# Servlet学习

相关链接：

https://www.cnblogs.com/GtShare/p/8033637.html

https://www.aliyun.com/jiaocheng/306417.html

## Servlet生命周期

servlet生命周期：加载（web服务完成）🡪实例化🡪服务🡪销毁

-1）init()方法

在Servlet的生命周期中，仅此执行一次init()方法。用于初始化Servlet对象。

-2）service()方法

Servlet的核心，负责处理和响应客户端的请求。

每当一个客户请求一个HttpServlet对象，该对象的service()方法就要调用，而且传递给这个方法一个请求对象ServletRequest和一个响应对象ServletResponse作为参数。

默认的服务功能是调用与http请求的方法相应的doGet/doPost功能。

-3）destroy()方法

仅执行一次，在服务器端停止且卸载Servlet时执行该方法。用于释放占用的资源。

## Servlet是单例的

在web访问中，每多一个用户，会在web容器中多一个用户线程。

多用户同时访问某个servlet的doget方法时，由于方法的执行在线程中是私有的，每个线程执行某个方法不影响。

servlet是单例的，init()方法只在servlet加载时执行，destroy()只在销毁时执行。

## Servlet处理多个请求访问

servlet容器默认采用“单实例多线程”方式处理多个请求：

-1）当web服务器启动时（或客户端第一次请求某个Servlet），Servlet容器将会根据web.xml配置文件实例化该Servlet类（只存在一个Servlet实例）。

-2）容器初始化Servlet主要就是读取配置文件（如tomcat可以通过server.xml的<Connector>设置线程池中线程数目，初始化线程池通过web.xml初始化每个参数值等等）。

-3）当请求到达时，Servlet容器通过调度线程(Dispatcher Thread)调度它管理下线程池中等待执行的线程(Work Thread)给请求者。

-4）线程执行Servlet的service()方法

-5）请求结束，放回线程池，等待被调用

好处：

-1）Servlet单例，减少了创建servlet的开销

-2）通过线程池来响应多个请求，提高了请求的响应时间

-3）servlet容器并不关心到达的Servlet请求访问的是否为同一Servlet，直接分配它一个新的线程。若是同一个Servlet的多个请求，那么该Servlet的service方法将在多线程中并发的执行。

-4）每一个请求由ServletRequest对象来接受请求，由ServletResponse对象来响应该请求。

注：避免使用成员变量

若存在成员变量，可能发生多线程同时访问和操作该变量，照成数据的不一致，因此产生线程安全问题。

## Java的内存模型JMM（Java Memory Model）

JMM主要是为了规定了线程和内存之间的一些关系。

根据JMM的设计，系统存在一个主内存（Main Memory）和多个工作内存（Working Memory）。

主内存：

Java中所有实例变量都存储在主内存中，对于所有线程都是共享的。

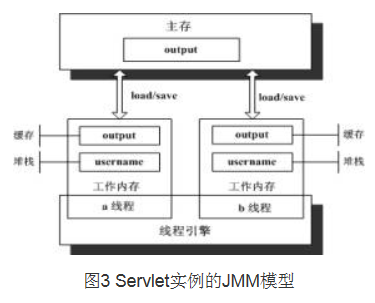
工作内存：

每条线程都有自己的工作内存，工作内存由缓存和堆栈两部分组成。

缓存中保存的是主内存中变量的拷贝，缓存可能并不和主存同步，即缓存中变量的修改可能没有立即写到主存中；

堆栈中保存的是线程的局部变量，线程之间无法相互直接访问堆栈中的变量。

Servlet实例的内存模型抽象：



## 工作者线程和调度线程

工作者线程（Work Thread）：执行代码的一组线程

调度线程（Dispatcher Thread）：每个线程都具有分配给它的线程优先级，线程是根据优先级调度执行的。

Servlet采用多线程来处理多个请求同时访问。Servlet依赖于一个线程池来服务请求，线程池实际上是一系列的工作者线程集合。Servlet使用一个调度线程来管理工作者线程。

当容器收到一个Servlet请求，调度线程从线程池中选出一个工作者线程，将请求传递给该工作者线程，然后由该线程来执行Servlet的service()方法。当这个线程正在执行时，容器收到另一请求，调度线程同样从线程池中选择另一工作者线程来执行Servlet的service()方法。容器并不关心这个请求是否访问的是同一Servlet，当容器同时接收对同一Servlet的多个请求时，该Servlet的service()方法将在多线程中并发执行。

Servlet容器默认采用单实例多线程的方式来处理请求的好处：减少产生Servlet实例的开销，提升了对请求的响应时间。

对于Tomcat可以在server.xml中通过<Connector>元素设置线程池中线程的数目。

## Servlet是非线程安全的

Servlet体系结构是建立在Java多线程机制之上的，其生命周期由Web容器负责。

当客户端第一次请求某个Servlet时，Servlet容器将会根据web.xml配置文件实例化该Servlet类。当有新的客户端请求该Servlet时，一般不会再实例化该Servlet类，即有多个线程在使用这个实例。Servlet容器会自动使用线程池等技术来支持系统的运行。

当两个或多个线程同时访问同一个Servlet时，可能会发生多个线程同时访问同一资源的情况，数据可能会变得不一致。

避免非线程安全问题：

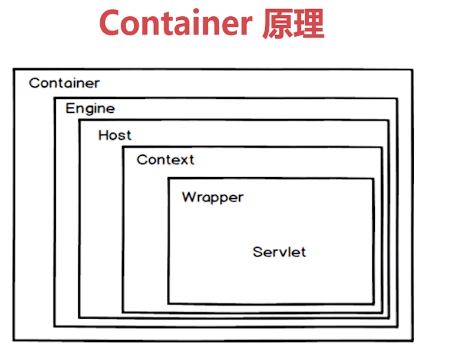
如果一个Servlet实现了SingleThreadModel接口，Servlet引擎将为每个新的请求创建一个单独的Servlet实例，这将引起大量的系统开销。（Servlet2.4不推荐使用）

线程安全问题主要是由实例变量造成的，因此在Servlet中应避免使用实例变量。

如果应用程序设计无法避免使用实例变量，那么使用同步synchronized来保护要使用的实例变量，但为保证系统的最佳性能，应该同步可用性最小的代码路径。

## Servlet在Tomcat的位置

Container组件：



Server处理Http请求：

