

第89期-理解WebAPI控制器与验证机制

2020年2月24日 10:11

ControllerBase 类

不要通过从 Controller 类派生来创建 Web API 控制器。Controller 派生自 ControllerBase，并添加对视图的支持，因此它用于处理 Web 页面，而不是 Web API 请求。此规则有一个例外：如果打算为视图和 Web API 使用相同的控制器，则从 Controller 派生控制器。

```
[ApiController]
[Route("[controller]")]
public class WeatherForecastController : ControllerBase
```

Attribute 特性

Microsoft.AspNetCore.Mvc 命名空间提供可用于配置 Web API 控制器的行为和操作方法的属性。下述示例使用属性来指定受支持的 HTTP 操作谓词和所有可返回的已知 HTTP 状态代码：

```
[HttpPost]
[ProducesResponseType(StatusCodes.Status201Created)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
public ActionResult<Pet> Create(Pet pet)
{
    pet.Id = _petsInMemoryStore.Any() ?
        _petsInMemoryStore.Max(p => p.Id) + 1 : 1;
    _petsInMemoryStore.Add(pet);

    return CreatedAtAction(nameof(GetById), new { id = pet.Id }, pet);
}
```

ApiController 特性

属性路由要求，自动 HTTP 400 响应，绑定源参数推理，Multipart/form-data 请求推理和错误状态代码的问题详细信息。

加在指定的控制器上

```
[ApiController]
[Route("[controller]")]
```

```
public class WeatherForecastController : ControllerBase
```

多个控制器复用

```
[ApiController]
public class MyControllerBase : ControllerBase
{
}

[Route("[controller]")]
public class PetsController : MyControllerBase
```

加载程序集上影响全部控制器

```
[assembly: ApiController]
namespace WebApiSample
{
    public class Startup
    {
        ...
    }
}
```

特性路由要求

```
[ApiController]
[Route("[controller]")]
public class WeatherForecastController : ControllerBase
```

不能通过 UseEndpoints、UseMvc、UseMvcWithDefaultRoute 配置 API 路由。

自动 HTTP 400 响应

[ApiController] 属性使模型验证错误自动触发 HTTP 400 响应。因此，操作方法中不需要以下代码：

```
if (!ModelState.IsValid)
{
    return BadRequest(ModelState);
}
```

ASP.NET Core MVC 使用 **ModelStateInvalidFilter** 操作筛选器来执行上述检查。

默认 BadRequest 响应

使用 2.1 的兼容性版本时，HTTP 400 响应的默认响应类型为 SerializableError。

```
{
  "": [
    "A non-empty request body is required."
  ]
}
```

使用 2.2 或更高版本的兼容性版本时，响应类型为 ValidationProblemDetails。

```
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
  "title": "One or more validation errors occurred.",
  "status": 400,
  "traceId": "|7fb5e16a-4c8f23bbfc974667.",
  "errors": {
    "": [
      "A non-empty request body is required."
    ]
  }
}
```

ValidationProblemDetails 类型：

- 1、提供计算机可读的格式来指定 Web API 响应中的错误。
- 2、符合 RFC 7807 规范。

禁用自动 400 响应

若要禁用自动 400 行为，请将 `SuppressModelStateInvalidFilter` 属性设置为 `true`。

```
services.AddControllers()
    .ConfigureApiBehaviorOptions(options =>
    {
        options.SuppressModelStateInvalidFilter = true;
    });
```

错误状态代码的问题详细信息

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 的结果。 ProblemDetails 类型基于 RFC 7807 规范，用于提供 HTTP 响应中计算机可读的错误详细信息。

```
if (pet == null)
{
    return NotFound();
}
```

```
}
```

```
{  
  type: "https://tools.ietf.org/html/rfc7231#section-6.5.4",  
  title: "Not Found",  
  status: 404,  
  traceId: "OHLHLV31KRN83:00000001"  
}
```

禁用 ProblemDetails 响应

```
services.AddControllers()  
    .ConfigureApiBehaviorOptions(options =>  
    {  
        options.SuppressMapClientErrors = true;  
        options.ClientErrorMapping[404].Link =  
            "https://httpstatuses.com/404";  
    });
```