# 线程池基本原理

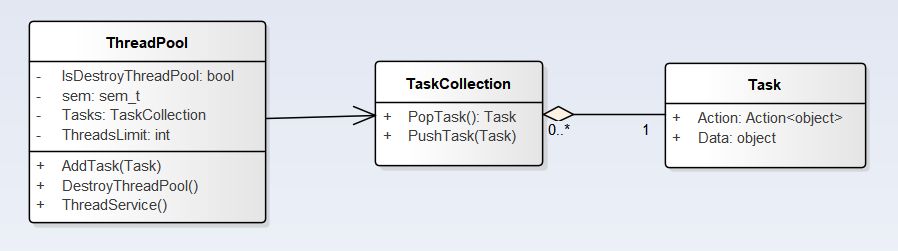
频繁地开辟与销毁线程极大地占用了系统的资源，[线程池](http://baike.baidu.com/link?url=6duZ8Z5rnpq0FVN6rbyMVAS2JUruSmLj-mrVE9T-uGzfAGzN5rE3I6T6yR5fUVNlELZGl2NB_UbEjNOKPYGpYa" \t "https://blog.csdn.net/lianghe_work/article/details/_blank)提供了一个解决外部大量用户与服务器有限资源的矛盾。

[线程池](http://baike.baidu.com/link?url=6duZ8Z5rnpq0FVN6rbyMVAS2JUruSmLj-mrVE9T-uGzfAGzN5rE3I6T6yR5fUVNlELZGl2NB_UbEjNOKPYGpYa" \t "https://blog.csdn.net/lianghe_work/article/details/_blank)的基本思想就是在程序开始时就在内存中开辟一些线程，线程的数目是固定的，他们独自形成一个类，屏蔽了对外的操作，而服务器只需要将数据包交给线程池就可以了。当有新的客户请求到达时，不是新创建一个线程为其服务，而是从“池子”中选择一个空闲的线程为新的客户请求服务，服务完毕后，线程进入空闲线程池中。如果没有线程空闲的话，就将数据包暂时积累， 等待线程池内有线程空闲以后再进行处理。

# 

# 线程池设计

线程池设计也是比较简单，如下：



IsDestroyThreadPool：线程池是否已被销毁

Sem: 信号量

Tasks：未处理的任务

ThreadsLimit：线程池线程的数量

ThreadPool构造函数流程：

1. 根据传入的ThreadsLimit设置属性ThreadsLimit
2. 开启ThreadsLimit个线程，线程使用的函数是ThreadService

AddTask函数流程：

1. 将Task添加到Tasks中
2. 将信号量Sem加1，表示有一个未处理任务

ThreadService函数流程

1. 等待信号量，如果获取到信号量，信号量会自动减1，往下执行
2. 判断IsDestroyThreadPool是否为true，是则退出线程，否则往下执行
3. 从Takes中取出一个Task
4. 执行Task的Action方法
5. 返回步骤1

DestroyThreadPool函数流程：

1. IsDestroyThreadPool设为true
2. 增加ThreadsLimit个信号（因为最多有ThreadsLimit个线程在等待信号），收到信号的线程将会退出，而Takes中的任务，将不会被处理

注：TaskCollection在添加任务和取任务时，应该要加锁