Java反射和内省

一 定义

反射：JAVA反射机制是在运行状态中，对于任意一个类，都能够得到这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法；这种动态获取的信息以及动态调用对象的方法的功能称为java语言的反射机制。

    用一句比较白的话来概括，反射就是让你可以通过名称来得到对象 ( 类，属性，方法 ) 的技术。例如我们可以通过类名来生成一个类的实例；知道了方法名，就可以调用这个方法；知道了属性名就可以访问这个属性的值。

内省：内省(Introspector)是 Java 语言对 Bean 类属性、事件的一种缺省处理方法。例如类 A 中有属性 name, 那我们可以通过 getName,setName 来得到其值或者设置新的值。通过 getName/setName 来访问 name 属性，这就是默认的规则。 Java 中提供了一套 API 用来访问某个属性的 getter/setter 方法。

二 使用方法

生成class对象的方法：

（对象）.getClass();

(类).calss

Class.forname(类的全限定名)

利用类名来构造一个类的实例

Class clazz = Class.forname(类的名字)

Object obj = clazz.newInstance()

反射：

调用有参数的构造函数来获得一个实例

// className类的全限定名，Object[] args构造函数参数

public Object newInstance(String className, Object[] args) throws Exception

{

        //args为参数数组

        Class newoneClass = Class.forName(className);

        //得到参数的Class数组(每个参数的class组成的数组)，使用class反射获得相应的构造函数时传递的参数

        Class[] argsClass = new Class[args.length];

        for (int i = 0, j = args.length; i < j; i++) {

            argsClass[i] = args[i].getClass();

        }

// 选择的相应的构造函数

        Constructor cons = newoneClass.getConstructor(argsClass); //根据argsClass选择函数

// 返回指定构造函数建立的对象实例

        return cons.newInstance(args); //根据具体参数实例化对象。

   }

获得对象的属性

// owner指定对象 fieldname获得的某一属性

public Object getProperty(Object owner, String fieldName) throws Exception

{

       // 反射获得Class

Class ownerClass = owner.getClass();//首先得到class

       Field field = ownerClass.getField(fieldName);

       //然后得到这个class具有的field,也可以通过getFields（）得到所有的field

// 此处为获得相应对象的属性，如果要获得类属性，field.get(Class)

       Object property = field.get(owner);

       //owner指出了取得那个实例的这个属性值，

         如果这个属性是非公有的，这里会报IllegalAccessException。

       return property;

   }

执行某对象的方法

// owner指定的对象，methodname方法名字，args方法的参数

public Object invokeMethod(Object owner, String methodName,

                             Object[] args) throws Exception {

     Class ownerClass = owner.getClass(); //也是从class开始的

     //得到参数的class数组，相当于得到参数列表的类型数组，来取决我们选择哪个函数。

     Class[] argsClass = new Class[args.length];

     for (int i = 0, j = args.length; i < j; i++) {

         argsClass[i] = args[i].getClass();

     }

     //类似于构造函数，通过方法名和参数来选择具体是哪一个方法

     Method method = ownerClass.getMethod(methodName, argsClass);

// 如果是类方法，则只需method.invoke(null,args)

     return method.invoke(owner, args);//具体实例下，具体参数值下调用此函数

  }

内省：

直接通过属性的描述器java.beans.PropertyDescriptor类，来访问属性的getter/setter 方法

// 定义一个javabean对象

Object obj = new Object();

// 获得属性描述器

       PropertyDescriptor pd = new PropertyDescriptor(propertyName,Class);     //声明属性描述对象，一次只可描述一个属性

       // 反射获得setter和getter方法

Method m = pd.getWriterMethod();//获取setter方法

       m.invoke(obj,value);

       Method m = pd.getReaderMethod();//获取getter方法

       Object value = m.invoke(obj);

通过类 Introspector 来获取某个对象的 BeanInfo 信息，然后通过 BeanInfo 来获取属性的描述器（ PropertyDescriptor ），通过这个属性描述器就可以获取某个属性对应的 getter/setter 方法，然后我们就可以通过反射机制来调用这些方法。

// getter方法

// pb为javabean对象，从pb对象到他的最上层继承对象

BeanInfo bi=Introspector.getBeanInfo(pb.getClass(), Object.class);

      //将每个属性的信息封装到一个PropertyDescriptor形成一个数组

        其中包括属性名字，读写方法，属性的类型等等

      PropertyDescriptor[] pd=bi.getPropertyDescriptors();

      //演示如何get

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

         if(pd[i].getPropertyType().isArray())  //getPropertyType得到属性类型。

         {

            //getReadMethod()得到此属性的get方法----Method对象，然后用invoke调用这个方法

            String[] result=(String[]) pd[i].getReadMethod().invoke(pb, null);

            System.out.println(pd[i].getName()+":");//getName得到属性名字

            for (int j = 0; j < result.length; j++) {

               System.out.println(result[j]);

            }

         }

         else

         {

            System.out.println(pd[i].getName()+":"+pd[i].getReadMethod().invoke(pb, null));

         }

      }

Setter方法

PersonBean pb0=new PersonBean();

      //模拟一个数据（数据名字和javabean的属性名一致）

      String name="luonan";

      String[] childname=new String[]{"luo","nan"};

      BeanInfo bi0=Introspector.getBeanInfo(pb0.getClass(), Object.class);

      PropertyDescriptor[] pd0=bi0.getPropertyDescriptors();

      for (int i = 0; i < pd0.length; i++) {

         if(pd0[i].getPropertyType().isArray())

         {

            if(pd0[i].getName().equals("childname"));

            {

               if(pd0[i].getPropertyType().getComponentType().equals(String.class))

               {//getComponentType()可以得到数组类型的元素类型

                  //getWriteMethod()得到此属性的set方法---Method对象，然后用invoke调用这个方法

                  pd0[i].getWriteMethod().invoke(pb0,new Object[]{childname});

               }

            }

         }

         else

         {

            if(pd0[i].getName().equals("name"));

            {

               pd0[i].getWriteMethod().invoke(pb0,name);

            }

         }

      }

三、注意事项

当方法中存在基本类型的数据，比如int时，在反射获得无论是方法还是构造函数的时候不能传递Integer类型的，而是传递Integer.type类型的

较好的网址 <http://weimeigudian.iteye.com/blog/806371>

较好的网址 <http://my.oschina.net/mlongbo/blog/77138>

发现一本好书 java经典实例