# 使用反射创建指定元素类型的List

## 概述思路

明确一下实现效果：使用反射创建指定元素类型的List

所谓指定元素类型的List，**等价于**List中所有元素全都是指定类型。

基于这个实现效果的实现思路：

1. 创建一个可以存放指定元素对象的List
2. 创建一个指定元素对象
3. 调用元素对象方法
4. 把对象存到list中
5. 重复2、3、4

假设创建元素类型为**entity.ChoiceQuestion**类型的List

## 首先定义一个泛型创建List的方法

**private** **static** <T> List<T> createListOfType(Class<T> type, **int** size){

**return** **new** ArrayList<T>(size);

}

## 接下来创建一个可以存放指定类型的List

type = “entity.ChoiceQuestion”;

// 可以这么创建一个只能存放指定类型的List

List resList = *createListOfType*(Class.*forName*(type), results.size());

// 或者这么创建一个可以存放任意类型的List

List resList = *new* ArrayList(results.size());

这里不指定List的类型，List默认是Object。

## 然后开始创建指定类型的对象

// 创建指定题目类型对象

Object tmpQues = Class.*forName*(type).newInstance();

## 再然后调用方法

// 获取方法对象

Method method = tmpQues.getClass()

.getMethod("setQuestion", String.**class**);

// 执行方法

method.invoke(tmpQues, "我是题目");

## 最后录入元素

// 新增元素

resList.add(tmp);

**为了不使反射失去意义**

无论怎么创建的List，都只能用List<Object>接收

无论怎么创建的元素对象，都只能用Object接收

这样就实现了**使用反射创建指定元素类型的List**的效果。

一切都还好，就是调用方法有点麻烦。

## 还有什么别的办法吗？

有个方法可以稍微简化一下方法调用方面的代码

思路：可以给**entity.ChoiceQuestion**类继承一个接口或对象，

假设继承了一个接口**entity.ChoiceQuestionFather**

这个接口有**entity.ChoiceQuestion**的setQuestion方法

// 执行方法于是我们可以用接口对象来接收反射创建的实体类

ChoiceQuestionFather tmpQues = Class.*forName*(type).newInstance();

// 这时候就可以直接使用其中的方法了

tmpQues.setQuestion("我是题目");