## **从 ControllerBase 派生类**

从控制器（旨在用作 Web API）中的 [ControllerBase](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.controllerbase) 类继承。 例如:

[Produces("application/json")]

[Route("api/[controller]")]

public class PetsController : ControllerBase

{

private readonly PetsRepository \_repository;

public PetsController(PetsRepository repository)

{

\_repository = repository;

}

[HttpGet("{id}")]

[ProducesResponseType(404)]

public async Task<ActionResult<Pet>> GetByIdAsync(int id)

{

var pet = await \_repository.GetPetAsync(id);

if (pet == null) {

return NotFound();

}

return pet;

}

[HttpPost]

[ProducesResponseType(400)]

public async Task<ActionResult<Pet>> CreateAsync(Pet pet)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

await \_repository.AddPetAsync(pet);

return CreatedAtAction(nameof(GetByIdAsync),

new { id = pet.Id }, pet);

}

}

通过 ControllerBase 类可使用一些属性和方法。 在前面的代码中，示例包括 [BadRequest(ModelStateDictionary)](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.controllerbase.badrequest" \l "Microsoft_AspNetCore_Mvc_ControllerBase_BadRequest_Microsoft_AspNetCore_Mvc_ModelBinding_ModelStateDictionary_) 和 [CreatedAtAction(String, Object, Object)](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.controllerbase.createdataction" \l "Microsoft_AspNetCore_Mvc_ControllerBase_CreatedAtAction_System_String_System_Object_System_Object_)。

## **使用 ApiController 属性进行注释**

ASP.NET Core 2.1 引入了 [[ApiController]](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.apicontrollerattribute) 特性，用于批注 Web API 控制器类。 例如:

[Route("api/[controller]")]

[ApiController]

public class ProductsController : ControllerBase

为在控制器级别使用此特性，需要兼容性版本 2.1 或更高版本

Startup.ConfigureServices 中突出显示的代码设置了 2.1 兼容性标志：services.AddMvc() .SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

在 ASP.NET Core 2.2 或更高版本中，可将 [ApiController] 特性应用于程序集。 以这种方式进行注释，会将 web API 行为应用到程序集中的所有控制器。

[assembly: ApiController]

namespace WebApiSample.Api.\_22

{

public class Startup

{

[ApiController] 特性通常结合 ControllerBase 来为控制器启用特定于 REST 行为。

另一种方法是创建使用 [ApiController] 特性进行批注的自定义基本控制器类：

[ApiController]

public class MyBaseController

{

}

以下各部分说明该特性添加的便利功能。

### **自动 HTTP 400 响应**

模型验证错误会自动触发 HTTP 400 响应。

使用 [InvalidModelStateResponseFactory](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.apibehavioroptions.invalidmodelstateresponsefactory" \l "Microsoft_AspNetCore_Mvc_ApiBehaviorOptions_InvalidModelStateResponseFactory) 自定义所生成的响应的输出。

services.AddMvc()

.SetCompatibilityVersion(CompatibilityVersion.Version\_2\_2)

.ConfigureApiBehaviorOptions(options =>

{

options.SuppressConsumesConstraintForFormFileParameters = true;

options.SuppressInferBindingSourcesForParameters = true;

options.SuppressModelStateInvalidFilter = true;

options.SuppressMapClientErrors = true;

options.ClientErrorMapping[404].Link =

"https://httpstatuses.com/404";

});

使用 2.2 或更高版本的兼容性标志时，HTTP 400 响应的默认响应类型为 [ValidationProblemDetails](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.validationproblemdetails)。 将 SuppressUseValidationProblemDetailsForInvalidModelStateResponses 属性设置为 true 以返回 ASP.NET Core 2.1 错误格式 [SerializableError](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.serializableerror)。

services.AddMvc()

.SetCompatibilityVersion(CompatibilityVersion.Version\_2\_2)

.ConfigureApiBehaviorOptions(options =>

{

options

.SuppressUseValidationProblemDetailsForInvalidModelStateResponses = true;

});

### **绑定源参数**

绑定源特性定义可找到操作参数值的位置。 存在以下绑定源特性：

| **特性** | **绑定源** |
| --- | --- |
| **[[FromBody]](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.frombodyattribute)** | 请求正文 |
| **[[FromForm]](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.fromformattribute)** | 请求正文中的表单数据 |
| **[[FromHeader]](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.fromheaderattribute)** | 请求标头 |
| **[[FromQuery]](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.fromqueryattribute)** | 请求查询字符串参数 |
| **[[FromRoute]](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.fromrouteattribute)** | 当前请求中的路由数据 |
| **[[FromServices]](https://docs.microsoft.com/zh-cn/aspnet/core/mvc/controllers/dependency-injection?view=aspnetcore-2.2" \l "action-injection-with-fromservices)** | 作为操作参数插入的请求服务 |

**警告**

当值可能包含 %2f（即 /）时，请勿使用 [FromRoute]。 %2f 不会转换为 /（非转移形式）。 如果值可能包含 %2f，则使用 [FromQuery]。

没有 [ApiController] 特性时，将显式定义绑定源特性。

[HttpGet]

public async Task<ActionResult<List<Product>>> GetAsync( [FromQuery] bool discontinuedOnly = false)

{

List<Product> products = null;

if (discontinuedOnly)

{

products = await \_repository.GetDiscontinuedProductsAsync();

}

else

{

products = await \_repository.GetProductsAsync();

}

return products;

}

### **Multipart/form-data 请求推理**

使用 [[FromForm]](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.fromformattribute) 特性批注操作参数时，将推断 multipart/form-data 请求内容类型。

当 [SuppressConsumesConstraintForFormFileParameters](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.apibehavioroptions.suppressconsumesconstraintforformfileparameters" \l "Microsoft_AspNetCore_Mvc_ApiBehaviorOptions_SuppressConsumesConstraintForFormFileParameters) 属性设置为 true 时，会禁用默认行为。

在 Startup.ConfigureServices 中添加下列代码：

services.AddMvc()

.SetCompatibilityVersion(CompatibilityVersion.Version\_2\_2)

.ConfigureApiBehaviorOptions(options =>

{

options.SuppressConsumesConstraintForFormFileParameters = true;

options.SuppressInferBindingSourcesForParameters = true;

options.SuppressModelStateInvalidFilter = true;

options.SuppressMapClientErrors = true;

options.ClientErrorMapping[404].Link =

"https://httpstatuses.com/404";

});

### **特性路由要求**

特性路由是必要条件。 例如:

[Route("api/[controller]")]

[ApiController]

public class ProductsController : ControllerBase

不能通过 [UseMvc](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.builder.mvcapplicationbuilderextensions.usemvc) 中定义的[传统路由](https://docs.microsoft.com/zh-cn/aspnet/core/mvc/controllers/routing?view=aspnetcore-2.2" \l "conventional-routing)或通过 Startup.Configure 中的 [UseMvcWithDefaultRoute](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.builder.mvcapplicationbuilderextensions.usemvcwithdefaultroute) 访问操作。

### **错误状态代码的问题详细信息响应**

在 ASP.NET Core 2.2 或更高版本中，MVC 会将错误结果（状态代码为 400 或更高的结果）转换为状态代码为 [ProblemDetails](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.problemdetails) 的结果。ProblemDetails 为：

* 基于 [RFC 7807 规范](https://tools.ietf.org/html/rfc7807)的类型。
* 用于指定 HTTP 响应中计算机可读的错误详细信息的标准化格式。

NotFound 的 HTTP 响应具有 404 状态代码和 ProblemDetails 正文。 例如:

{

type: "https://tools.ietf.org/html/rfc7231#section-6.5.4",

title: "Not Found",

status: 404,

traceId: "0HLHLV31KRN83:00000001"

}

需具备 2.2 版本或更高版本的兼容性标志，才可使用问题详细信息功能。 当 SuppressMapClientErrors 属性设置为 true 时，会禁用默认行为。 在 Startup.ConfigureServices 中添加下列代码：

services.AddMvc()

.SetCompatibilityVersion(CompatibilityVersion.Version\_2\_2)

.ConfigureApiBehaviorOptions(options =>

{

options.SuppressConsumesConstraintForFormFileParameters = true;

options.SuppressInferBindingSourcesForParameters = true;

options.SuppressModelStateInvalidFilter = true;

options.SuppressMapClientErrors = true;

options.ClientErrorMapping[404].Link =

"https://httpstatuses.com/404";

});

使用 ClientErrorMapping 属性配置 ProblemDetails 响应的内容。 例如，以下代码会更新 404 响应的 type 属性：

services.AddMvc()

.SetCompatibilityVersion(CompatibilityVersion.Version\_2\_2)

.ConfigureApiBehaviorOptions(options =>

{

options.SuppressConsumesConstraintForFormFileParameters = true;

options.SuppressInferBindingSourcesForParameters = true;

options.SuppressModelStateInvalidFilter = true;

options.SuppressMapClientErrors = true;

options.ClientErrorMapping[404].Link = "https://httpstatuses.com/404";

});