# 单例模式

作者：Yangzzz
原文链接：https://ld246.com/article/1498408582170
来源网站：链滴

**什么是单例模式?**

可以简单的理解在系统中只存在一个实例对象

**适用于什么场合?**

- 一个对象需要频繁创建和销毁
- 对象创建需要消耗一定的资源
- 一个对象即可满足需要

**单例模式优点**： 节约内存，提高性能，避免了大对象频繁创建的消耗

**单例模式缺点**： 不符合开闭原则，该类不支持拓展，当功能变化时修改代码是必然的

**单例的几种实现方式**:

1、 饿汉式

```
public class Sington {

public static Sington sington = new Sington();

public static Sington getInstance() { return sington};

private Sington() {}

}
```

案例： Runtime.getRuntime();

点评： 简单粗暴的实现方式

2、 懒汉式

```
public class Sington {

public static Sington sington = null;

public static Sington getInstance() {

if(sington == null)

sington = new Sington();

return sington;

}

private Sington() {}

}
```

点评：在单线程中是非常合适的但不适合多线程使用，那么怎么解决多线程问题呢，没错，可以使用synchronized关键字，具体实现如下：

```
public class Singtон {

public static Singtон singtон = null;

public static  synchronized Singtон getInstance() {

if(singtон == null)

singtон = new Singtон();

return singtон;

}

private Singtон() {}

}
```

细心的朋友会发现这样确实解决了多线程问题，但是每次调用该方法时都使用了同步机制，这多少会些性能问题，因为其实我们只需要在第一次调用该方法时保证同步即可，那就是接下来的第三种方式。

3、 DCL (double check lock)（推荐使用)

```
public class Singtон {

public static Singtон singtон = null;

public static Singtон getInstance() {

if(singtон == null) {// 该处判断避免不必要的同步

sychronized(Singtон.class) {

if(singtон == null) {// 该处判断保证singtон为null时才创建

singtон = new Singtон();

}

}

}

return singtон;

}

private Singtон() {}

}
```

点评： 这种实现方式是比较推荐的一种实现方式，在很多地方被使用，比如我们常用的ImageLoad片框架便采用了这种实现方式。

注意： 其实在多线程特别复杂的环境下，该方式也不完全能保证正确。这个有机会再讲

4、 静态内部类单例模式 (推荐使用)

```
public class Singtон {

private Singtон() {}

public static Singtон getInstance() {

return SingtonHolder.singtон;

}

private static class SingtonHolder {

private static final Singtон singtон = new Singtон();

}

}
```

点评： 第一次加载Singtон类时并不会实例化Singon,只有在第一次调用getInstance方法时才会创建因为调用该方法虚拟机就会主动加载SingtonHolder类。这种方式几乎完美，既保证唯一性又是线程全的

5、 借助容器实现单例模式

```
public class SingtonManger {

private static HashMap  objMap = new HashMap(String,Object);

private SingtonManger(){};

public static void registerService(String key, Object obj) {

if(!objMap.contains(key))

objMap.put(key,obj);

}

public static Object getService(String key) {

return objMap.get(key)

}

}
```

学安卓的朋友看到这个有没有觉得似曾相识，没错安卓里面的服务就是这种形式。有兴趣可以去看看卓源码

点评： 这种方式适合管理多个单例对象