【第28话：再也不怕面试官拿出一张纸让我们手写单例模式】

Hello 小伙伴们，这节课给大家讲解下单例设计模式。

单例设计模式在面试中的笔试部分出现频率比较高。有的面试官在面试过程中突然拿过来一张A4纸，让我们手写单例模式。所以单例模式是我们面试时必须要掌握的一项技能。

我们经常在网上看到单例设计模式被分为好多种：有饿汉式、懒汉式、双重锁式、登记式、可见性单例式。可以说五花八门，千奇百怪。其实单例设计模式我们重点关注三个就可以：饿汉式、懒汉式、登记式。

1.饿汉式

```
package com.bjsxt.singleton;

/*
    单例：希望类只有一个
    核心思想：
        1. 构造方法私有
        2. 对外提供一个能够获取对象的方法。

    饿汉式：
        优点：实现简单
        缺点：无论是否使用当前类对象，加载类时一定会实例化。
 */
public class Singleton {
    // 之所以叫做饿汉式：因为类加载时就创建了对象
    private static Singleton singleton = new Singleton();
    private Singleton(){}
    public static Singleton getInstance(){
        return singleton;
    }
}
```

2.懒汉式

```
package com.bjsxt.singleton;

/**
 * 核心思想：
 * 1. 构造方法私有
 * 2. 对外提供一个能够获取对象的方法。
 *
 * 懒汉式优点和缺点：
 *  优点：
 *      按需创建对象。不会在加载类时直接实例化对象。
 *  缺点：
 *      写法相对复杂
 *      多线程环境下，第一次实例化对象效率低。
 */
public class Singleton2 {
```

```
    //懒汉式：不会立即实例化
    private static Singleton2 singleton2;
    private Singleton2() {
    }
    public static Singleton2 getInstance() {
        if (singleton2 == null) {// 不是第一次访问的线程，直接通过if判断条件不成立。直接
return
            synchronized (Singleton2.class) {
                if(singleton2==null) {// 防止多个线程已经执行到synchronized
                    singleton2 = new Singleton2();
                }
            }
        }
        return singleton2;
    }
}
```

3.登记式。这种方式在Spring的Bean管理使用的。思想：把类名当做Key，存储到Map中。

```
public class Singleton3{
    private static Map<String,Singleton3> map=new ConcurrentHashMap<>();
    static{
        Singleton3 single=new Singleton3();
        map.put(single.getClass().getName(),single);
    }
    private Singleton3(){}

    public static Singleton3 getInstance(String name) throws
ClassNotFoundException, IllegalAccessException, InstantiationException {
        if(name==null){
            name=Singleton3.class.getName();
        }
        if(map.get(name)==null){
            map.put(name,(Singleton3)Class.forName(name).newInstance());
        }
        return map.get(name);
    }
}
```

这三种单例模式不需要在笔试时或手写时都写出来。优先记忆懒汉式，如果还需要我们写一个就写登记式。一般最多写两个就够了。