# 通过反射APi动态调用 方法,属性,构造器

先获取反射对象

String path = "com.wiceflow.Reflection.pojo.User";

Class<User> clazz = (Class<User>) Class.forName(path);

## 通过反射API调用构造方法,构造对象

User u1 = clazz.newInstance(); // 其实是调用了User的无参构造方法

System.out.println(u1.getUname()); // null

获取构造器

Constructor<User> c = clazz.getDeclaredConstructor(int.class,int.class,String.class);

User u2 = c.newInstance(1001,01,"高琪");

System.out.println(u2.getUname()); // 高琪

## 通过反射API调用方法

User u3 = clazz.newInstance();

// 利用这个方法可以动态的从配置文件,数据库中读取信息动态调用不同方法

Method method = clazz.getDeclaredMethod("setUname", String.class);

method.invoke(u3,"BF");

System.out.println(u3.getUname()); // BF

## 通过反射API操作属性

User u4 = clazz.newInstance();

Field f = clazz.getDeclaredField("uname");

// 私有属性不允许在别的类进行操作

// 添加这句话后 默认关闭安全检查,可以设置私有属性

f.setAccessible(true);

f.set(u4,"BF2");

System.out.println(u4.getUname()); // BF2

# 测试源代码

