## **特性路由**

WebApi2默认的路由规则我们称作基于约定路由,很多时候我们使用RESTful风格的URI.简单的路由是没问题的，如 api/Products/{id}，但有些事很难处理的，如资源之间存在嵌套关系：客户包含订单，书有作者属性等等。对于这种Uri，我们希望的路由是这样的：/costomers/{customerid}/orders 或 /costomers/{customerid}/orders/{orderid}

考虑到这只是某个Controller的路由格式，而我们会有很多个Controller，用基于约定路由显然不合适(要配置很多的路由)

使用特性路由就简单了，在action上加一个特性即可

[Route("customers/{customerId}/orders")]public IEnumerable<Order> GetOrdersByCustomer(int customerId) { ... }

通过使用特性路由，我们还可以做API的版本控制

/api/v1/products  
/api/v2/products

启用特性路由需要在配置过程中调用System.Web.HttpConfigurationExtensions类的MapHttpAttributeRoutes方法

[IMG_256](https://www.cnblogs.com/TiestoRay/p/javascript:void(0);)

using System.Web.Http;

namespace WebApplication

{

public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

// Web API routes config.MapHttpAttributeRoutes();

// Other Web API configuration not shown. }

}

}

[IMG_257](https://www.cnblogs.com/TiestoRay/p/javascript:void(0);)

在WebApi1中  项目模板是这样的

protected void Application\_Start()

{

WebApiConfig.Register(GlobalConfiguration.Configuration);

//。。。

}

如果要启用特性路由，需要改成如下代码

protected void Application\_Start()

{

GlobalConfiguration.Configure(WebApiConfig.Register);

//。。。

}

**注**：特性路由和基于约定路由是可以结合使用大的。

HttpMethod

默认情况，WebApi会根据action的方法名前缀查找action(不区分大小写)，比如GetUsers，会匹配Get。通过在action上添加HttpMethod特性，可以覆盖action需要映射的Http Method。

可使用的特性包括：[HttpDelete],[HttpPost],[HttpHead],[HttpOptions],[HttpPatch],[HttpGet],[HttpPut]

通过AcceptVerbs特性，我们还可以指定非标准方法以及多个方法，如 [AcceptVerbs("MKCOL","GET","POST")]

## ****路由前缀****

通常情况下，一个Controller下的action会使用相似的路由模板，如

* [Route("api/books")]
* [Route("api/books/{id:int}")]
* [Route("api/books/{bookid}/authors")]

这时候可以为整个controller指定[RoutePrefix]特性，以使用共同的前缀，把[RoutePrefix("/api/books")]加到controller上，action的路由特性就变成这样：

* [Route("")]
* [Route("{id:int}")]
* [Route("{bookid}/authors")]

此外，路由前缀中也可以包含参数，如[RoutePrefix("api/{userid}/books")]

这里还有两个小技巧

如果有某个特殊路由不希望使用前缀，可以在路由中添加~，如[Route("~api/otherbooks")]

有时候需要几个路由片段结合起作用，如日期 /api/books/date/2013/06/17

这时候就需要使用字符\* ,[Route("date/{\*date:datetime:regex(\\d{4}/\\d{2}/\\d{2})}")],不过这种参数只能用作路由的最后一个参数

## **路由约束**

路由约束让我们可以限制模板参数的匹配方式。一般的语法是 "{参数：约束类型}"：

[Route("users/{id:int}"]public User GetUserById(int id) { ... }

[Route("users/{name}"]public User GetUserByName(string name) { ... }

如果参数int，则选中第一个GetUserById，否则是GetUserByName。(跟方法定义的顺序无关)

下面的表格列出了支持的约束

| **约束** | **介绍** | **示例** |
| --- | --- | --- |
| alpha | 匹配大写或小写字母 (a-z, A-Z) | {x:alpha} |
| bool |  | {x:bool} |
| datetime |  | {x:datetime} |
| decimal |  | {x:decimal} |
| double |  | {x:double} |
| float | 匹配一个 32位浮点数 | {x:float} |
| guid |  | {x:guid} |
| int |  | {x:int} |
| length | 匹配一个长度在指定范围内的字符串 | {x:length(6)} {x:length(1,20)} |
| long |  | {x:long} |
| max | 匹配指定了最大值的整数 | {x:max(10)} |
| maxlength | 匹配指定了最大长度字符串 | {x:maxlength(10)} |
| min | 匹配指定了最小值的整数 | {x:min(10)} |
| minlength | 匹配指定了最小长度字符串 | {x:minlength(10)} |
| range | 匹配指定了大小区间的整数 | {x:range(10,50)} |
| regex | 匹配一个正则表达式 | {x:regex(^\d{3}-\d{3}-\d{4}$)} |

 如果要指定多个约束，需要用冒号间隔 [Route("users/{id:int:min(1)}")]

通过实现IHttpRouteConstraint接口，还可以创建自定义路由约束。（不过一般正则就可以搞定了）

还可以通过实现IInlineConstraintResolver接口替换整个DefaultInlineConstraintResolver类。这样做将取代所有的内置的约束，除非实现IInlineConstraintResolver的类将它们添加进去。

IMG_258 自定义路由约束Demo

## **可选URI参数，默认值**

 通过在参数约束后面添加一个问号，可以设定URI参数是可选的；也可以像普通方法那样指定默认值：

[Route("api/books/locale/{lcid:int?}")]public IEnumerable<Book> GetBooksByLocale(int lcid = 1033) { ... }

[Route("api/books/locale/{lcid:int=1033}")]public IEnumerable<Book> GetBooksByLocale(int lcid) { ... }

 这两者是等价的

## **路由名称**

WebApi中，每一个路由都有一个名字，用于生成链接，并在放入Http响应中。（应该是用于重定向吧）

例如对某个action A指定Name，[Route("api/books/{id}", Name="GetBookById")]

那么其他action B在需要返回这个action A的链接时，就可以这样使用

[IMG_259](https://www.cnblogs.com/TiestoRay/p/javascript:void(0);)

public HttpResponseMessage Post(Book book)

{

var response = Request.CreateResponse(HttpStatusCode.Created);

string uri = Url.Link("GetBookById", new { id = book.BookId });

response.Headers.Location = new Uri(uri);

return response;

}

[IMG_260](https://www.cnblogs.com/TiestoRay/p/javascript:void(0);)

路由顺序

通过设定特性[Route("xxx",RouteOrder=n)]可以指定路由的查找顺序

[Route("pending", RouteOrder = 1)]public HttpResponseMessage GetPending() { ... }

不过意义不大，通过顺序来控制，还不如设定更好的路由来的实际，而且不至于让开发人员觉得混乱。

https://www.cnblogs.com/TiestoRay/p/5755454.html