反射

1、什么是反射技术？

**动态获取指定类以及类中的内容(成员)，并运行其内容。**

应用程序已经运行，无法在其中进行new对象的建立，就无法使用对象。这时可以根据配置文件的类全名去找对应的字节码文件，并加载进内存，并创建该类对象实例。这就需要使用反射技术完成

2、获取class对象的三种方式

**获取Class对象的方式一：**

通过对象具备的getClass方法(源于Object类的方法)。有点不方便，需要用到该类，并创建该类的对象，再调用getClass方法完成。

Person p = new Person();//创建Peron对象

Class clazz = p.getClass();//通过object继承来的方法(getClass)获取Person对应的字节码文件对象

**获取Class对象的方式二：**

每一个类型都具备一个class静态属性，通过该属性即可获取该类的字节码文件对象。比第一种简单了一些，仅用一个静态属性就搞定了。但是，还是有一点不方便，还必须要使用到该类。

Class clazz = Person.class;

**获取Class对象方式三：**

\* 去找找Class类中是否有提供获取的方法呢？

\* 找到了，static Class forName(className);

\* 相对方便的多，不需要直接使用具体的类，只要知道该类的名字即可。

\* 而名字完成可以作为参数进行传递 ,这样就可以提高扩展性。

\* 所以为了动态获取一个类，第三种方式最为常用。

Class clazz = Class.forName("cn.itcast.bean.Person");//必须类全名

创建Person对象的方式

以前：1，先加载cn.itcast.bean.Person类进内存。

2,将该类封装成Class对象。

3,根据Class对象，用new操作符创建cn.itcast.bean.Person对象。

4,调用构造函数对该对象进行初始化。

cn.itcast.bean.Person p = new cn.itcast.bean.Person();

通过方式三：(此外还可以使用构造，构造可以指定参数---如String.class)

String className = "cn.itcast.bean.Person";

//1，根据名称获取其对应的字节码文件对象

1，通过forName()根据指定的类名称去查找对应的字节码文件，并加载进内存。

2，并将该字节码文件封装成了Class对象。

3，直接通过newIntstance方法，完成该对象的创建。

4，newInstance方法调用就是该类中的空参数构造函数完成对象的初始化。

Class clazz = Class.forName(className);

//2，通过Class的方法完成该指定类的对象创建。

Object object = clazz.newInstance();//该方法用的是指定类中默认的空参数构造函数完成的初始化。

清单1，获取字节码文件中的字段。

Class clazz = Class.*forName*("cn.itcast.bean.Person");

//获取该类中的指定字段。比如age

Field field = clazz.getDeclaredField("age");//clazz.getField("age"); //为了对该字段进行操作，必须要先有指定类的对象。

Object obj = clazz.newInstance();

//对私有访问，必须取消对其的访问控制检查，使用AccessibleObject父类中的setAccessible的方法

field.setAccessible(**true**);//暴力访问。建议大家尽量不要访问私有

field.set(obj, 789);

//获取该字段的值。

Object o = field.get(obj);

System.*out*.println(o);

备注：getDeclaredField：获取所有属性，包括私有。

getField:获取公开属性，包括从父类继承过来的，不包括非公开方法。

清单2，获取字节码文件中的方法。

//根据名称获取其对应的字节码文件对象

Class clazz = Class.*forName*("cn.itcast.bean.Person");

//调用字节码文件对象的方法getMethod获取class对象所表示的类的公共成员方法(指定方法)，参数为方法名和当前方法的参数，无需创建对象，它是静态方法

Method method = clazz.getMethod("staticShow", **null**);

//调用class对象所表示的类的公共成员方法，需要指定对象和方法中的参数列表

method.invoke(**null**, **null**);

………………………………………………………………………………………………………

Class clazz = Class.*forName*("cn.itcast.bean.Person");

//获取指定方法。

Method method = clazz.getMethod("publicShow", **null**);

//获取指定的类对象。

Object obj = clazz.newInstance();

method.invoke(obj, **null**);//对哪个对象调用方法，是参数组

好处：大大的提高了程序的扩展性。