链滴

理解设计模式之单例模式

作者：michael

原文链接：https://ld246.com/article/1538204780849

来源网站：链滴

什么是单例

之前文章里面写过设计模式总结，里面有一类设计模式，叫创建模式，也就是说用来规范类对象创建的，单例模式就是创建模式其中的一种。我们知道Java创建对象，通过new关键字来创建的话，new一次就会新创建一个与之对应的对象，也就是在JVM堆中会分配内存空间给这个对象使用（Java内模型这块，可以参考周志明写的《深入理解Java虚拟机》）。如果new的对象过多，一是对象占用内多，二是影响GC效率，严重的甚至会影响应用程序响应速度（跟GC有关）。如果说一个对象是无状的，也就是说这个对象本身的属性，经过对象初始化过后，在以后各个线程中执行时候，是不会被修的，那么这个对象就是无状态的，像这种无状态的对象，没有必要每次线程调用都去new一个对象，为对象内部没有随业务变化的属性，因此是需要new一个这样的对象就行了，后期多线程访问就只需访问这一个对象的属性即可。像这种情况，我们需要控制这个类的实例生成，保证整个JVM堆中只有个此类的对象，这种设计模式就叫单例模式，单例意思就是单实例，英文叫SingleTon。

常用的单例模式

这里虽然叫单例模式，但是实现的方式有很多种，看过网上的资料，可能叫法有很多，什么懒汉、饿汉式，但是常用的我只做简单说明，也不取名，懂意思就行。

1、静态属性直接初始化对象，静态方法中直接返回此对象，适合简单对象的初始化，如下所示：

```
public class A{
  private static final A instance=new A();
  private A(){}
  public static A instance(){
    return instance;
  }
}
```

2、复杂一点的对象，单例模式的初始化，一般采用双重校验方式，如下所示：

```
public class A {
    private String field1;

    private static A instance;

    private A() {
    }

    public static A instance() {
        if (instance == null) {
            synchronized (A.class) {
                if (instance == null) {   //第二次校验，防止多线程操作导致对象覆盖
                    instance=new A();
                    instance.setField1("SingleTon"); //初始化对象属性
                }
            }
        }
        return instance;
    }

    public String getField1() {
        return field1;
    }
```

原文链接：理解设计模式之单例模式

```
    public void setField1(String field1) {
        this.field1 = field1;
    }

    ......
}
```

在整个开发过程中，可能使用第二种的情况会比较多，是需要记住的代码方式。

Spring

上面是通过手动写单例模式，不过Spring框架已经为我们提供了单例的容器，Spring的Bean注管理，默认就是单例的。当然可以通过修改Scope来修改单例或者是原型模式，这个可以百度一下具了解。

我们在整个使用设计模式的过程中，一定要明白为什么要使用这种模式，而不是只记住它的代码怎么写的。