**HRM CODE TUTORIAL COLLECTION**

**Collection By:** R&D Team

Index

[CONFIGURE SWAGGER API 2](#_Toc23858531)

[Features 2](#_Toc23858532)

[Update Note 2](#_Toc23858533)

[Access to the Code 2](#_Toc23858535)

[How to use 3](#_Toc23858536)

[REQUEST POST LARGE DATA JSON – WEB MVC 5](#_Toc23858538)

[Features 5](#_Toc23858539)

[Resolve Exception 5](#_Toc23858540)

[Access to the Code 5](#_Toc23858541)

[How to use 5](#_Toc23858542)

[Example 6](#_Toc23858543)

[Access to the Code 19](#_Toc23858558)

[How to use 19](#_Toc23858559)

[Example 20](#_Toc23858560)

# CONFIGURE SWAGGER API

## Features

Cấu hình Swagger API.

Có 3 Component chính của SwashBuckle

* Swashbuckle.AspNetCore.Swagger: Một Swagger Object Model và Middleware để hiển thị SwaggerDocument object dưới dạng Json endpoints.
* Swashbuckle.AspNetCore.SwaggerGen: Là Swagger generator có nhiệm vụ build SwaggerDocument object trực tiếp từ routes, controllers, và models. Được kết hợp từ Swagger endpoint middleware để tự động phát sinh Swagger JSON
* Swashbuckle.AspNetCore.SwaggerUI: là một phiên bản nhúng của Swagger UI Tool tạo nên giao diện thể hiện API

## Update Note

* 28/12/2020: Create docs

## Access to the Code

##### Server Side

* ProjectName → Startup.cs
  + class Startup
    - ConfigureServices
    - Configure

## How to use

##### Server Side

###### Cấu hình Swagger Middleware

Thêm Swagger Generator vào Services collection tại Startup.ConfigureServices method

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<TodoContext>(opt =>

opt.UseInMemoryDatabase("TodoList"));

services.AddControllers();

// Register the Swagger generator, defining 1 or more Swagger documents

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "My API", Version = "v1" });

});

}

Tại method Startup.Configure, đăng ký middleware cho phép tạo JSON Document và Swagger UI

public void Configure(IApplicationBuilder app)

{

// Enable middleware to serve generated Swagger as a JSON endpoint.

app.UseSwagger();

// Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),

// specifying the Swagger JSON endpoint.

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");

});

app.UseRouting();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

###### XML Comment

XML Comment có thể được cấu hình theo các bước

Visual Studio

Double click vào Project để sửa file Projectname.csproj và thêm code XML như sau:

<PropertyGroup>

<GenerateDocumentationFile>true</GenerateDocumentationFile>

<NoWarn>$(NoWarn);1591</NoWarn>

</PropertyGroup>

Visual Studio for Mac

Từ Solution Pad, nhấn Control và Click vào Project Name -> Chọn Tools -> Edit File

Thêm XML code

<PropertyGroup>

<GenerateDocumentationFile>true</GenerateDocumentationFile>

<NoWarn>$(NoWarn);1591</NoWarn>

</PropertyGroup>

Visual Studio Code

Thêm XML code vào file .csproj

<PropertyGroup>

<GenerateDocumentationFile>true</GenerateDocumentationFile>

<NoWarn>$(NoWarn);1591</NoWarn>

</PropertyGroup>

Cấu hình Swagger để xử dụng XML file được tạo ở trên

/// NOTE LAST 3 LINES IN THIS SNIPPET

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<TodoContext>(opt =>

opt.UseInMemoryDatabase("TodoList"));

services.AddControllers();

// Register the Swagger generator, defining 1 or more Swagger documents

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Version = "v1",

Title = "ToDo API",

Description = "A simple example ASP.NET Core Web API",

TermsOfService = new Uri("https://example.com/terms"),

Contact = new OpenApiContact

{

Name = "Shayne Boyer",

Email = string.Empty,

Url = new Uri("https://twitter.com/spboyer"),

},

License = new OpenApiLicense

{

Name = "Use under LICX",

Url = new Uri("https://example.com/license"),

}

});

// Set the comments path for the Swagger JSON and UI.

var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";

var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);

c.IncludeXmlComments(xmlPath);

});

}

Trong đoạn code ở trên, Reflection được sử dụng để build một XML file name phù hợp với Web API Project.

AppContext.BaseDirectory property được sử dụng để tạo đường dẫn tới XML file.

Thêm Comment cho API

Nhập 3 gạch chéo /// comments để thêm mô tả cho Swagger UI

/// <summary>

/// Deletes a specific TodoItem.

/// </summary>

/// <param name="id"></param>

[HttpDelete("{id}")]

public IActionResult Delete(long id)

{

var todo = \_context.TodoItems.Find(id);

if (todo == null)

{

return NotFound();

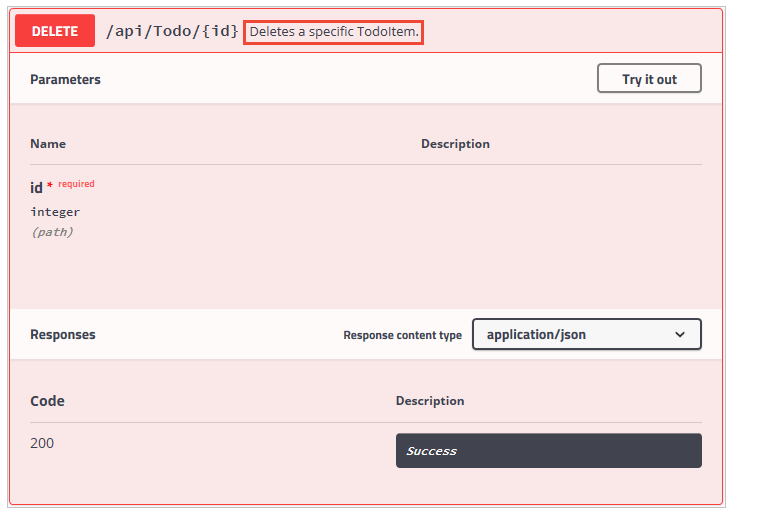
}

\_context.TodoItems.Remove(todo);

\_context.SaveChanges();

return NoContent();

}

Swagger UI sẽ hiển thị phần mô tả tại <summary> trên giao diện

Thêm <remarks> element vào Create method sẽ cung cấp thêm thông tin cho Swagger UI

/// <summary>

/// Creates a TodoItem.

/// </summary>

/// <remarks>

/// Sample request:

///

/// POST /Todo

/// {

/// "id": 1,

/// "name": "Item1",

/// "isComplete": true

/// }

///

/// </remarks>

/// <param name="item"></param>

/// <returns>A newly created TodoItem</returns>

/// <response code="201">Returns the newly created item</response>

/// <response code="400">If the item is null</response>

[HttpPost]

[ProducesResponseType(StatusCodes.Status201Created)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public ActionResult<TodoItem> Create(TodoItem item)

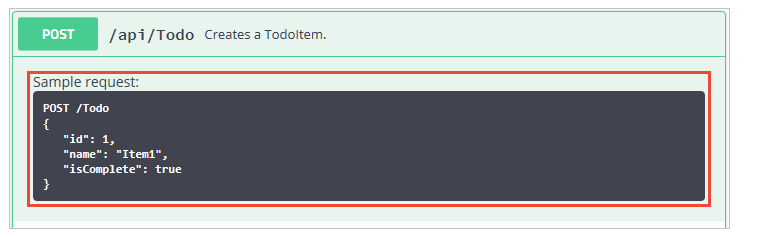
{

\_context.TodoItems.Add(item);

\_context.SaveChanges();

return CreatedAtRoute("GetTodo", new { id = item.Id }, item);

}

UI sau khi thêm <remarks>  


###### Data Annotation

Sử dụng namespace System.ComponentModel.DataAnnotations để thêm Attributes cho model

Thêm [Required] Attribute cho Name property, điều này khiến tro Name là một field bắt buộc phải điền trên UI nếu muốn gửi Request

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

namespace TodoApi.Models

{

public class TodoItem

{

public long Id { get; set; }

[Required]

public string Name { get; set; }

[DefaultValue(false)]

public bool IsComplete { get; set; }

}

}

Thêm [Produces(“application/json”)] Attribute cho API Controller, việc này chỉ định response type của API là kiểu application/json

[Produces("application/json")]

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

[ApiController]

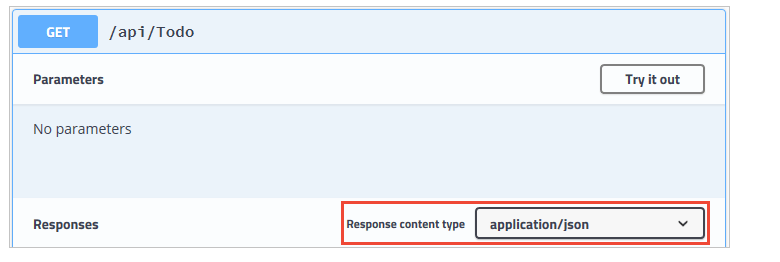
public class TodoController : ControllerBase

{

private readonly TodoContext \_context;

…

Response Content Type drop-down



###### Describe Response Type

Response types và error codes được chỉ biểu thị trong XML Comments và Data Annotations

Create action trả về một HTTP 201 Status code nếu Request thành công, HTTP 400 status code khi request body null. Muốn thể hiện việc này trên Swagger UI cần thêm code như sau

/// <returns>A newly created TodoItem</returns>

/// <response code="201">Returns the newly created item</response>

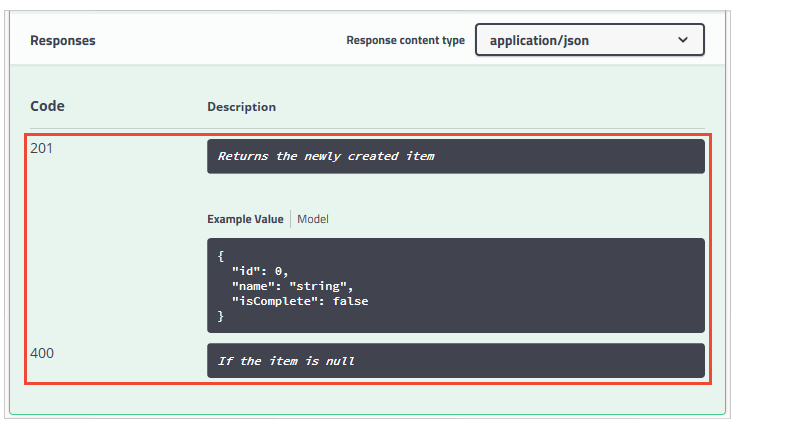
/// <response code="400">If the item is null</response>

[HttpPost]

[ProducesResponseType(StatusCodes.Status201Created)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public ActionResult<TodoItem> Create(TodoItem item)

Kết quả hiển thị trên Swagger UI  


# VALIDATE CONTROL DATESTART – DATEEND

## Features

Hỗ trợ bắt lỗi cố tình chọn ngày bắt đầu và ngày kết thúc bị sai trong hệ thống

## Update Note

* 05/11/2019: Release

## Access to the Code

##### Client Side

Function **vnr\_validateDateFrom\_DateTo** *in* HRM.Presentation.Main → js → vnr\_Hrm\_Function.js

## How to use

* Sử dụng hàm đối với trường hợp control nằm trên popup

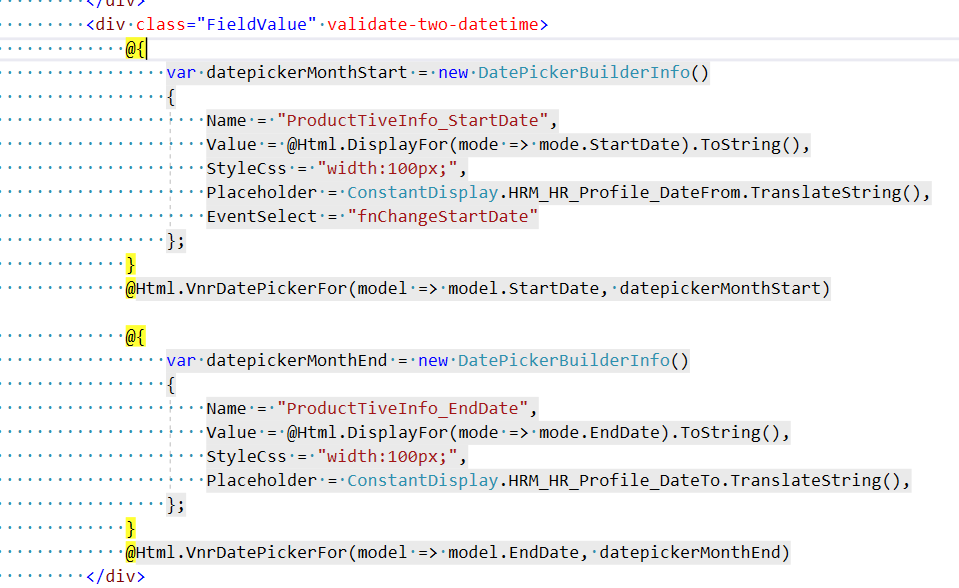
**vnr\_validateDateFrom\_DateTo(**elementName)

* Sử dụng attribute đối với các màn hình tìm kiếm

validate-two-datetime

* Với các cài đặt
  + **elementName** : Là “ID” của thẻ chứa 2 control cần kiểm tra

## Example

* Sử dụng attribute đối với các màn hình tìm kiếm