ASP.NET Core REST API Model Validation

In this video we will discuss **model validation in ASP.NET Core REST API**.

ASP.NET Core built-in model validation attributes

ASP.NET Core provides several built-in attributes for model validation. The following is the complete list.

* Required - Specifies the field is required
* Range - Specifies the minimum and maximum value allowed
* MinLength - Specifies the minimum length of a string
* MaxLength - Specifies the maximum length of a string
* Compare - Compares 2 properties of a model. For example compare Email and ConfirmEmail properties
* RegularExpression - Validates if the provided value matches the pattern specified by the regular expression

These validation attributes are in System.ComponentModel.DataAnnotations namespace. Also do not forget to install the following Nuget package.

Install-Package System.ComponentModel.Annotations

Copy

Model Validation in ASP.NET Core REST API

To implement model validation in an ASP.NET Core REST API, decorate the respective properties with the validation attributes. In the following example, FirstName is a required property. Should contain a minimum of 2 characters and must not exceed 100 characters.

public class Employee

{

public int EmployeeId { get; set; }

[Required]

[StringLength(100, MinimumLength = 2)]

public string FirstName { get; set; }

[Required]

public string LastName { get; set; }

[Required]

public string Email { get; set; }

public DateTime DateOfBrith { get; set; }

public Gender Gender { get; set; }

public int DepartmentId { get; set; }

public string PhotoPath { get; set; }

}

Copy

Custom model validation errors in ASP.NET Core REST API

To add a custom model validation error, use AddModelError() method of the ModelState object.

[HttpPost]

public async Task<ActionResult<Employee>> CreateEmployee(Employee employee)

{

try

{

if(employee == null)

{

return BadRequest();

}

// Add custom model validation error

var emp = employeeRepository.GetEmployeeByEmail(employee.Email);

if(emp != null)

{

ModelState.AddModelError("email", "Employee email already in use");

return BadRequest(ModelState);

}

var createdEmployee = await employeeRepository.AddEmployee(employee);

return CreatedAtAction(nameof(GetEmployee), new { id = createdEmployee.EmployeeId },

createdEmployee);

}

catch (Exception)

{

return StatusCode(StatusCodes.Status500InternalServerError,

"Error retrieving data from the database");

}

}

Copy

ASP.NET Core REST API and ModelState.IsValid check

In an ASP.NET Core REST API, there is no need to explicitly check if the model state is Valid. Since the controller class is decorated with the [ApiController] attribute, it takes care of checking if the model state is valid and automatically returns 400 response along the validation errors.

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

[ApiController]

public class EmployeesController : ControllerBase

{

}

Copy

The following is the response we get when the model validation fails

{

"type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",

"title": "One or more validation errors occurred.",

"status": 400,

"traceId": "|65b7c07c-4323622998dd3b3a.",

"errors": {

"Email": [

"The Email field is required."

],

"FirstName": [

"The field FirstName must be a string with a minimum length of 2 and a maximum length of 100."

]

}

}

Copy

EmployeeRepository GetEmployeeByEmail

public interface IEmployeeRepository

{

// Other interface methods

Task<Employee> GetEmployeeByEmail(string email);

}

public class EmployeeRepository : IEmployeeRepository

{

private readonly AppDbContext appDbContext;

public EmployeeRepository(AppDbContext appDbContext)

{

this.appDbContext = appDbContext;

}

public async Task<Employee> GetEmployeeByEmail(string email)

{

return await appDbContext.Employees

.FirstOrDefaultAsync(e => e.Email == email);

}

}