Patterns and Practices of Effective API Development in ASP.NET Core

David Berry

@DavidCBerry73

https://github.com/DavidCBerry13/FoodTruckNationApi

Why Getting APIs Right Matters

APIs power today's connected applications

APIs serve as an integration point

Danger: Exampleware

Goals

❖ An API that is easy for users to consume

❖ A codebase that is easy the maintain

❖ A pragmatic implementation that meets our business goals

"Think of this talk as a discussion rather than a hard and fast set of rules"

Principles

API Design Guidelines

Design APIs from the consumer point of view

Think about how many individual API calls it takes to complete a business operation

Try to keep each API focused on one logical business domain

Be consistent. Within your API. With established practices

Be informative when something goes wrong

Resource URLs

`GET /api/FoodTrucks`
    Gets a list of all of the food trucks

`GET /api/FoodTrucks/10`
    Gets the food truck with the id number of 10

`GET /api/FoodTrucks?tag=tacos`
    Gets a list of food trucks that have the tag "tacos"

`GET /api/FoodTrucks/10/Reviews`
    Gets a list of the reviews for food truck with the id number of 10

HTTP Verbs

	GET	**POST**	**PUT**	**DELETE**
`/api/foodtrucks`	Gets all food trucks	Create a food truck	Error	Error
`/api/foodtrucks/10`	Get food truck 10	Error	Update food truck 10	Update food truck 10

Resource URL Guidelines

Favor plural nouns for resource names

Avoid URLs more than two layers deep

Use query parameters to your advantage

HTTP Status Codes

Response Code	Name	Notes
200	OK	Most frequently used status code
201	Created	Used when a resource is crated with a PUT request. The URI of the new resource should be included in a Location header field
304	Not Modified	Used for GET requests when the resource has not changed and caching is enabled
400	Bad Request	The request contained invalid data. The client can fix the errors in the request and try again
401	Unauthorized	Used when the user has not authenticated
403	Forbidden	Used when the user has authenticated but does not have permission to perform the requested action
404	Not Found	The specified resource could not be found
409	Conflict	The resource you are trying to add or update is in conflict with another resource. Resolve the conflict and try again
500	Internal Server Error	Something went wrong on the server. Usually a coding error in the API itself

HTTP Status Code 404

/api/foodtrucks/123

- Makes sense to return HTTP 404 when food truck 123 does not exist
- Everyone understands and expects this behavior

/api/foodtrucks or /api/foodtrucks?tags=Tacos

- I personally prefer to return a status of HTTP 200 and an empty collection in these cases
- Think it is a little easier to develop against
- Avoids confusion of no results found vs incorrect URL

Some Good Coding Principles

Keep separate concerns separate

Aim for a balanced distribution of responsibilities

Code should easily reveal its intentions about how it works
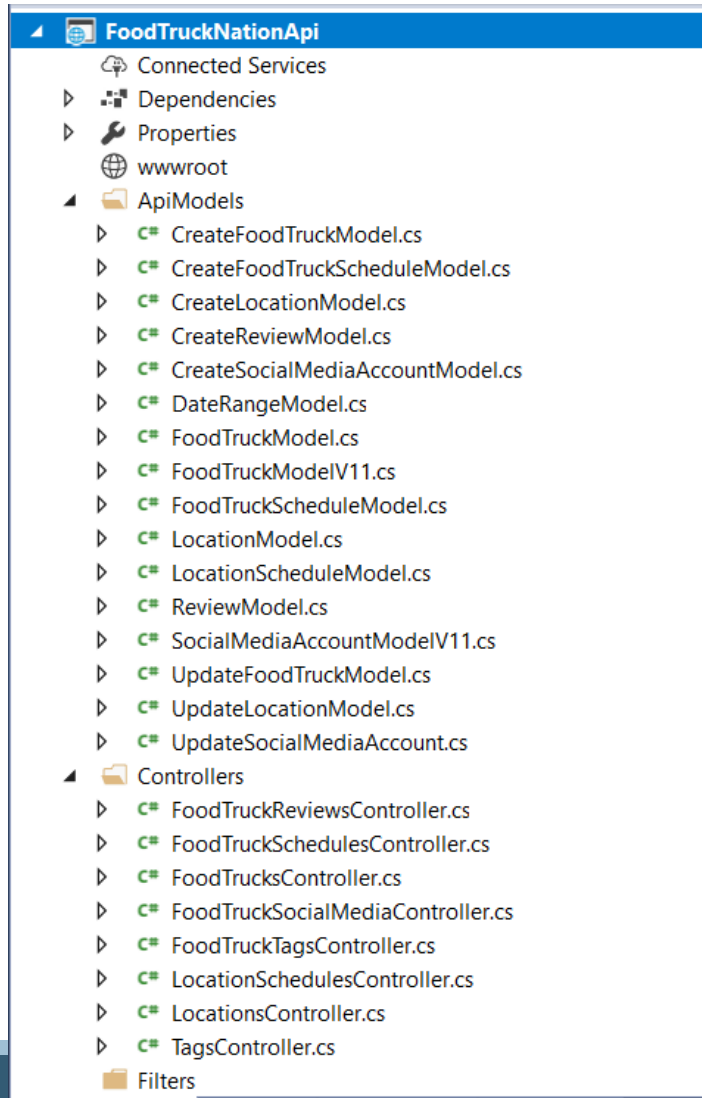
Keep code simple and concise

Practices

Use Feature Folders to Organize Code

Everything related to a feature is in one place

Keeps features independent and testable

Traditional/Standard Folder Layout

```
▲ 🌐 FoodTruckNationApi
    ☁ Connected Services
  ▷ ⚙ Dependencies
  ▷ 🔧 Properties
    🌐 wwwroot
  ▲ 📁 ApiModels
    ▷ C# CreateFoodTruckModel.cs
    ▷ C# CreateFoodTruckScheduleModel.cs
    ▷ C# CreateLocationModel.cs
    ▷ C# CreateReviewModel.cs
    ▷ C# CreateSocialMediaAccountModel.cs
    ▷ C# DateRangeModel.cs
    ▷ C# FoodTruckModel.cs
    ▷ C# FoodTruckModelV11.cs
    ▷ C# FoodTruckScheduleModel.cs
    ▷ C# LocationModel.cs
    ▷ C# LocationScheduleModel.cs
    ▷ C# ReviewModel.cs
    ▷ C# SocialMediaAccountModelV11.cs
    ▷ C# UpdateFoodTruckModel.cs
    ▷ C# UpdateLocationModel.cs
    ▷ C# UpdateSocialMediaAccount.cs
  ▲ 📁 Controllers
    ▷ C# FoodTruckReviewsController.cs
    ▷ C# FoodTruckSchedulesController.cs
    ▷ C# FoodTrucksController.cs
    ▷ C# FoodTruckSocialMediaController.cs
    ▷ C# FoodTruckTagsController.cs
    ▷ C# LocationSchedulesController.cs
    ▷ C# LocationsController.cs
    ▷ C# TagsController.cs
    📁 Filters
```

❖ Too many files in a single folder

❖ Code for any one "feature" is spread across multiple folders

❖ Hard to tell what goes with what

Use Multiple Projects to Organize Your Solution

Keep the API project focused on the API work

Extract common functionality into a shared project

Sample Solution Organization
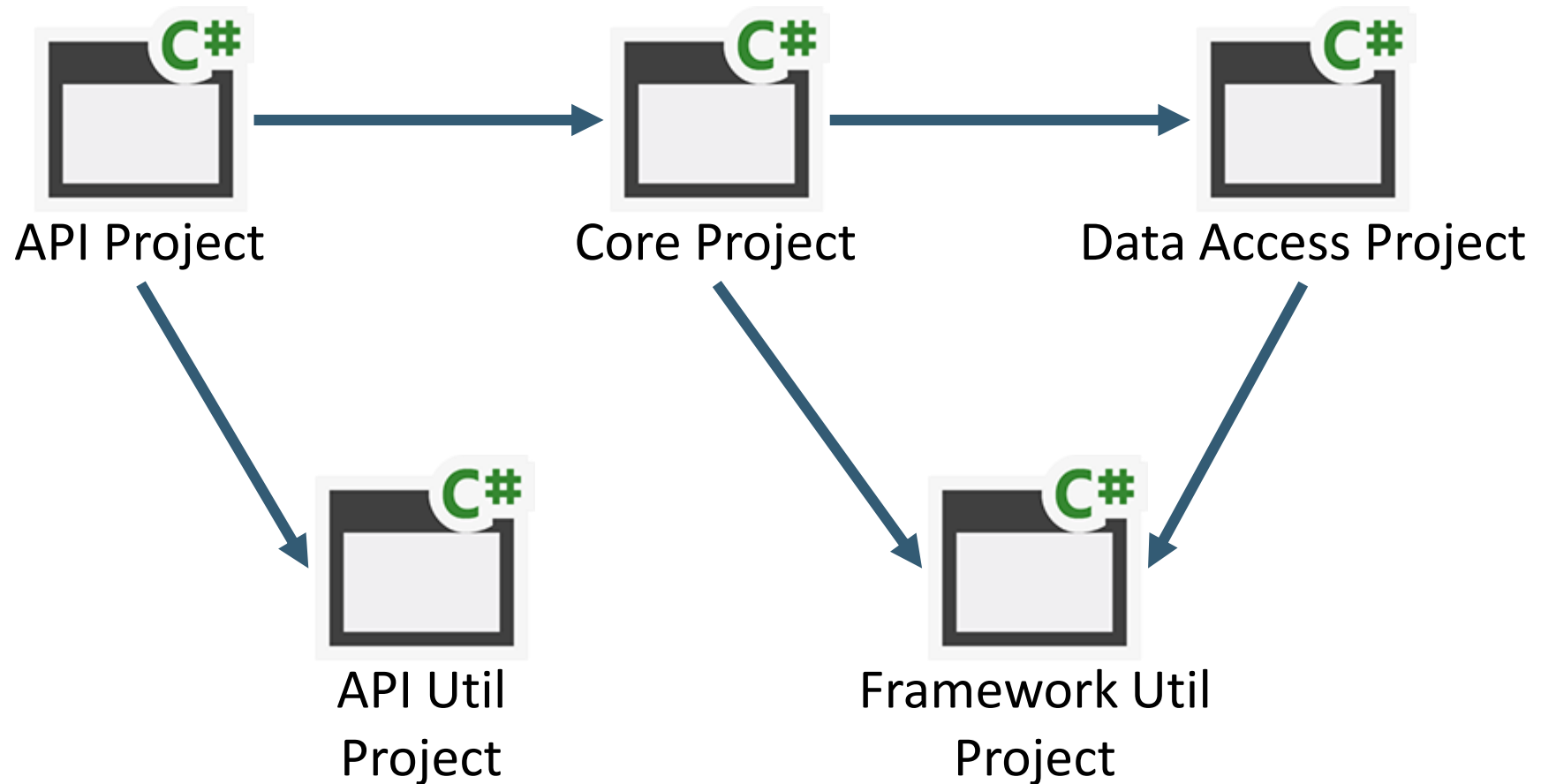
Create a Base Controller for all Controllers to Inherit From

Provides a place to implement common functionality

Important if you want to implement HATEOS

Return IActionResult From Your Action Methods

Enables use of controller helper methods to return the correct HTTP status

Allows you to return custom error objects for non-HTTP 200 responses

Default Action Return Type

```csharp
public FoodTruck Get(int id)
{
    FoodTruck foodTruck = this.foodTruckService.GetFoodTruck(id);

    if (foodTruck == null)
        return null;

    var model = this.mapper.Map<FoodTruck, FoodTruckModel>(foodTruck);
    return model;
}
```

Using IActionResult

```csharp
public IActionResult Get(int id)
{
    FoodTruck foodTruck = this.foodTruckService.GetFoodTruck(id);

    if ( foodTruck == null)
    {
        return this.NotFound(new ApiMessageModel()
            { Message = $"No food truck found with id {id}"} );
    }
    else
    {
        var model = this.mapper.Map<FoodTruck, FoodTruckModel>(foodTruck);
        return this.Ok(model);
    }
}
```

Controller Action Result Helper Methods

Built into ASP.NET Core

Ok()	HTTP 200
Crated()	HTTP 201
CreatedAtAction()	HTTP 201
CreatedAtRoute()	HTTP 201
Accepted()	HTTP 202
AcceptedAtAction()	HTTP 202
AcceptedAtRoute()	HTTP 203
NoContent()	HTTP 204
BadRequest()	HTTP 400
NotFound()	HTTP 404

Not Included in ASP.NET Core

Conflict()	HTTP 409
InternalServerError()	HTTP 500

Use View Models

Allows you to control the shape of the data sent to and from your API

Clearly communicates what is expected in POST and PUT operations

Food Truck View Model

```json
{
    "foodTruckId": 1,
    "name": "American Burger",
    "description": "Classic American Burgers, Hot Dogs and Fries",
    "website": "http://foodtrucknation/BurgerTruck",
    "tags": [
        "Burgers",
        "Hot Dogs",
        "Fries“
    ],
    "socialMediaAccounts": [
        {
            "platformName": "Facebook",
            "accountName": "demo-AmericanBurger“
        },
        {
            "platformName": "Twitter",
            "accountName": "@DemoAmericanBurger“
        }
    ],
    "meta": {
        "self": "http://localhost:8000/api/FoodTrucks/1",
        "reviews": "http://localhost:8000/api/FoodTrucks/1/Reviews",
        "schedules": http://localhost:8000/api/FoodTrucks/1/Schedules
    }
}
```

Create Food Truck View Model

```json
{
    "name": "That's a Wrap!",
    "description": "Your favorite sandwiches served as a wrap",
    "website": "http://thatsawrapmke.com",
    "tags": [
        "Sandwiches",
        "Gluten Free",
        "Vegetarian"
    ],
    "socialMediaAccounts": [
        {
            "socialMediaPlatformId": 2,
            "accountName": "thatsawrapmke"
        }
    ]
}
```

Name Your Routes

Allows you to use ASP.NET helper methods that build URLs based on route names

Makes it easier to implement HATEOS

Using a Route Name With CreateAtRoute

```csharp
public const String GET_FOOD_TRUCK_BY_ID = "GetFoodTruckById";

[HttpGet("{id:int}", Name = GET_FOOD_TRUCK_BY_ID)]
public IActionResult Get(int id)
{
    // Method implemented here…
}


public IActionResult Post([FromBody]CreateFoodTruckModel createModel)
{
    var createCommand = this.mapper.Map<CreateFoodTruckModel,
        CreateFoodTruckCommand>(createModel);

    FoodTruck foodTruck = this.foodTruckService.CreateFoodTruck(createCommand);

    var model = this.mapper.Map<FoodTruck, FoodTruckModel>(foodTruck);
    return this.CreatedAtRoute(GET_FOOD_TRUCK_BY_ID,
        new { id = model.FoodTruckId }, model);
}
```

Returning an HTTP 201 Created

Response Headers

access-control-allow-origin →*
api-supported-versions →1.0, 1.1
content-type →application/json; charset=utf-8
date →Thu, 14 Sep 2017 02:57:16 GMT
location →http://localhost:8000/api/FoodTrucks/16
server →Kestrel
transfer-encoding →chunked
x-powered-by →ASP.NET

⬆

Response headers should include a location header giving the location of the newly created resource

Response Body

```
{
    "foodTruckId": 16,
    "name": "Sushi Truck",
    "description": "Fresh sushi on wheels",
    "website": "http://www.sushifoodtruck.com",
    "tags": [
        "Sushi",
        "Japanese",
        "Asian"
    ],
    "meta": {
        "self": "http://localhost:8000/api/FoodTrucks/16",
        "reviews":
            "http://localhost:8000/api/FoodTrucks/16/Reviews",
        "schedules":
            "http://localhost:8000/api/FoodTrucks/16/Schedules"
    }
}
```

Leverage Query Objects

Enables multiple parameters to be encapsulated into one object

Allows you to make use of validation frameworks for query parameters

Create Custom Error Model Objects

Allows you to return richer error information than a simple message

Helps create a consistent error experience across the application

Make Use of Action Filters

Handle cross cutting concerns in one place

→ Validation

→ Error Handling

Can be applied globally to make sure something always gets done

Create cleaner code by factoring out common functionality

```csharp
public IActionResult Post([FromBody]CreateFoodTruckModel createModel)
{

    if ( !ModelState.IsValid)
    {

        var errors = ModelState
            .Where(e => e.Value.Errors.Count > 0)
            .Select(e => new RequestErrorModel() { Field = e.Key, Message = e.Value.Errors.First().ErrorMessage })
            .ToArray();
    }


    try
    {

        var createCommand = this.mapper.Map<CreateFoodTruckModel, CreateFoodTruckCommand>(createModel);

        FoodTruck foodTruck = this.foodTruckService.CreateFoodTruck(createCommand);

        var model = this.mapper.Map<FoodTruck, FoodTruckModel>(foodTruck);
        return this.CreatedAtRoute(GET_FOOD_TRUCK_BY_ID, new { id = model.FoodTruckId }, model);
    }
    catch (ResourceAlreadyExistsException ex)
    {

        var message = new ApiMessageModel() { Message = ex.Message };
        return this.Conflict(message);
    }
    catch (Exception ex)
    {

        var message = new ApiMessageModel() { Message = ex.Message };
        return this.InternalServerError(message);
    }
}
```

```csharp
public IActionResult Post([FromBody]CreateFoodTruckModel createModel)
{
    var createCommand = this.mapper.Map<CreateFoodTruckModel,
        CreateFoodTruckCommand>(createModel);

    FoodTruck foodTruck = this.foodTruckService.CreateFoodTruck(createCommand);

    var model = this.mapper.Map<FoodTruck, FoodTruckModel>(foodTruck);
    return this.CreatedAtRoute(GET_FOOD_TRUCK_BY_ID,
        new { id = model.FoodTruckId }, model);
}
```

Document Your API With Swagger

Developers have to know what to call and how to call it

Provides a simple way to test out your interface

Important if you want to import your API into an API gateway

Swashbuckle

Swashbuckle Configuration

```csharp
public void Configure(IApplicationBuilder app, IHostingEnvironment env,
    ILoggerFactory loggerFactory)
{
    // Code removed for brevity ...

    app.UseSwagger(c =>
    {
        c.PreSerializeFilters.Add((swagger, httpReq) => swagger.Host = httpReq.Host.Value);
    });

    app.UseSwaggerUI(c =>
    {
        c.SwaggerEndpoint("/swagger/v1.0/swagger.json", "Food Truck API v1.0");
        c.SwaggerEndpoint("/swagger/v1.1/swagger.json", "Food Truck API v1.1");
    });

}
```

Swashbuckle Configuration Part 2

```csharp
public void ConfigureServices(IServiceCollection services)
{
    var pathToXmlComments = Configuration["Swagger:FileName"];
    services.AddSwaggerGen(options =>
    {
        options.SwaggerDoc("v1.1", new Info { Title = "Food Truck API v1.1", Version = "v1.1" });
        options.SwaggerDoc("v1.0", new Info { Title = "Food Truck API v1.0", Version = "v1.0" });
        options.DocInclusionPredicate((docName, apiDesc) =>
        {
            var actionApiVersionModel = apiDesc.ActionDescriptor?.GetApiVersion();
            if (actionApiVersionModel == null)
                return true;

            if (actionApiVersionModel.DeclaredApiVersions.Any())
                return actionApiVersionModel.DeclaredApiVersions.Any(v => $"v{v.ToString()}" == docName);

            return actionApiVersionModel.ImplementedApiVersions.Any(v => $"v{v.ToString()}" == docName);
        });

        var filePath = Path.Combine(PlatformServices.Default.Application.ApplicationBasePath, pathToXmlComments);
        options.IncludeXmlComments(filePath);
        options.DescribeAllEnumsAsStrings();

}
```

Have a Versioning Strategy

Every API will change over time

ASP.NET Core provides some low friction ways to handle versioning

Versioning Package

Microsoft.AspNetCore.Mvc.Versioning 2.0.0

A service API versioning library for Microsoft ASP.NET Core.

Requires NuGet 2.5 or higher.

Package Manager | .NET CLI | Paket CLI

```
PM> Install-Package Microsoft.AspNetCore.Mvc.Versioning -Version 2.0.0
```

Release Notes

https://github.com/Microsoft/aspnet-api-versioning/releases/tag/v2.0.0

Info

🕓 last updated a month ago

🌐 Project Site

📄 License Info

✉ Contact owners

🚩 Report

☁ Manual download

Statistics

↓ 259,129 total downloads

Versioning Configuration

```csharp
public void ConfigureServices(IServiceCollection services)
{
    // Code removed for brevity ...

    services.AddApiVersioning(cfg =>
    {
        cfg.DefaultApiVersion = new Microsoft.AspNetCore.Mvc.ApiVersion(1, 0);
        cfg.AssumeDefaultVersionWhenUnspecified = true;
        cfg.ReportApiVersions = true;
        cfg.ApiVersionReader = new HeaderApiVersionReader("api-version");
    });

    // Code removed for brevity ...
}
```

```csharp
[Route("api/FoodTrucks")]
[ApiVersion("1.0")]
[ApiVersion("1.1")]
public class FoodTrucksController : BaseController
{
    [HttpGet(Name = GET_ALL_FOOD_TRUCKS)]
    [MapToApiVersion("1.0")]
    public IActionResult Get([FromQuery]String tag = null)
    { // Method Body would be here… }

    [HttpGet(Name = GET_ALL_FOOD_TRUCKS)]
    [MapToApiVersion("1.1")]
    public IActionResult GetV11([FromQuery]String tag = null)
    { // Method Body would be here… }

    [HttpDelete("{id}", Name ="DeleteFoodTruck")]
    public IActionResult Delete(int id)
    { // Method Body would be here… }

}
```

Validation is Really, Really Important

One of your first lines of defense for security issues

Fail fast – saves debugging time

Unit test your validations to make sure they work

A Better Validation Library

FluentValidation 7.2.0

A validation library for .NET that uses a fluent interface to construct strongly-typed validation rules.

Package Manager | .NET CLI | Paket CLI

PM> Install-Package FluentValidation -Version 7.2.0

Release Notes

Changes in 7.2:
* Updated AspNetCore integration to support IValidatableObject
* Updated AspNetCore integration to allow mixing multiple validation strategies in the same model
* Updated AspNetCore integration to support RulesetForClientsideMessagesAttribute
* Fix - Property names were not generated properly for nested types when using AddFailure inside a Custom validator
* Fix - NullReferenceException when using CustomAsync but the validator is invoked synchronously
* Fix - Clientside integration in AspNetCore being resolved from the root-level service provider
* Fix - Allow empty string to be passed to OverridePropertyName

Info

🕓 last updated 3 days ago

🌐 Project Site

🔖 License Info

✉ Contact owners

🏳 Report

☁ Manual download

Statistics

↓ 3,206,741 total downloads

🎁 989 downloads of latest version

📈 1,300 downloads per day (avg)

View full stats

HATEOS

Part of the REST specification

Provides links to related resources and actions

Some controversy over if you really need to do it

HATEOS

```json
{
    "foodTruckId": 1,
    "name": "American Burger",
    "description": "Classic American Burgers, Hot Dogs and Fries",
    "website": "http://foodtrucknation/BurgerTruck",
    "tags": [
        "Burgers",
        "Hot Dogs"
     ],
    "meta": {
        "self": "http://localhost:8000/api/FoodTrucks/1",
        "reviews": http://localhost:8000/api/FoodTrucks/1/Reviews",
        "schedules": "http://localhost:8000/api/FoodTrucks/1/Schedules"
    }
}
```

HATEOS Mapping

```csharp
this.CreateMap<FoodTruck, FoodTruckLinks>()
    .ForMember(
        dest => dest.Self,
        opt => opt.ResolveUsing<UrlResolver, RouteUrlInfo>(src =>
            new RouteUrlInfo()
            {
                RouteName = FoodTrucksController.GET_FOOD_TRUCK_BY_ID,
                RouteParams = new { id = src.FoodTruckId }
            }
        )
    )
    .ForMember(
        dest => dest.Reviews,
        opt => opt.ResolveUsing<UrlResolver, RouteUrlInfo>(src =>
            new RouteUrlInfo()
            {
                RouteName = Reviews.FoodTruckReviewsController.GET_ALL_FOOD_TRUCK_REVIEWS,
                RouteParams = new { foodTruckId = src.FoodTruckId }
            }
        )
    );
```

```csharp
this.CreateMap<FoodTruck, FoodTruckLinks>()
    .ForMember(
        dest => dest.Self,
        opt => opt.ResolveUsing<UrlResolver, RouteUrlInfo>(src =>
            new RouteUrlInfo()
            {
                RouteName = FoodTrucksController.GET_FOOD_TRUCK_BY_ID,
                RouteParams = new { id = src.FoodTruckId }
            }
        )
    )
    .ForMember(
        dest => dest.Reviews,
        opt => opt.ResolveUsing<UrlResolver, RouteUrlInfo>(src =>
            new RouteUrlInfo()
            {
                RouteName = Reviews.FoodTruckReviewsController.GET_ALL_FOOD_TRUCK_REVIEWS,
                RouteParams = new { foodTruckId = src.FoodTruckId }
            }
        )
    );
```

Resources

REST Design Resources

https://pages.apigee.com/rs/apigee/images/api-design-ebook-2012-03.pdf

https://apigee.com/about/blog/developer/web-api-design-missing-link

ASP.NET Core APIs

Implementing and Securing an API with ASP.NET Core

By Shawn Wildermuth

https://www.pluralsight.com/courses/aspdotnetcore-implementing-securing-api

Thank You

David Berry

@DavidCBerry73

https://github.com/DavidCBerry13/FoodTruckNationApi