关于单例模式，相信大家都所有了解，比较经典的实现有饿汉式、借助内部类、双重锁检测，这些实现可以保证线程安全，但是在某些特殊情况下并不能够保证仅仅只有一个单例，因为像序列化、反射攻击等往往可以生成新的实例对象，本文将重点分析枚举单例模式如何防止反射攻击。

**枚举单例：**

**public** **enum** Singleton {

INSTANCE {

@Override

**protected** **void** **read**() {

System.**out**.println("read");

}

@Override

**protected** **void** **write**() {

System.**out**.println("write");

}

};

**protected** **abstract** **void** **read**();

**protected** **abstract** **void** **write**();

}

以上是一个单例枚举的例子，而我们要获取该实例只需要Singleton.INSTANCE，并且此种方式可以保证该单例**线程安全、防反射攻击、防止序列化生成新的实例**。

枚举单例关于防反射攻击，当然和枚举的实现有关，枚举也是java类，我们对Singleton的class进行反编译，可以得到一个新的类（对于枚举的实现不了解的可以补补相关知识）：

**反编译后的类：**

**public** **abstract** **class** **Singleton** **extends** **Enum**

{

**private** **Singleton**(String s, **int** i)

{

**super**(s, i);

}

**protected** **abstract** **void** **read**();

**protected** **abstract** **void** **write**();

**public** **static** Singleton[] values() {

Singleton asingleton[];

**int** i;

Singleton asingleton1[];

System.arraycopy(asingleton = ENUM$VALUES, 0, asingleton1 = **new** Singleton[i = asingleton.length], 0, i);

**return** asingleton1;

}

**public** **static** Singleton **valueOf**(String s)

{

**return** (Singleton)Enum.valueOf(singleton/Singleton, s);

}

Singleton(String s, **int** i, Singleton singleton)

{

**this**(s, i);

}

**public** **static** **final** Singleton INSTANCE;

**private** **static** **final** Singleton ENUM$VALUES[];

**static**

{

INSTANCE = **new** Singleton("INSTANCE", 0) {

**protected** **void** **read**()

{

System.out.println("read");

}

**protected** **void** **write**()

{

System.out.println("write");

}

};

ENUM$VALUES = (**new** Singleton[] {

INSTANCE

});

}

}

看到了这个类的真身过后，相信很多人对于枚举单例防反射的的原理就了解了：

* 类的修饰**abstract**，所以没法实例化，反射也无能为力。
* 关于线程安全的保证，其实是通过**类加载机制**来保证的，我们看看INSTANCE的实例化时机，是在**static块**中，JVM加载类的过程显然是线程安全的。
* 对于防止反序列化生成新实例的问题还不是很明白，一般的方法我们会在该类中添加上如下方法，不过枚举中也没有显示的写明该方法。

//readResolve to prevent another instance of Singleton

**private** Object **readResolve**(){

**return** INSTANCE;

}

如果写的有问题，欢迎指正~