**什么是反射机制**

**简单的来说，反射机制指的是程序在运行时能够获取自身的信息。在java中，只要给定类的名字，**

**那么就可以通过反射机制来获得类的所有信息。**

**获取Book类的完整==》类名**

**1.class.forname("类的完整名")**

**2.类名.class.getname(); 获取完整类名，类.class 都是获取了class类**

**3.new Book().getClass().getName()，任何对象.getcalss()都是获取了class类**

**try {**

**//加载**

**System.out.println(Class.forName("cn.Reflex.Book"));**

**System.out.println(Book.class.getName());//获取到完整类名**

**System.out.println(new Book().getClass().getName());//获取到类名**

**} catch (ClassNotFoundException e) {**

**e.printStackTrace();**

**}**

**获取包名，类名和访问修饰符**

**try {**

**Class a = Class.forName("cn.Reflex.Book");//加载**

**System.out.println("Book所在的包"+a.getPackage().getName());**

**System.out.println("Book全类名"+a.getName());**

**//获取类的访问修饰符**

**int modifiers = a.getModifiers();**

**System.out.println("public对应的数字"+modifiers);**

**} catch (ClassNotFoundException e) {**

**e.printStackTrace();**

**}**

**获取类中所有的属性相关信息**

**try {**

**//加载**

**Class a = Class.forName("cn.Reflex.Book");**

**//获取所有的字段使用getDeclaredFields();**

**Field[] fields = a.getDeclaredFields();**

**for (Field s:fields){**

**System.out.println(s);**

**}**

**//获取所有字段的访问修饰符**

**for (Field f:fields){**

**System.out.println("访问修饰符"+Modifier.toString(f.getModifiers()));**

**System.out.println("访问修饰符队应的值"+f.getModifiers());**

**}**

**} catch (ClassNotFoundException e) {**

**e.printStackTrace();**

**}**

**获取类中的所有方法 ，不包含构造方法**

**try {**

**Class a = Class.forName("cn.Reflex.Book");**

**Method[] methods = a.getDeclaredMethods();**

**for (Method f:methods){**

**System.out.println("方法的名字是："+f.getName());**

**System.out.println("方法的修饰数值："+f.getModifiers());**

**System.out.println("方法的修饰符："+Modifier.toString(f.getModifiers()));**

**System.out.println("方法的返回值类型："+f.getReturnType());**

**}**

**} catch (ClassNotFoundException e) {**

**e.printStackTrace();**

**}**

**获取类中所有的构造方法相关信息**

**try {**

**Class a=Class.forName("cn.Reflex.Book");**

**Constructor[] constructors = a.getDeclaredConstructors();**

**for(Constructor c:constructors){**

**System.out.println("构造方法的名称"+c.getName());**

**System.out.println("构造方法的修饰符数值"+c.getModifiers());**

**System.out.println("构造方法的修饰符"+Modifier.toString(c.getModifiers()));**

**System.out.println("构造方法的参数个数是"+c.getParameterCount());**

**}**

**} catch (ClassNotFoundException e) {**

**e.printStackTrace();**

**}**

**获取私有的方法和属性并执行**

**try {**

**Class a = Class.forName("cn.Reflex.Book");//加载**

**try {**

**Book o = (Book) a.newInstance();//实例化**

**try {**

**Field field = a.getDeclaredField("bookname");**

**String name = field.getName();**

**System.out.println("字段名"+name);**

**//打开字段开关**

**field.setAccessible(true);**

**System.out.println("字段的值"+field.get(o));**

**//获取私有的方法**

**try {**

**Method method=a.getDeclaredMethod("getsum",double.class);**

**method.setAccessible(true);**

**try {**

**double d = (double) method.invoke(o, 50.00);**

**System.out.println(d);**

**} catch (InvocationTargetException e) {**

**e.printStackTrace();**

**}**

**} catch (NoSuchMethodException e) {**

**e.printStackTrace();**

**}**

**} catch (NoSuchFieldException e) {**

**e.printStackTrace();**

**}**

**} catch (InstantiationException e) {**

**e.printStackTrace();**

**} catch (IllegalAccessException e) {**

**e.printStackTrace();**

**}**

**} catch (ClassNotFoundException e) {**

**e.printStackTrace();**

**}**