Java 内省(Introspector)深入理解

Java 内省(Introspector)深入理解

一些概念：

内省(Introspector) 是Java 语言对 JavaBean 类属性、事件的一种缺省处理方法。

JavaBean是一种特殊的类，主要用于传递数据信息，这种类中的方法主要用于访问私有的字段，且方法名符合某种命名规则。如果在两个模块之间传递信息，可以将信息封装进JavaBean中，这种对象称为“值对象”(Value Object)，或“VO”。方法比较少。这些信息储存在类的私有变量中，通过set()、get()获得。

例如类UserInfo：

```
package com.peidasoft.Introspector;

public class UserInfo {

  private long userId;
  private String userName;
  private int age;
  private String emailAddress;

  public long getUserId() {
    return userId;
  }
  public void setUserId(long userId) {
    this.userId = userId;
  }
  public String getUserName() {
    return userName;
  }
  public void setUserName(String userName) {
    this.userName = userName;
  }
  public int getAge() {
    return age;
  }
  public void setAge(int age) {
    this.age = age;
  }
  public String getEmailAddress() {
    return emailAddress;
  }
  public void setEmailAddress(String emailAddress) {
    this.emailAddress = emailAddress;
  }

}
```

在类UserInfo中有属性 userName, 那我们可以通过 getUserName,setUserName来得到其值或者设置新的值。通过 getUserName/setUserName来访问 userName属性，这就是默认的规则。 Java JDK中提供了一套 API 用来访问某个属性的 getter/setter 方法，这就是内省。

JDK内省类库：

PropertyDescriptor类：

PropertyDescriptor类表示JavaBean类通过存储器导出一个属性。主要方法：

1. getPropertyType()，获得属性的Class对象；
2. getReadMethod()，获得用于读取属性值的方法；getWriteMethod()，获得用于写入属性值的方法；
3. hashCode()，获取对象的哈希值；
4. setReadMethod(Method readMethod)，设置用于读取属性值的方法；
5. setWriteMethod(Method writeMethod)，设置用于写入属性值的方法。

实例代码如下：

```
package com.peidasoft.Introspector;

import java.beans.BeanInfo;
import java.beans.Introspector;
import java.beans.PropertyDescriptor;
import java.lang.reflect.Method;

public class BeanInfoUtil {

  public static void setProperty(UserInfo userInfo,String userName)throws Exception{
    PropertyDescriptor propDesc=new PropertyDescriptor(userName,UserInfo.class);
    Method methodSetUserName=propDesc.getWriteMethod();
    methodSetUserName.invoke(userInfo, "wong");
    System.out.println("set userName:"+userInfo.getUserName());
  }

  public static void getProperty(UserInfo userInfo,String userName)throws Exception{
    PropertyDescriptor proDescriptor =new PropertyDescriptor(userName,UserInfo.class);
    Method methodGetUserName=proDescriptor.getReadMethod();
    Object objUserName=methodGetUserName.invoke(userInfo);
    System.out.println("get userName:"+objUserName.toString());
  }
}
```

Introspector类：

将JavaBean中的属性封装起来进行操作。在程序把一个类当做JavaBean来看，就是调用Introspector.getBeanInfo()方法，得到的BeanInfo对象封装了把这个类当做JavaBean看的结果信息，即属性的信息。

getPropertyDescriptors()，获得属性的描述，可以采用遍历BeanInfo的方法，来查找、设置类的属性。具体代码如下：

```
package com.peidasoft.Introspector;

import java.beans.BeanInfo;
import java.beans.Introspector;
import java.beans.PropertyDescriptor;
import java.lang.reflect.Method;


public class BeanInfoUtil {

  public static void setPropertyByIntrospector(UserInfo userInfo,String userName)throws Exception{
    BeanInfo beanInfo=Introspector.getBeanInfo(UserInfo.class);
    PropertyDescriptor[] proDescrtptors=beanInfo.getPropertyDescriptors();
    if(proDescrtptors!=null&&proDescrtptors.length>0){
      for(PropertyDescriptor propDesc:proDescrtptors){
        if(propDesc.getName().equals(userName)){
          Method methodSetUserName=propDesc.getWriteMethod();
          methodSetUserName.invoke(userInfo, "alan");
          System.out.println("set userName:"+userInfo.getUserName());
          break;
        }
      }
    }
  }

  public static void getPropertyByIntrospector(UserInfo userInfo,String userName)throws Exception{
    BeanInfo beanInfo=Introspector.getBeanInfo(UserInfo.class);
    PropertyDescriptor[] proDescrtptors=beanInfo.getPropertyDescriptors();
    if(proDescrtptors!=null&&proDescrtptors.length>0){
      for(PropertyDescriptor propDesc:proDescrtptors){
        if(propDesc.getName().equals(userName)){
          Method methodGetUserName=propDesc.getReadMethod();
          Object objUserName=methodGetUserName.invoke(userInfo);
          System.out.println("get userName:"+objUserName.toString());
          break;
        }
      }
    }
  }
```

3. Java 内省(Introspector)深入理解(
4. 使用jquery的ajax方法获取下拉列表
5. 作为一名合格的JAVA程序员需要点赏
? (5300)

推荐排行榜

1. java内部类作用(3)
2. 使用jquery的ajax方法获取下拉列表
3. 作为一名合格的JAVA程序员需要点赏
? (1)
4. 阿里巴巴java开发手册阅读笔记(1)

```
40
41 }
```

通过这两个类的比较可以看出，都是需要获得PropertyDescriptor，只是方式不一样：前者通过创建对象直接获得，后者需要遍历，所以使用PropertyDescriptor类更加方便。

使用实例：

```
1  package com.peidasoft.Introspector;
2
3  public class BeanInfoTest {
4
5    /**
6     * @param args
7     */
8    public static void main(String[] args) {
9      UserInfo userInfo=new UserInfo();
10     userInfo.setUserName("peida");
11     try {
12       BeanInfoUtil.getProperty(userInfo, "userName");
13
14       BeanInfoUtil.setProperty(userInfo, "userName");
15
16       BeanInfoUtil.getProperty(userInfo, "userName");
17
18       BeanInfoUtil.setPropertyByIntrospector(userInfo, "userName");
19
20       BeanInfoUtil.getPropertyByIntrospector(userInfo, "userName");
21
22       BeanInfoUtil.setProperty(userInfo, "age");
23
24     } catch (Exception e) {
25       // TODO Auto-generated catch block
26       e.printStackTrace();
27     }
28
29   }
30
31 }
```

输出：

```
1  get userName:peida
2  set userName:wong
3  get userName:wong
4  set userName:alan
5  get userName:alan
6  java.lang.IllegalArgumentException: argument type mismatch
7    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
8    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
9    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
10   at java.lang.reflect.Method.invoke(Method.java:597)
11   at com.peidasoft.Introspector.BeanInfoUtil.setProperty(BeanInfoUtil.java:14)
12   at com.peidasoft.Introspector.BeanInfoTest.main(BeanInfoTest.java:22)
```

说明：BeanInfoUtil.setProperty(userInfo, "age");报错是应为age属性是int数据类型，而setProperty方法里面默认给age属性赋的值是String类型。所以会爆出argument type mismatch参数类型不匹配的错误信息。

BeanUtils工具包：

由上述可看出，内省操作非常的繁琐，所以所以Apache开发了一套简单、易用的API来操作Bean的属性——BeanUtils工具包。

BeanUtils工具包：下载：http://commons.apache.org/beanutils/ 注意：应用的时候还需要一个logging包 http://commons.apache.org/logging/

使用BeanUtils工具包完成上面的测试代码：

```
1  package com.peidasoft.Beanutil;
2
3  import java.lang.reflect.InvocationTargetException;
4
5  import org.apache.commons.beanutils.BeanUtils;
6  import org.apache.commons.beanutils.PropertyUtils;
7
8  import com.peidasoft.Introspector.UserInfo;
```

```
9
10 public class BeanUtilTest {
11   public static void main(String[] args) {
12     UserInfo userInfo=new UserInfo();
13      try {
14       BeanUtils.setProperty(userInfo, "userName", "peida");
15
16       System.out.println("set userName:"+userInfo.getUserName());
17
18       System.out.println("get userName:"+BeanUtils.getProperty(userInfo, "userName"));
19
20       BeanUtils.setProperty(userInfo, "age", 18);
21       System.out.println("set age:"+userInfo.getAge());
22
23       System.out.println("get age:"+BeanUtils.getProperty(userInfo, "age"));
24
25       System.out.println("get userName type:"+BeanUtils.getProperty(userInfo,
26 "userName").getClass().getName());
27       System.out.println("get age type:"+BeanUtils.getProperty(userInfo, "age").getClass().getName());
28
29       PropertyUtils.setProperty(userInfo, "age", 8);
30       System.out.println(PropertyUtils.getProperty(userInfo, "age"));
31
32       System.out.println(PropertyUtils.getProperty(userInfo, "age").getClass().getName());
33
34       PropertyUtils.setProperty(userInfo, "age", "8");
35     }
36      catch (IllegalAccessException e) {
37       e.printStackTrace();
38     }
39      catch (InvocationTargetException e) {
40       e.printStackTrace();
41     }
42     catch (NoSuchMethodException e) {
43       e.printStackTrace();
44     }
45   }
}
```

运行结果：

```
set userName:peida
get userName:peida
set age:18
get age:18
get userName type:java.lang.String
get age type:java.lang.String
8
java.lang.Integer
Exception in thread "main" java.lang.IllegalArgumentException: Cannot invoke
com.peidasoft.Introspector.UserInfo.setAge
on bean class 'class com.peidasoft.Introspector.UserInfo' - argument type mismatch - had objects of type
"java.lang.String"
but expected signature "int"
  at org.apache.commons.beanutils.PropertyUtilsBean.invokeMethod(PropertyUtilsBean.java:2235)
  at org.apache.commons.beanutils.PropertyUtilsBean.setSimpleProperty(PropertyUtilsBean.java:2151)
  at org.apache.commons.beanutils.PropertyUtilsBean.setNestedProperty(PropertyUtilsBean.java:1957)
  at org.apache.commons.beanutils.PropertyUtilsBean.setProperty(PropertyUtilsBean.java:2064)
  at org.apache.commons.beanutils.PropertyUtils.setProperty(PropertyUtils.java:858)
  at com.peidasoft.orm.Beanutil.BeanUtilTest.main(BeanUtilTest.java:38)
Caused by: java.lang.IllegalArgumentException: argument type mismatch
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
  at java.lang.reflect.Method.invoke(Method.java:597)
  at org.apache.commons.beanutils.PropertyUtilsBean.invokeMethod(PropertyUtilsBean.java:2170)
  ... 5 more
```

说明：

1.获得属性的值，例如，BeanUtils.getProperty(userInfo,"userName")，返回字符串

2.设置属性的值，例如，BeanUtils.setProperty(userInfo,"age",8)，参数是字符串或基本类型自动包装。设置属性的值是字符串，获得的值也是字符串，不是基本类型。 3.BeanUtils的特点：

1). 对基本数据类型的属性的操作：在WEB开发、使用中，录入和显示时，值会被转换成字符串，但底层运算用的是基本类型，这些类型转到动作由BeanUtils自动完成。

2）. 对引用数据类型的属性的操作：首先在类中必须有对象，不能是null，例如，private Date birthday=new Date();。操作的是对象的属性而不是整个对象，例如，BeanUtils.setProperty(userInfo,"birthday.time",111111);

```
package com.peidasoft.Introspector;
import java.util.Date;

public class UserInfo {

  private Date birthday = new Date();

  public void setBirthday(Date birthday) {
    this.birthday = birthday;
  }
  public Date getBirthday() {
    return birthday;
  }
}
```

```
package com.peidasoft.Beanutil;

import java.lang.reflect.InvocationTargetException;
import org.apache.commons.beanutils.BeanUtils;
import com.peidasoft.Introspector.UserInfo;

public class BeanUtilTest {
  public static void main(String[] args) {
    UserInfo userInfo=new UserInfo();
     try {
      BeanUtils.setProperty(userInfo, "birthday.time","111111");
      Object obj = BeanUtils.getProperty(userInfo, "birthday.time");
      System.out.println(obj);
    }
     catch (IllegalAccessException e) {
      e.printStackTrace();
    }
     catch (InvocationTargetException e) {
      e.printStackTrace();
    }
    catch (NoSuchMethodException e) {
      e.printStackTrace();
    }
  }
}
```

3.PropertyUtils类和BeanUtils不同在于，运行getProperty、setProperty操作时，没有类型转换，使用属性的原有类型或者包装类。由于age属性的数据类型是int，所以方法PropertyUtils.setProperty(userInfo, "age", "8")会爆出数据类型不匹配，无法将值赋给属性。

分类: Java

« 上一篇： Digester解析xml文件
» 下一篇： HttpClient学习整理

相关博文：
· JAVA中反射机制五（JavaBean的内省与BeanUtils库）
· 深入理解Java：内省(Introspector)
· java自省机制(Introspector)
· Java内省IntroSpector应用
· 内省、JavaBean、PropertyDescriptor类、Introspector类、BeanUtils工具包、注解、Rentention、Tar...
» 更多推荐...

最新 IT 新闻：
· 微信在印度正式下线！印度手机号强制退出
· 代谢规律告诉你，四年的宿便不是没可能
· 微软Windows预览新选项 让部分客户管理诊断数据
· UC回应被印度前雇员起诉：致力于服务印度市场和保护本地员工的福利
· 吴忌寒卸任比特大陆全资子公司监事，詹克团为该公司法定代表人
» 更多新闻...