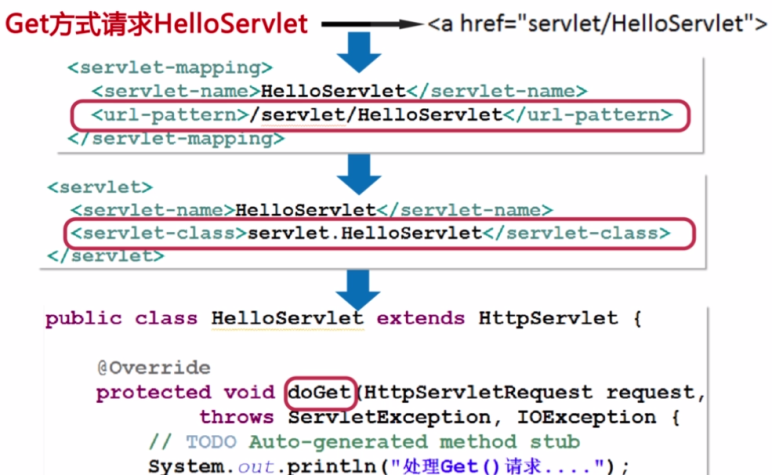
[**servlet执行流程和生命周期**](http://www.cnblogs.com/Wonghy/p/5542277.html)

**参考：**[**https://www.cnblogs.com/Wonghy/p/5542277.html**](https://www.cnblogs.com/Wonghy/p/5542277.html)

**一、servlet执行流程：**



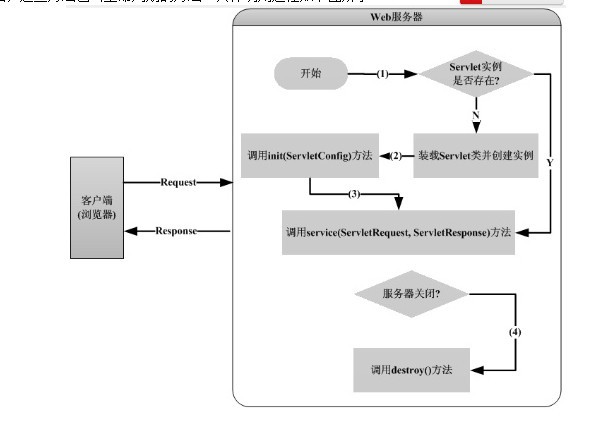
**二、生命周期：**

Servlet的生命周期可以分为四个阶段，即**装载类及创建实例阶段、初始化阶段、服务阶段和实例销毁阶段**。

　　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，由service()方法根据提交的方式选择执行doGet()或者doPost()方法

　　3、终止阶段　　调用destroy()方法

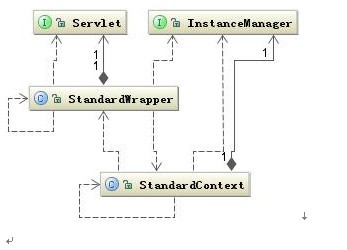


**1、创建servlet实例：**

在默认情况下Servlet实例是在第一个请求到来的时候创建，以后复用。如果有的Servlet需要复杂的操作需要载初始化时完成，比如打开文件、初始化网络连接等，可以通知服务器在启动的时候创建该Servlet的实例。具体配置如下：

    <servlet>  
      <servlet-name>TimeServlet</servlet-name>  
      <servlet-class>com.allanlxf.servlet.basic.TimeServlet</servlet-class>  
      **<load-on-startup>1</load-on-startup>**</servlet>

创建servlet对象的相关类结构：



**2、初始化**

        一旦Servlet实例被创建，Web服务器会自动调用init(ServletConfig config)方法来初始化该Servlet。其中方法参数config中包含了Servlet的配置信息，比如初始化参数，该对象由服务器创建。

I.**如何配置Servlet的初始化参数？**

   在web.xml中该Servlet的定义标记中，比如：

    <servlet>  
         <servlet-name>TimeServlet</servlet-name>  
         <servlet-class>com.allanlxf.servlet.basic.TimeServlet</servlet-class>  
        **<init-param>  
            <param-name>user</param-name>  
            <param-value>username</param-value>  
       </init-param>  
       <init-param>  
           <param-name>blog</param-name>  
           <param-value>http://。。。</param-value>  
       </init-param>**    </servlet>

配置了两个初始化参数user和blog它们的值分别为**username**和[http://](http://allanlxf.blog.sohu.com/)。。。， 这样以后要修改用户名和博客的地址不需要修改Servlet代码，只需修改配置文件即可。

**II.如何读取Servlet的初始化参数？**

       ServletConfig中定义了如下的方法用来读取初始化参数的信息：

       public String getInitParameter(String name)

          参数：初始化参数的名称。  
          返回：初始化参数的值，如果没有配置，返回null。

**III.init(ServletConfig)方法执行次数**

       在Servlet的生命周期中，该方法执行一次。

**IV.init(ServletConfig)方法与线程**

该方法执行在单线程的环境下，因此开发者不用考虑线程安全的问题。

**V.init(ServletConfig)方法与异常**

该方法在执行过程中可以抛出ServletException来通知Web服务器Servlet实例初始化失败。一旦ServletException抛出，Web服务器不会将客户端请求交给该Servlet实例来处理，而是报告初始化失败异常信息给客户端，该Servlet实例将被从内存中销毁。如果在来新的请求，Web服务器会创建新的Servlet实例，并执行新实例的初始化操作

**3、服务**  
    一旦Servlet实例成功创建及初始化，该Servlet实例就可以被服务器用来服务于客户端的请求并生成响应。在服务阶段Web服务器会调用该实例的service(ServletRequest request, ServletResponse response)方法，request对象和response对象有服务器创建并传给Servlet实例。request对象封装了客户端发往服务器端的信息，response对象封装了服务器发往客户端的信息。

**I. service()方法的职责**

     service()方法为Servlet的核心方法，客户端的业务逻辑应该在该方法内执行，典型的服务方法的开发流程为：

    解析客户端请求-〉执行业务逻辑-〉输出响应页面到客户端

**II.service()方法与线程**

     为了提高效率，Servlet规范要求一个Servlet实例必须能够同时服务于多个客户端请求，即service()方法运行在多线程的环境下，Servlet开发者必须保证该方法的线程安全性。

**III.service()方法与异常**

     service()方法在执行的过程中可以抛出ServletException和IOException。其中ServletException可以在处理客户端请求的过程中抛出，比如请求的资源不可用、数据库不可用等。一旦该异常抛出，容器必须回收请求对象，并报告客户端该异常信息。IOException表示输入输出的错误，编程者不必关心该异常，直接由容器报告给客户端即可。

编程注意事项说明：

1) 当Server Thread线程执行Servlet实例的init()方法时，所有的Client Service Thread线程都不能执行该实例的service()方法，更没有线程能够执行该实例的destroy()方法，因此Servlet的init()方法是工作在单线程的环境下，开发者不必考虑任何线程安全的问题。

2) 当服务器接收到来自客户端的多个请求时，服务器会在单独的Client Service Thread线程中执行Servlet实例的service()方法服务于每个客户端。此时会有多个线程同时执行同一个Servlet实例的service()方法，因此必须考虑线程安全的问题。

3) 请大家注意，虽然service()方法运行在多线程的环境下，并不一定要同步该方法。而是要看这个方法在执行过程中访问的资源类型及对资源的访问方式。分析如下：

     i. 如果service()方法没有访问Servlet的成员变量也没有访问全局的资源比如静态变量、文件、数据库连接等，而是只使用了当前线程自己的资源，比如非指向全局资源的临时变量、request和response对象等。该方法本身就是线程安全的，不必进行任何的同步控制。

      ii. 如果service()方法访问了Servlet的成员变量，但是对该变量的操作是只读操作，该方法本身就是线程安全的，不必进行任何的同步控制。

      iii. 如果service()方法访问了Servlet的成员变量，并且对该变量的操作既有读又有写，通常需要加上同步控制语句。

      iv. 如果service()方法访问了全局的静态变量，如果同一时刻系统中也可能有其它线程访问该静态变量，如果既有读也有写的操作，通常需要加上同步控制语句。

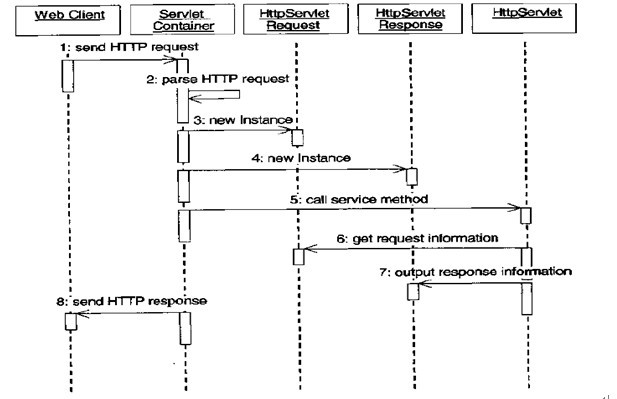
      v. 如果service()方法访问了全局的资源，比如文件、数据库连接等，通常需要加上同步控制语句。

**4、销毁**

     当Web服务器认为Servlet实例没有存在的必要了，比如应用重新装载，或服务器关闭，以及Servlet很长时间都没有被访问过。服务器可以从内存中销毁（也叫卸载）该实例。Web服务器必须保证在卸载Servlet实例之前调用该实例的destroy()方法，以便回收Servlet申请的资源或进行其它的重要的处理。

  Web服务器必须保证调用destroy()方法之前，让所有正在运行在该实例的service()方法中的线程退出或者等待这些线程一段时间。一旦destroy()方法已经执行，Web服务器将拒绝所有的新到来的对该Servlet实例的请求，destroy()方法退出，该Servlet实例即可以被垃圾回收。

四、servlet解析客户端http请求流程图：



1.     web客户向Servlet容器发出HTTP请求;

2.     Servlet容器解析web的HTTP请求.

3.     Servlet容器创建一个HttpRequest对象,在这个对象中封装了http请求信息;

4.     Servlet容器创建一个HttpResponse对象;

5.     Servlet容器（如果访问的该servlet不是在服务器启动时创建的，则先创建servlet实例并调用init()方法初始化对象）调用HttpServlet的service()方法,把HttpRequest和HttpResponse对象为service方法的参数传给HttpServlet对象;

6.     HttpServlet调用HttpRequest的有关方法,获取HTTP请求信息;

7.     HttpServlet调用HttpResponse的有关方法,生成响应数据;

8.    Servlet容器把HttpServlet的响应结果传给web客户.

**三、Tomcat装载servlet三种方法：**

**1、Serlvet容器启动时自动装载某些Servlet,实现它只需要在web.xml文件中<servlet>  </servlet>之间添加<load-on-startup>1</load-on-startup>，数字越小，优先级越大。**

**2、在Servlet容器启动时，客户首次向Servlet发送请求**

**3、Servlet类文件被更新后，重新装载Servlet**