# URL

## URL的使用

## URLConnection的使用

## HttpURLConnection的使用

# Socket

Socket中文叫做套接字.是一个虚拟的.Socket的对象一旦创建一般都是一式两份.A和B两端各保留一份.A和B的所有通信,都是通过这个Socket互相进行读写来实现的.

想要使用Socket都需要有个服务端:ServerSocket.

## TCP模型

## UDP模型

## TCP和UDP区别

# 课堂练习

1. 对已有代码做出改进.效果为可以一直聊天直到一方输入bye.才结束程序.
2. 对上面在进行修改.要求绑定成功后.提示客户端需要注册用户名.并且验证用户名是唯一的

# 多人聊天室

聊天室是分群聊和一对一聊天的.整个思路是设计成一个服务端对应多个客户端.

# 多线程断点续传下载

# 线程池

## 为什么要用线程池

线程每次的创建都是比较消耗资源,还消耗时间.并且线程的run方法一旦执行结束.线程就死亡.

使用线程池来减少线程的创建次数.

## Java中常见的线程池

Java中常用的有四种线程池.

1. 带有缓存的线程池
2. 指定工作线程数量的线程池
3. 单例的线程池
4. 定长的线程池

### 带有缓存的线程池

创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。

这种类型的线程池特点是：

* 工作线程的创建数量几乎没有限制(其实也有限制的,数目为Interger. MAX\_VALUE), 这样可灵活的往线程池中添加线程。
* 如果长时间没有往线程池中提交任务，即如果工作线程空闲了指定的时间(默认为1分钟)，则该工作线程将自动终止。终止后，如果你又提交了新的任务，则线程池重新创建一个工作线程。
* 在使用CachedThreadPool时，一定要注意控制任务的数量，否则，由于大量线程同时运行，很有会造成系统瘫痪。

1. /\*\*
2. \*
3. \*/
4. package com.hwua.demo02;
6. import java.util.concurrent.Executor;
7. import java.util.concurrent.ExecutorService;
8. import java.util.concurrent.Executors;
10. /\*\*
11. \* @author Administrator
12. \*
13. \*/
14. **public** **class** Demo01 {
16. **public** **static** **void** main(String[] args) {
17. //-- 新建一个带有缓存的线程池
18. ExecutorService es = Executors.newCachedThreadPool();
20. **for** (**int** i = 0; i < 10; i++) {
21. final **int** index = i;
22. //-- 让线程池去执行某个线程. 需要参数Runnable的对象
23. es.execute(**new** Runnable() {
24. @Override
25. **public** **void** run() {
26. Thread.currentThread().setName("线程-" + (index + 1));
27. System.out.println(index);
28. System.out.println("这是线程内部的打印");
29. System.out.println("--------------------");
30. }
31. });
32. }
33. }
34. }

### 指定线程池容量的

创建一个指定工作线程数量的线程池。每当提交一个任务就创建一个工作线程，如果工作线程数量达到线程池初始的最大数，则将提交的任务存入到池队列中。

FixedThreadPool是一个典型且优秀的线程池，它具有线程池提高程序效率和节省创建线程时所耗的开销的优点。但是，在线程池空闲时，即线程池中没有可运行任务时，它不会释放工作线程，还会占用一定的系统资源。

1. /\*\*
2. \*
3. \*/
4. package com.hwua.demo02;
6. import java.util.concurrent.ExecutorService;
7. import java.util.concurrent.Executors;
9. /\*\*
10. \* @author Administrator
11. \*
12. \*/
13. **public** **class** Demo02 {
15. **public** **static** **void** main(String[] args) {
16. //-- 新建一个带有缓存的线程池
17. ExecutorService es = Executors.newFixedThreadPool(3);
19. **for** (**int** i = 0; i < 10; i++) {
20. final **int** index = i;
21. //-- 让线程池去执行某个线程. 需要参数Runnable的对象
22. es.execute(**new** Runnable() {
23. @Override
24. **public** **void** run() {
25. Thread.currentThread().setName("线程-" + (index + 1));
26. System.out.println(Thread.currentThread().getName());
27. **try** {
28. Thread.sleep(2000);
29. } **catch** (InterruptedException e) {
30. // TODO Auto-generated catch block
31. e.printStackTrace();
32. }
33. }
34. });
35. }
36. }
37. }

因为在创建池子的时候指定了池子的大小为3.配合下面的代码每隔2秒睡眠,所以可以看到每隔2秒打印3次的效果.

# 预习

多线程断点下载

* 1. 随机存储技术RandomAccessFile
  2. 线程池技术
  3. 集合
  4. 网络(URL)

JSON

XML

1. 解析在线天气

数据库(7)

4天MySQL

1天Oracle

2天JDBC

* 1. DBCP
  2. JDBC Template

# 参考

1. [Java中常用的四种线程池介绍](https://www.cnblogs.com/aaron911/p/6213808.html)