**Java 反射 Class类**

@author ixenos

 摘要：Class类在反射中的地位、构造Class对象的三种方式、Class对象构造对应类型对象的三种方式

**Class类在反射中的地位**

 位于java.lang下的Class类是位于java.lang.reflect下的Constructor类、Method类、Field类和Array类的入口类

-Class类　　代表一个类

-Field类　　代表类的成员变量（也成为类的属性、域）

-Array类　　提供了动态创建数组、以及访问数组元素的静态方法

-Method类　　代表类的方法

-Constructor类　　代表类的构造方法

**构造Class对象的三种方式**

**0.前言：**

为了**使用类**而做的**准备工作**包括三个步骤：

1）**加载**：**类加载器**查找字节码（一般在classpath中找），从字节码创建一个Class对象

2）**链接**：验证字节码，为**静态域**（只是static修饰的域，不包含static final ）分配存储空间，**解析**此类对其他类的所有**引用**

3）**初始化**：若该类有超类，对其初始化，执行**静态初始化器**（构造器算一个）和**静态初始化块**。

-----------------------------

\*\*\***static final** int = 47 是**编译期常量，不需要对类进行初始化就可以读取**

\*\*\***static final** int = **Random.nextInt(100)**  是运行时常量，这种一般要在对象创建后才会运行，超过初始化的阶段了！

\*\*\***static** int = 47 是非常数的静态域，不是常量，更不是编译期常量，链接阶段只分配存储空间，初始化阶段才初始化

**1.Class类的forName静态方法（自动初始化）**

**只知道对应类型名时，**使用Class.forName(String name) 动态生成Class<String>对象，name是完全限定名

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifAPI文档描述

**2.Object类对象自带getClass方法（已经初始化）**

**通过已经存在的对象，**即持有对应类型对象的引用时**，**使用对象的getClass()（属于根类Object的一部分），如new Integer(1).getClass()将返回Integer.class，而此时的类型对象也必定是在运行中了，所以**已经初始化**

**3.使用类字面常量（不会自动初始化）**

例如 Fancy.class、String.class、Integer.TYPE

1）类字面常量应用于：普通类（包含包装类哟）、接口、数组、基本数据类型

\*\*\*基本数据类型 使用标准字段TYPE，这是个指向基本数据类型的Class对象的引用

\*\*\*例如int.class等价于Integer.TYPE，但是不等价于Integer.class

2）使用".class"来创建Class对象的引用时，**不会自动地初始化该Class对象。**

那**为什么**不会自动初始化呢？由补充内容可知初始化被延迟到了对静态(static)方法（构造器等同于隐式静态）或者非常数(final)静态域进行**首次引用时**才执行，而**引用类字面常量在运行时只是到了加载和链接的阶段**

**Class对象构造对应类型对象的三种方式**

**无参构造**

**1.Class对象的newInstance方法（不能传递参数）**

先获得Class对象，然后通过Class对象的newInstance方法直接生成

1 Class<?> classType = String.class;

2 Object obj = classType.newInstance();

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifAPI文档描述

**2.Contructor对象的newInstance方法（不填参数就和1.一样）**

先获得Class对象，然后通过Class对象生成Constructor对象，通过Constructor对象的newInstance方法生成

1 Class<?> classType = Customer.class;

2 //Class[]数组对应形参类型、个数，这里为空

3 Constructor cons = classType.getConstructor(new Class[]{});

4 //Object[]数组对应传入的实参的类型、个数，这里为空

5 Object obj = cons.newInstance(new Object[]{});

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifAPI-getConstructor

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifAPI-Cons-newInstance

**含参构造**

**3.Contructor对象的newInstance方法（填入形参类型和实际参数）**

先获得Class对象，然后通过Class对象生成Constructor对象，通过Constructor对象的newInstance方法生成

1 Class<?> classType = Customer.class;

2 //Class[]数组对应形参类型、个数，这里为Integer.TYPE,Xenos.class,String.class

3 Constructor cons = classType.getConstructor(new Class[]{Integer.TYPE, Xenos.class, String.class});

4 //Object[]数组对应传入的实参的类型、个数，这里为1, new Xenos(), "Think"

5 Object obj = cons.newInstance(new Object[]{1, new Xenos(), "Think"});