**当ASP.NET MVC邂逅jQuery.Ajax提交数组**

2012-12-24 10:00 Zork 博客园 [我要评论(0)](http://developer.51cto.com/art/201212/373975.htm#commment) 字号：**[T](javascript:setfont(12);" \t "_self)** | [**T**](javascript:setfont(16);)

[一键收藏，随时查看，分享好友！](javascript:favorBox('open');)

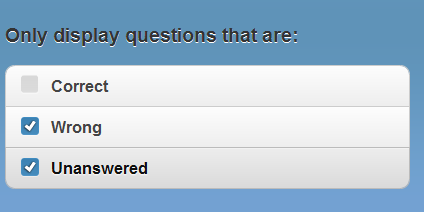
硬编码味道太重，换个类型又得重写，工作量跟之前比还视乎增加了，只是Controller变得优雅了。这种浪费青春又耗电的做法还是不符合要求。

AD：[51CTO 网+ 第十二期沙龙：大话数据之美\_如何用数据驱动用户体验](http://mobile.51cto.com/mobile/mdsa12/" \t "_blank)

当ASP.NET MVC 通过JQuery的Ajax 提交数组时，MVC的model binder机制就失效了。我们不得不在Controller里面编写自定义代码，将Request提交的数据转换成需要的数据类型。这个过程往往枯燥乏味。下面以某项目的实际例子来演示如何解决这个问题，提供一个通用的解决方案。

**需求描述**

当用户更改了配置，需要Ajax提交到服务器。



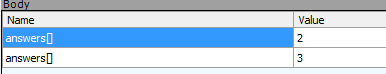
前端代码:

1. **var** items = [];
2. $("input:checked").each(**function** () {
3. items.push($(**this**).val());
4. });
5. $.ajax({
6. type: 'post',
7. url: 'Configure/Status',
8. data: { answers: items }
9. });

后端代码:

1. public enum AnswerStatus
2. {
3. Correct = 1,
4. Incorrect = 2,
5. Unanswered = 3
6. }
7. [HttpPost]
8. public ActionResult Status(IList**<AnswerStatus>** answers)
9. {
10. ….
11. }

这里的answers始终为null. 神器fiddler出场，发现用JQuery.Ajax 提交Array的数据，提交的时候始终会在名称后面加上”[]”， 问题就出在这里。



根据发现的结果修改代码：

1. [HttpPost]
2. public ActionResult Status(IList**<AnswerStatus>** answers)
3. {
4. answers = Request.Form.GetValues(“answers[]”).Select(d =**>** d.ToEnum**<AnswerStatus>**(AnswerStatus.Unanswered).ToList();
5. }

虽然这样能够通过解决我的问题，但每次提交Array都要这样手工解析request，视乎一夜回到石器时代了。其实我们马上会想到MVC 的Mode Binder。

**尝试进行第一次重构：**

1. public class AnswerModelBinder : IModelBinder
2. {
3. public override object BindModel(ControllerContext controllerContext, ModelBindingContext bindingContext)
4. {
5. return controllerContext.RequestContext.HttpContext.Request.Form.GetValues(“answers[]”).Select(d =**>** d.ToEnum**<AnswerStatus>**(AnswerStatus.Unanswered).ToList();
6. }
7. }

硬编码味道太重，换个类型又得重写，工作量跟之前比还视乎增加了，只是Controller变得优雅了。这种浪费青春又耗电的做法还是不符合要求。

**进行第二次重构 ： DefaultModelBinder 出场**

万能的DefaultModelBinder，能够绑定任何类型，可惜就是client传过来的name后面多加了”[]”，导致DefaultModelBinder无法准确解析。那我们能不能欺骗DefaultModelBInder呢？

查看ModelBindingContext发现有一个ModelName属性，感觉有点像要绑定的参数的名称，调试跟踪发现ModelName确实就是参数的名称，那我们修改ModelName让他跟client传过来的name保持一致是否就能够充分发挥DefaultModelBinder。于是动手创建一个JQAjaxModelBinder

并继承自DefaultModelBinder：

1. **public** **override** **object** BindModel(ControllerContext controllerContext, ModelBindingContext bindingContext)
2. {
3. **if**(bindingContext.ModelType.IsEnumerable())
4. {
5. var key = bindingContext.ModelName + "[]";
6. var valueResult = bindingContext.ValueProvider.GetValue(key);
7. **if**(valueResult != **null** && !**string**.IsNullOrEmpty(valueResult.AttemptedValue))
8. {
9. bindingContext.ModelName = key;
10. }
11. }
12. **return** **base**.BindModel(controllerContext, bindingContext);
13. }//如何使用自定义ModelBinder。该方法是Controller里面的Action
14. **public** ActionResult Status([ModelBinder(**typeof**(ModelBinder.JQAjaxModelBinder))] IList<AnswerStatus> answers)
15. {
16. …
17. }

这时，Controller里面的Status （Action）方法已经能够正确得到前端传来的数据。并且还是强类型的。当然很多程序员都是懒惰的，笔者也是这其中一份子。笔者连Parameter前面的参数（[ModelBinder(typeof(ModelBinder.JQAjaxModelBinder))]）都不想写，那我们直接在ModelBinders里面注册吧。其实注册的时候也有点麻烦，必须设定Type，我那能提前知道有那些类型啊。干脆将JQAjaxModelBinder设置成默认的ModerBinder，一劳永逸，再也没有烦心事情了。

ModelBinder不同注册方法

通过在Action方法的参数前面添加ModelBinder标签，上文则是采用的这种方法。

数据类型上面添加ModelBinder标签

1. [ModelBinder(typeof(ModelBinder.JQAjaxModelBinder))]
2. Public class User
3. {
4. }

通过ModelBinders注册

1. ModelBinders.Binders.Add(typeof(User), new ModelBinder.JQAjaxModelBinder());

设置默认的ModerBinder

1. ModelBinders.Binders.DefaultBinder = new ModelBinder.JQAjaxModelBinder();

后记： 当我们在开发的时候，经常做重复的事情，当一件事情重复多次后，我们就需要停下来认真思考，能不能将这些事情抽象出来，做一个通用的解决方案呢？一劳永逸的解决这些问题。

原文链接：<http://www.cnblogs.com/coolite/archive/2012/12/24/JQModelBinder.html>