## JAVA反射機制

JAVA反射機制是在執行狀態中，對於任意一個類，都能夠知道這個類的所有屬性和方法；對於任意一個物件，都能夠呼叫它的任意一個方法；這種動態獲取的資訊以及動態呼叫物件的方法的功能稱為java語言的反射機制。

Java反射機制主要提供了以下功能： 在執行時判斷任意一個物件所屬的類；在執行時構造任意一個類的物件；在執行時判斷任意一個類所具有的成員變數和方法；在執行時呼叫任意一個物件的方法；生成動態代理。

### 1. 得到某個物件的屬性

複製程式碼 程式碼如下:

```
public Object getProperty(Object
owner, String fieldName) throws
Exception {
    Class ownerClass =
```

```
owner.getClass();

    Field field =
ownerClass.getField(fieldName);

    Object property = field.get(owner);

    return property;
}
```

Class ownerClass = owner.getClass()：得到該物件的Class。

Field field = ownerClass.getField(fieldName)：通過Class得到類宣告的屬性。

Object property = field.get(owner)：通過物件得到該屬性的例項，如果這個屬性是非公有的，這裡會報IllegalAccessException。

**2. 得到某個類的靜態屬性**

複製程式碼 程式碼如下:

```
public Object getStaticProperty(String
```

```
className, String fieldName)
            throws Exception {
    Class ownerClass =
Class.forName(className);

    Field field =
ownerClass.getField(fieldName);

    Object property =
field.get(ownerClass);

    return property;
}
```

Class ownerClass = Class.forName(className) ：首先得到這個類的Class。

Field field = ownerClass.getField(fieldName)：和上面一樣，通過Class得到類宣告的屬性。

Object property = field.get(ownerClass)：這裡和上面有些不同，因為該屬性是靜態的，所以直接從類的Class裡取。

## 3. 執行某物件的方法

複製程式碼 程式碼如下:

```
public Object invokeMethod(Object
owner, String methodName, Object[]
args) throws Exception {

    Class ownerClass =
owner.getClass();

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

    for (int i = 0, j = args.length; i < j; i )
{
        argsClass[i] = args[i].getClass();
    }

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

    return method.invoke(owner, args);
}
```

Class owner_class = owner.getClass() ：首先還是必須得到這個物件的Class。

5～9行：配置引數的Class陣列，作為尋找Method的條件。

Method method = ownerClass.getMethod(methodName, argsClass)：通過methodName和引數的argsClass（方法中的引數型別集合）陣列得到要執行的Method。

method.invoke(owner, args)：執行該Method.invoke方法的引數是執行這個方法的物件owner，和引數陣列args，可以這麼理解：owner物件中帶有引數args的method方法。返回值是Object，也既是該方法的返回值。

**4. 執行某個類的靜態方法**

複製程式碼 程式碼如下:

```
public Object
invokeStaticMethod(String className,
String methodName,
                Object[] args) throws Exception
```

```
{
    Class ownerClass =
Class.forName(className);

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

    for (int i = 0, j = args.length; i < j; i )
{
        argsClass[i] = args[i].getClass();
    }

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

    return method.invoke(null, args);
 }
```

基本的原理和例項3相同，不同點是最後一行，invoke的一個引數是null，因為這是靜態方法，不需要藉助例項執行。

## 5. 新建例項

複製程式碼 程式碼如下:

```
public Object newInstance(String
className, Object[] args) throws
Exception {
    Class newoneClass =
Class.forName(className);

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

    for (int i = 0, j = args.length; i < j; i )
{
        argsClass[i] = args[i].getClass();
    }

    Constructor cons =
newoneClass.getConstructor(argsClass);

    return cons.newInstance(args);

}
```

這裡說的方法是執行帶引數的建構函式來新建例項的方法。如果不需要引數，可以

直接使用newoneClass.newInstance()來實現。

Class newoneClass = Class.forName(className)：第一步，得到要構造的例項的Class。

第5～第9行：得到引數的Class陣列。

Constructor cons = newoneClass.getConstructor(argsClass)：得到構造子。

cons.newInstance(args)：新建例項。

**6. 判斷是否為某個類的例項**

複製程式碼 程式碼如下:

```
public boolean isInstance(Object obj,
Class cls) {
    return cls.isInstance(obj);
}
```

**7. 得到陣列中的某個元素**

複製程式碼 程式碼如下:

```
public Object getByArray(Object array,
int index) {
```

```
    return Array.get(array,index);
}
```

目錄



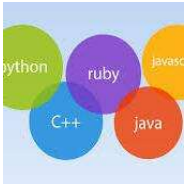
## 您可能感興趣的文章:

Java反射機制的實現詳解Java反射機制的學習總結java 利用反射機制,獲取實體所有屬性和方法,並對屬性賦值Java通過反射機制動態設定物件屬性值的方法利用java反射機制呼叫類的私有方法(推薦)java 利用java反射機制動態載入類的簡單實現通過java反射機制動態呼叫某方法的總結(推薦)java基於執行緒池和反射機制實現定時任務完整例項利用java反射機制實現自動呼叫類的簡單方法Java程式設計反射機制用法入門與例項總結

# 写评论

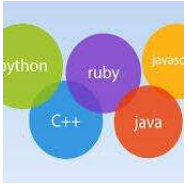
很抱歉，必須登入網站才能發佈留言。

## 近期文章



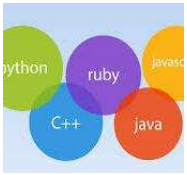

**Vue中容易被忽視的知識點**

2019.12.09



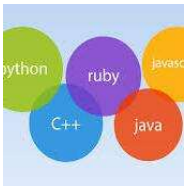

**if我是前端Leader，談談前端框架體系建設**

2019.12.09




## Spark入門（一）用SparkShell初嘗Spark滋味

2019.12.08




## Spark入門（二）如何用Idea運行我們的Spark項目

2019.12.08