroy()** 4. getServletConfig() 5. **getServletInfo()** |
| Life Cycle of a Servlet (Servlet Life Cycle)  |  | | --- | | The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:   1. Servlet class is loaded. 2. Servlet instance is created. 3. init method is invoked. 4. service method is invoked. 5. destroy method is invoked. |  Life cycle of a servlet  |  | | --- | | As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state. |  1) Servlet class is loaded  |  | | --- | | The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container. |  2) Servlet instance is created  |  | | --- | | The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle. |  3) init method is invoked  |  | | --- | | The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below: |  1. public void init(ServletConfig config) throws ServletException  4) service method is invoked  |  | | --- | | The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below: |  1. public void service(ServletRequest request, ServletResponse response) 2. throws ServletException, IOException  5) destroy method is invoked  |  | | --- | | The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below: |  1. public void destroy()  |  | | --- | |  |  How to create servlet in Java=============================There are 3 ways to create servlet in java1) by implementing Servlet Interface2) by Extending GenericServlet Class3) by extending HttpServlet class=========================================================by implementing Servlet Interface=================================import javax.servlet.\*;import java.io.\*;public class Register implements Servlet{int a,b,c;public void init(ServletConfig cfg){a=10;b=20;}public void destroy(){a=0;b=0;}public void service(ServletRequest req,ServletResponse res){res.setContentType("text/html");String a= req.getParameter("t1");// database connectivityPrintWriter pw=res.getWriter();pw.print("You are Register ......");}public ServletConfig getServletConfig(){}public String getServletInfo(){}}2).by Extendting GenericServlet Class======================================import javax.servlet.\*;import java.io.\*;public class Register extends GenericServlet{public void service(ServletRequest req,ServletResponse res){res.setContentType("text/html");String a= req.getParameter("t1");// database connectivityPrintWriter pw=res.getWriter();pw.print("You are Register ......");}}3) by Extendting HttpServlet Class======================================import javax.servlet.http.\*;import java.io.\*;public class Register extends HttpServlet{public void doPost(HttpServletRequest req,HttpServletResponse res){res.setContentType("text/html");String a= req.getParameter("t1");// database connectivityPrintWriter pw=res.getWriter();pw.print("You are Register ......");}} |