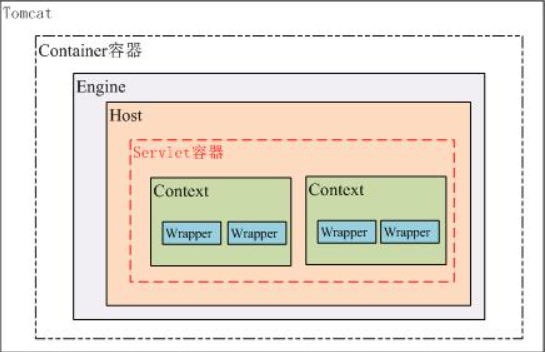
**一，Servlet容器**

**Tomcat有两个作用——web服务器+Servlet容器**

我们一般使用Tomcat作为Servlet的容器，但实际上Tomcat容器有多层，如下图

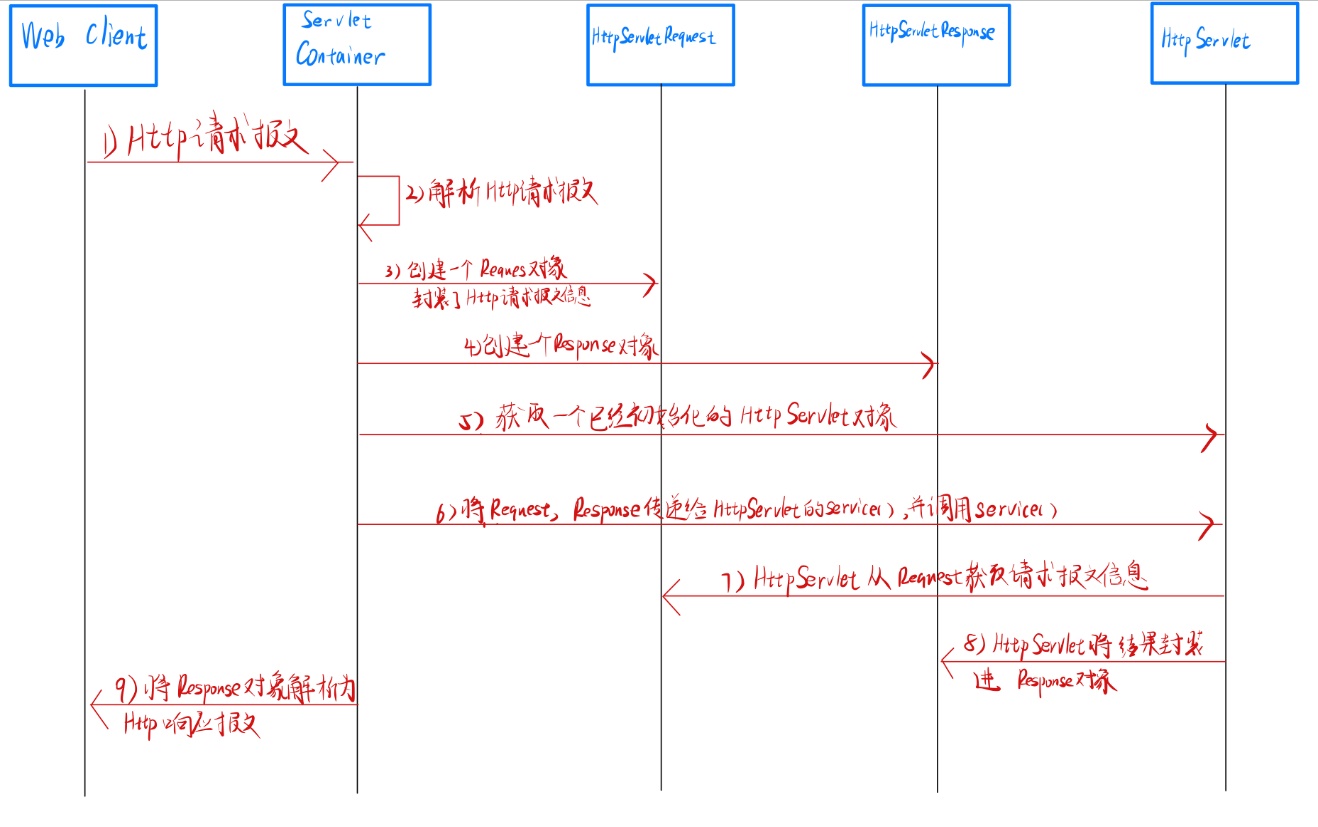


**真正用来管理Servlet的是Context容器，一个项目对应一个Context容器**

**所以一个Tomcat里面可能存放有多个JavaWeb项目**

**二，Servlet工作原理**

**Tomcat既web server，也是Servlet container**

****

**假如我在游览器地址栏输入Web地址localhost:8080/MyJavaWebDemo/first**

**1）**

**①Web Client生成一条Http请求报文，用于请求服务器上的资源**

**②然后Client会解析localhost:8080找到指定的web server**

**③然后将Http请求报文传递给 web server（也就是Tomcat）**

**2)**

**Web Server接受Client的Http请求报文，解析Http请求报文**

**3)**

**Servlet Container解析完Http请求报文，会生成一个HttpServletRequest对象**

**HttpServletRequest对象封装了了请求报文的全部信息**

**4)**

**同时Servlet Container生成一个HttpServletResponse对象，用于封装响应信息**

**5）**

**获取完成初始化的HttpServlet对象**

**①Servlet Container通过JavaWebDemo找到Serlvet容器中的JavaWebDemo项目**

**然后检查是否已经存在HttpServlet对象**

**②如果对象已经存在跳至 6）**

**③如果对象不存在，那么Sevlet容器会找到JavaWebDemo项目下的配置文件web.xml**

**根据first，匹配web.xml中的 <servlet-mapping>中的<url-patten>**

**找到匹配的<servlet-mappring>,取出里面的<servlet-name>**

**然后找到具有相同<servlet-name>的<servlet>，从中取出<servlet-class>**

**然后利用<servlet-class>反射获得HttpServlet对象**

**最后，调用该HttpServlet对象的init()方法，完成HttpServlet对象的初始化操作**

**6）获取对应的HttpServlet对象后，Servlet容器将HttpServletRequest对象，HttpServletResponse对象，传递给HttpServlet对象的service( )，调用service( )**

**7）HttpServlet的service( )会从HttpServletRequest对象中获取请求报文信息，并进行处理**

**8）HttpServlet的service()将结果封装进HttpServletResponse对象**

**9）Web Server将HttpServletResponse对象解析为响应报文，发送给Client**

**然后删除Request对象和Response对象**

**三，Servlet对象的生命周期**

**1）Servlet部署**

**如果想要让客户端可以访问Servlet，那么必须在web.xml配置文件中对Servlet进行部署**

**每个Servlet类在web.xml配置文件中,都有<servlet>和<servlet-mappring>两个元素**

**①<servlet>用于注册Servlet，包含了<servlet-name>和<servlet-class>两个子元素**

**一个设置servlet的名称，一个必须为Servlet的完全限定名（包名.类名）**

**②<servlet-mappring>用于映射<servlet>中的对外访问路径**

**有<servlet-name>和<url-patten>两个子元素**

**一个设置servlet的名称（必须与<servlet>的相同），一个设置servlet对外访问路径**

**2）Servlet对象的生命周期**

**①Servlet对象的创建——（注意Servlet都是单例设计模式）**

**Servlet容器接收Client的请求后，会检查是否存在对应的Servlet对象**

**如果已经存在Servlet对象，那么不在创建，执行③（单例设计模式）**

**如果不存在：通过web.xml配置文件，获取Servlet的完全限定名(<servlet>中的<servlet-class>)**

**然后通过反射，创建一个Servlet实例对象，执行②**

**②Servlet对象的初始化——init( )**

**Servlet容器调用Servlet对象的init( )方法，完成Servlet对象的初始化**

**③请求处理——service( )**

**Servlet容器将Request对象和Response对象传递给 Servlet对象的service( )方法，并调用service( )，Client每请求一次Servlet，都会调用一个service( )**

**④销毁——destory( )**

**当Servlet容器决定将一个Servlet对象删除，或者当Servlet容器即将关闭或重新启动，会调用Servlet对象的destory( )来销毁该Servlet对象**

**注意：**

**①创建，初始化，销毁在Servlet的整个声明周期中只会执行一次**

**②只有Client首次请求Servlet时，才会进行Servlet对象的创建，初始化操作**

**后续对该Servlet的请求，只会直接调用Servlet对象的service( )**

**③默认情况下，只有在 Client首次请求Servlet时，才会进行Servlet对象的创建和初始化**

**但是我们可以通过在<servlet>中添加子元素<load-on-startup>，使得Servlet对象在Servlet容器启动时完成创建和初始化**

**四，Servlet的单例多线程，和线程安全**

**1）单例**

**从Servlet的生命周期，我们可以看出，Servlet采用的是单例设计模式**

**2）多线程**

**对于每一个请求，Servlet容器都会其创建一个Request对象和Response对象**

**并且会为每一个请求分配一个线程，来执行后续操作**

**3）线程安全**

**所以，如果多个客户端请求同一个Servlet对象，那么Servlet容器会为每个请求都创建一个线程，这多个线程会调用同一个Servlet对象的service( )，这就可能造成线程安全问题**

**所以，使用Servlet最好保证Servlet是无状态的，也就是没有可以修改的成员变量**