# **Java的反射机制**

### 一、什么是class类

在面向对象的世界里，万物皆对象。类是对象，类是java.lang.Class类的实例对象。另外class类只有java虚拟机才能new出来。任何一个类都是Class 类的实例对象。这实例对象有三种表达方式：

public class User{

}

public class ClassTest{

User u=new User();

//方式1:

Class c1=User.class;//方式2:

Class c2=u.getClass();//方式3:

Class c3=Class.forName("com.forezp.User");

//可以通过类的类型创建该类的实例对象

User user=(User)c1.newInstance();

}

### 二、class类的动态加载

Class.forName(类的全称);该方法不仅表示了类的类型，还代表了动态加载类。编译时刻加载类是静态加载、运行时刻加载类是动态加载类。

### 三、获取方法信息

基本的数据类型，void关键字都Class 类的实例;可以通过get  
ame();getSimpleName()获取类的名称。

Class c1=String.class;

Class c2=int.class;

Class c3=void.class;

System.out.println(c1.getName());

System.out.println(c2.getSimpleName());

获取类的所有方法，并打印出来：

public static void printClassInfo(Object object){

Class c=object.getClass();

System.out.println("类的名称："+c.getName());

/\*\*

\* 一个成员方法就是一个method对象

\* getMethod()所有的 public方法，包括父类继承的 public

\* getDeclaredMethods()获取该类所有的方法，包括private ,但不包括继承的方法。

\*/

Method[] methods=c.getMethods();//获取方法

//获取所以的方法，包括private ,c.getDeclaredMethods();

for(int i=0;i<methods.length;i++){

//得到方法的返回类型

Class returnType=methods[i].getReturnType();

System.out.print(returnType.getName());

//得到方法名：

System.out.print(methods[i].getName()+"(");

Class[] parameterTypes=methods[i].getParameterTypes();

for(Class class1:parameterTypes){

System.out.print(class1.getName()+",");

}

System.out.println(")");

}

}

public class ReflectTest {

public static void main(String[] args){

String s="ss";

ClassUtil.printClassInfo(s);

}

}

运行：

类的名称：java.lang.String

booleanequals(java.lang.Object,)

java.lang.StringtoString()

inthashCode()

...

### 四、获取成员变量的信息

也可以获取类的成员变量信息

public static void printFiledInfo(Object o){

Class c=o.getClass();

/\*\*

\* getFileds()获取public

\* getDeclaredFields()获取所有

\*/

Field[] fileds=c.getDeclaredFields();

for(Field f:fileds){

//获取成员变量的类型

Class filedType=f.getType();

System.out.println(filedType.getName()+" "+f.getName());

}

}

public static void main(String[] args){

String s="ss";

//ClassUtil.printClassInfo(s);

ClassUtil.printFiledInfo(s);

}

运行：

[C value  
int hash  
long serialVersionUID  
[Ljava.io.ObjectStreamField; serialPersistentFields  
java.util.Comparator CASE\_INSENSITIVE\_ORDER  
int HASHING\_SEED  
int hash32

### 五、获取构造函数的信息

public static void printConstructInfo(Object o){

Class c=o.getClass();

Constructor[] constructors=c.getDeclaredConstructors();

for (Constructor con:constructors){

System.out.print(con.getName()+"(");

Class[] typeParas=con.getParameterTypes();

for (Class class1:typeParas){

System.out.print(class1.getName()+" ,");

}

System.out.println(")");

}

}

public static void main(String[] args){

String s="ss";

//ClassUtil.printClassInfo(s);

//ClassUtil.printFiledInfo(s);

ClassUtil.printConstructInfo(s);

}

运行：

java.lang.String([B ,)  
java.lang.String([B ,int ,int ,)  
java.lang.String([B ,java.nio.charset.Charset ,)  
java.lang.String([B ,java.lang.String ,)  
java.lang.String([B ,int ,int ,java.nio.charset.Charset ,)  
java.lang.String(int ,int ,[C ,)  
java.lang.String([C ,boolean ,)  
java.lang.String(java.lang.StringBuilder ,)  
java.lang.String(java.lang.StringBuffer ,)

...

### 六、方法反射的操作

获取一个方法：需要获取方法的名称和方法的参数才能决定一个方法。

方法的反射操作：

method.invoke(对象，参数列表);

举个例子：

class A{

public void add(int a,int b){

System.out.print(a+b);

}

public void toUpper(String a){

System.out.print(a.toUpperCase());

}

}

public static void main(String[] args) {

A a=new A();

Class c=a.getClass();

try {

Method method=c.getMethod("add",new Class[]{int.class,int.class});

//也可以 Method method=c.getMethod("add",int.class,int.class);

//方法的反射操作

method.invoke(a,10,10);

}catch (Exception e){

e.printStackTrace();

}

}

运行：

20