Servlet生命周期分为三个阶段：

　　1，初始化阶段  调用init()方法

　　2，响应客户请求阶段　　调用service()方法

　　3，终止阶段　　调用destroy()方法

Servlet初始化阶段：

　　在下列时刻Servlet容器装载Servlet：

　　　　1，Servlet容器启动时自动装载某些Servlet，实现它只需要在web.XML文件中的<Servlet></Servlet>之间添加如下代码：

|  |
| --- |
| <loadon-startup>1</loadon-startup> |

　　　　2，在Servlet容器启动后，客户首次向Servlet发送请求

　　　　3，Servlet类文件被更新后，重新装载Servlet

　　Servlet被装载后，Servlet容器创建一个Servlet实例并且调用Servlet的init()方法进行初始化。在Servlet的整个生命周期内，init()方法只被调用一次。

Servlet工作原理：

　　首先简单解释一下Servlet接收和响应客户请求的过程，首先客户发送一个请求，Servlet是调用service()方法对请求进行响应的，通过源代码可见，service()方法中对请求的方式进行了匹配，选择调用doGet,doPost等这些方法，然后再进入对应的方法中调用逻辑层的方法，实现对客户的响应。在Servlet接口和GenericServlet中是没有doGet,doPost等等这些方法的，HttpServlet中定义了这些方法，但是都是返回error信息，所以，我们每次定义一个Servlet的时候，都必须实现doGet或doPost等这些方法。

　　每一个自定义的Servlet都必须实现Servlet的接口，Servlet接口中定义了五个方法，其中比较重要的三个方法涉及到Servlet的生命周期，分别是上文提到的init(),service(),destroy()方法。GenericServlet是一个通用的，不特定于任何协议的Servlet,它实现了Servlet接口。而HttpServlet继承于GenericServlet，因此HttpServlet也实现了Servlet接口。所以我们定义Servlet的时候只需要继承HttpServlet即可。

　　Servlet接口和GenericServlet是不特定于任何协议的，而HttpServlet是特定于HTTP协议的类，所以HttpServlet中实现了service()方法，并将请求ServletRequest,ServletResponse强转为HttpRequest和HttpResponse。

|  |
| --- |
| public void service(ServletRequest req,ServletResponse res)    throws ServletException,IOException  {        HttpRequest request;        HttpResponse response;         try       {           req = (HttpRequest)request;           res = (HttpResponse)response;        }catch(ClassCastException e)        {           throw new ServletException("non-HTTP request response");        }        service(request,response);  } |

    代码的最后调用了HTTPServlet自己的service(request,response)方法，然后根据请求去调用对应的doXXX方法，因为HttpServlet中的doXXX方法都是返回错误信息，

|  |
| --- |
| protected void doGet(HttpServletRequest res,HttpServletResponse resp)    throws ServletException,IOException  {     String protocol = req.getProtocol();     String msg = IStrings.getString("http.method\_get\_not\_supported");     if(protocol.equals("1.1"))     {        resp.sendError(HttpServletResponse.SC.METHOD.NOT.ALLOWED,msg);      }     esle      {        resp.sendError(HttpServletResponse.SC\_BAD\_REQUEST,msg);      }  } |

所以需要我们在自定义的Servlet中override这些方法！

**源码面前，了无秘密！**

---------------------------------------------------------------------------------------------------------------------------------

Servlet响应请求阶段：

　　对于用户到达Servlet的请求，Servlet容器会创建特定于这个请求的ServletRequest对象和ServletResponse对象，然后调用Servlet的service方法。service方法从ServletRequest对象获得客户请求信息，处理该请求，并通过ServletResponse对象向客户返回响应信息。

　　对于Tomcat来说，它会将传递过来的参数放在一个Hashtable中，该Hashtable的定义是：

|  |
| --- |
| private Hashtable<String String[]> paramHashStringArray = new Hashtable<String String[]>(); |

　　这是一个String-->String[]的键值映射。

　　HashMap线程不安全的，Hashtable线程安全。

-----------------------------------------------------------------------------------------------------------------------------------

Servlet终止阶段：

　　当WEB应用被终止，或Servlet容器终止运行，或Servlet容器重新装载Servlet新实例时，Servlet容器会先调用Servlet的destroy()方法，在destroy()方法中可以释放掉Servlet所占用的资源。

-----------------------------------------------------------------------------------------------------------------------------------

Servlet何时被创建：

　　1，默认情况下，当WEB客户第一次请求访问某个Servlet的时候，WEB容器将创建这个Servlet的实例。

　　2，当web.xml文件中如果<servlet>元素中指定了<load-on-startup>子元素时，Servlet容器在启动web服务器时，将按照顺序创建并初始化Servlet对象。

　　注意：在web.xml文件中，某些Servlet只有<serlvet>元素，没有<servlet-mapping>元素，这样我们无法通过url的方式访问这些Servlet，这种Servlet通常会在<servlet>元素中配置一个<load-on-startup>子元素，让容器在启动的时候自动加载这些Servlet并调用init()方法，完成一些全局性的初始化工作。

Web应用何时被启动：

　　1，当Servlet容器启动的时候，所有的Web应用都会被启动

　　2，控制器启动web应用

-----------------------------------------------------------------------------------------------------------------------------------------------

Servlet与JSP的比较：

　　有许多相似之处，都可以生成动态网页。

　　JSP的优点是擅长于网页制作，生成动态页面比较直观，缺点是不容易跟踪与排错。

　　Servlet是纯Java语言，擅长于处理流程和业务逻辑，缺点是生成动态网页不直观。