**引言**

单例模式比较常见的实现方法有懒汉模式，DCL模式公有静态成员等，从**[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "_blank)** 1.5版本起，单元素枚举实现单例模式成为最佳的方法。

**Java枚举**

**基本用法**

枚举的用法比较多，本文主要旨在介绍利用枚举实现单例模式的原理，所以这里也主要介绍一些相关的基础内容。   
首先，枚举类似类，一个枚举可以拥有成员变量，成员方法，构造方法。先来看枚举最基本的用法：

enum Type{

A,B,C,D;

}

创建enum时，编译器会自动为我们生成一个继承自**[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "_blank)**.lang.Enum的类，我们上面的enum可以简单看作：

class Type extends Enum{

public static final Type A;

public static final Type B;

...

}

对于上面的例子，我们可以把Type看作一个类，而把A，B，C，D看作类的Type的实例。   
当然，这个构建实例的过程不是我们做的，一个enum的构造方法限制是private的，也就是不允许我们调用。

**“类”方法和“实例”方法**

上面说到，我们可以把Type看作一个类，而把A，B。。。看作Type的一个实例。同样，在enum中，我们可以定义类和实例的变量以及方法。看下面的代码：

enum Type{

A,B,C,D;

static int value;

public static int getValue() {

return value;

}

String type;

public String getType() {

return type;

}

}

在原有的基础上，添加了类方法和实例方法。我们把Type看做一个类，那么enum中静态的域和方法，都可以视作类方法。和我们调用普通的静态方法一样，这里调用类方法也是通过 Type.getValue()即可调用，访问类属性也是通过Type.value即可访问。

下面的是实例方法，也就是每个实例才能调用的方法。那么实例是什么呢？没错，就是A，B，C，D。所以我们调用实例方法，也就通过 Type.A.getType()来调用就可以了。

最后，对于某个实例而言，还可以实现自己的实例方法。再看下下面的代码：

enum Type{

A{

public String getType() {

return "I will not tell you";

}

},B,C,D;

static int value;

public static int getValue() {

return value;

}

String type;

public String getType() {

return type;

}

}

这里，A实例后面的{…}就是属于A的实例方法，可以通过覆盖原本的方法，实现属于自己的定制。   
除此之外，我们还可以添加抽象方法在enum中，强制ABCD都实现各自的处理逻辑:

enum Type{

A{

public String getType() {

return "A";

}

},B {

@Override

public String getType() {

return "B";

}

},C {

@Override

public String getType() {

return "C";

}

},D {

@Override

public String getType() {

return "D";

}

};

public abstract String getType();

}

**枚举单例**

有了上面的基础，我们可以来看一下枚举单例的实现方法：

class Resource{

}

public enum SomeThing {

INSTANCE;

private Resource instance;

SomeThing() {

instance = new Resource();

}

public Resource getInstance() {

return instance;

}

}

上面的类Resource是我们要应用单例模式的资源，具体可以表现为网络连接，**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)**连接，线程池等等。   
获取资源的方式很简单，只要 SomeThing.INSTANCE.getInstance() 即可获得所要实例。下面我们来看看单例是如何被保证的：   
首先，在枚举中我们明确了构造方法限制为私有，在我们访问枚举实例时会执行构造方法，同时每个枚举实例都是static final类型的，也就表明只能被实例化一次。在调用构造方法时，我们的单例被实例化。   
也就是说，因为enum中的实例被保证只会被实例化一次，所以我们的INSTANCE也被保证实例化一次。   
可以看到，枚举实现单例还是比较简单的，除此之外我们再来看一下Enum这个类的声明：

public abstract class Enum<E extends Enum<E>>

implements Comparable<E>, Serializable

可以看到，枚举也提供了序列化机制。某些情况，比如我们要通过网络传输一个数据库连接的句柄，会提供很多帮助。   
最后借用 《Effective Java》一书中的话，

单元素的枚举类型已经成为实现Singleton的最佳方法。