

19 线程_设计模式

JAVAAEE高级

一：单例设计模式

- 1、保证类在内存中只有一个对象
- 2/单例写法两种 (饿汉式 / 懒汉式)

1、饿汉式

- 1、饿汉式是空间换时间

```
1.  class Singleton {
2.      // 1私有构造方法其他类不能访问该构造方法了
3.      private Singleton() {
4.      }
5.      // 2创建本类对象
6.      private static Singleton s = new Singleton();
7.      // 3对外提供公共的访问方法
8.      public static Singleton getInstance() { // 获取实例
9.          return s;
10.     }
11. }
```

```
1.  public static void main(String[] args) {
2.      Singleton s1 = Singleton.getInstance();
3.      Singleton s2 = Singleton.getInstance();
4.      System.out.println(s1 == s2);
5.  }
```

2、懒汉式

1、懒汉式是时间换空间

2、懒汉式单例的延迟加载模式

```
1.  class Singleton1 {
2.      // 1私有构造方法其他类不能访问该构造方法了
3.      private Singleton1() {
4.      }
5.      // 2声明一个引用
6.      private static Singleton1 s;
7.      // 3对外提供公共的访问方法
8.      public static Singleton1 getInstance() { // 获取实例
9.          if (s == null) {
10.              // 线程1等待线程2等待
11.              s = new Singleton1();
12.          }
13.          return s;
14.      }
15. }
```

3、三种格式

```
1.  class Singleton2 {
2.      // 1私有构造方法其他类不能访问该构造方法了
3.      private Singleton2() {
4.      }
5.      // 2声明一个引用
6.      public static final Singleton2 s = new Singleton2();
7.  }
```

```
1.      public static void main(String[] args) {
2.          Singleton2 s1 = Singleton2.s;
3.          Singleton2 s2 = Singleton2.s;
4.          System.out.println(s1 == s2);
5.      }
```

二：Runtime类设计模式

1、Runtime类是一个单例类饿汉模式

```
1.     public static void main(String[] args) throws IOException {
2.         Runtime r = Runtime.getRuntime();           //获取运行时对象
3.         r.exec("shutdown -s -t 300");//300秒后关机
4.         r.exec("shutdown -a");//取消关机
5.     }
```

源码解析

```
6 public class Runtime {
7     private static Runtime currentRuntime = new Runtime();
8
9     /**
10      * Returns the runtime object associated with the current Java applic
11      * Most of the methods of class <code>Runtime</code> are instance
12      * methods and must be invoked with respect to the current runtime ob
13      *
14      * @return the <code>Runtime</code> object associated with the curre
15      *         Java application.
16     */
17     public static Runtime getRuntime() {
18         return currentRuntime;
19     }
20
21     /** Don't let anyone else instantiate this class */
22     private Runtime() {}
23 }
```

2、shutdown 的描述

```
管理员: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\admin>shutdown
用法: shutdown [/i | /l | /s | /r | /g | /a | /p | /h | /e] [/f]
      [/m \\computer][[/t xxx]][/d [plu:]xx:yy [/c "comment"]]

没有参数 显示帮助。这与键入 /? 是一样的。
/?        显示帮助。这与不键入任何选项是一样的。
/i        显示图形用户界面(GUI)。
          这必须是第一个选项。
/l        注销。这不能与 /m 或 /d 选项一起使用。
/s        关闭计算机。
/r        关闭并重新启动计算机。
/g        关闭并重新启动计算机。系统重新启动后，
          重新启动所有注册的应用程序。
/a        中止系统关闭。
          这只能在超时期间使用。
/p        关闭本地计算机，没有超时或警告。
          可以与 /d 和 /f 选项一起使用。
/h        休眠本地计算机。
          可以与 /f 选项一起使用。
/e        记录计算机意外关闭的原因。
/m \\computer 指定目标计算机。
/t xxx    设置关闭前的超时为 xxx 秒。
          有效范围是 0-315360000 (10 年)，默认值为 30。
          如果超时时间大于 0，则默示 /f
          参数。
/c "comment" 重新启动或关闭的原因的注释。
          最多允许 512 个字符。
/f        强制正在运行的应用程序关闭，不前台警告用户。
          当为 /t 参数指定大于 0 的值时，
          则默示 /f 参数。
/d [plu:]xx:yy 提供重新启动或关机的原因。
```

三：Timer定时器

```
1. class MyTimerTask extends TimerTask {
2.
3.     @Override
4.     public void run() {
5.         System.out.println("起床背英语单词");
6.     }
7. }
```

```
1. public static void main(String[] args) throws InterruptedException {
2.     Timer t = new Timer();
3.     //在指定时间安排指定任务
4.     //第一个参数:任务;第二个参数:执行的时间;第三个参数是:重复执行
```

```
5.         t.schedule(new MyTimerTask(), new Date(119, 6, 1, 14, 22, 50), 3
000);
6.         while(true) {
7.             Thread.sleep(1000);
8.             System.out.println(new Date());
9.         }
10.    }
```