1、什么是反射

2、反射可以获取的内容

3、class的构建

4、反射的方法

编程中直接用到反射的地方很少，不过如果研究过spring、mybatis等框架，会发现反射是这些框架的基础。让我们一块看下反射和它的应用吧。

1、什么是反射

2、class加载有哪些方法

3、通过反射可以获取到类的那些东西

一、反射的定义

反射机制是指在程序的运行状态中，可以构造任意类的对象，了解任意对象所属的类，了解任意类的成员变量和方法，可以调用任意对象的属性和方法。

即，在不了解类和实例化对象的情况下可以进行以下操作：

1、通过类构造实例化对象

2、通过类读取类包含的属性方法

3、通过实例化对象获取类的信息

4、通过实例化对象操作对象的所有属性方法

这么方便的操作，为什么在日常编码中很少直接使用呢，反射有如下问题：

1、性能差，反射是一种解释性操作，直接通知JVM进行的操作，无法进行代码优化，比直接使用类和实例化对象要慢的多。

2、安全问题，通过反射可以执行实例化对象所有的属性方法，包括private的方法，会引起一些安全错误

3、可读维护性差，使用反射相对直接调用，不能直观的看到调用的属性方法，出问题了，代码的可读、调试和维护性会很差

4、抽象性，对象改变了，通过反射逻辑可能无法感知，引起一些莫名的错误

二、Class的获取

反射的第一步需要获取Class对象，怎么获取呢

首先构建一个pojo对象Person作为例子

public class Person {  
 static {  
 System.*out*.println("i am person");  
 }  
 private String name;  
 private int age;  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
  
 public void showMessage()  
 {  
 System.*out*.println("this person message");  
 }  
}

1、实例化对象获取Class对象

Person person = new Person();  
Class cl1 = person.getClass();

2、类获取Class对象

Class cl2 = Person.class;

3、使用class.forName根据类的全名获取

Class cl3 = Class.*forName*("com.mg.empty.demo.mg20200531.Person");

4、使用类加载器获取

Class cl4 = ClassLoader.*getSystemClassLoader*().loadClass("com.mg.empty.demo.mg20200531.Person");

附：forName和loadClass的区别

类的加载过程包括以下步骤：**加载、连接（连接过程不细分，在class相关文章中详述）、初始化、使用、卸载**

forName默认是进行到初始化，loadClass进行到连接这一步

即，loadClass只把类加载到了jvm中，forName加载并完成了初始化

这个怎么来证明一下呢

在Person这个类中有一块代码

static {  
 System.*out*.println("i am person");  
}

这个是在初始化阶段执行的，试下两个方法是否有这个日志打印

如果forName不想走初始化过程，可以使用public static Class<?> forName(String name, boolean initialize, ClassLoader loader) 这个方法加载，将initialize设置为false即可

三、反射的使用

1、构造函数

2、属性

3、方法

4、注解

四、应用场景