Title: Model Binding & Validations

Module: ASP.NET Core

Presented by: Narasimha Rao T

Model Binding & Validation in ASP.NET Core

By

Narasimha Rao T

Microsoft.Net FSD Trainer

Professional Development Trainer

tnrao.trainer@gmail.com

What is Model Binding?

1. What is Model Binding?

- **Definition:**
  Model binding in ASP.NET Core is the process of **mapping HTTP request data** (from form fields, query strings, route values, headers, etc.) to action method parameters or model objects.
- **Purpose:**
  Eliminates the need to manually parse request data (`Request.Form`, `Request.Query`, etc.).

- **Supported Sources:**
    - Form data (POST requests)
    - Query string (GET requests)
    - Route values

2. Implementing Model Binding

a) From Form

```
public class Student
{
    public string Name { get; set; }
    public int Age { get; set; }
}

[HttpPost]
public IActionResult Create(Student student)
{
    // Model binding maps form fields to Student properties
    return Ok(student);
}
```

- If the form has `<input name="Name">` and `<input name="Age">`, they automatically bind.

b) From Query String

```
[HttpGet("search")]
public IActionResult Search(string keyword, int page)
{
    // Example request: /search?keyword=ASP.NET&page=2
    return Ok($"Searching for {keyword}, page {page}");
}
```

c) From Route Values

```
[HttpGet("students/{id}")]
public IActionResult GetStudent(int id)
{
    // Example request: /students/10 -> id = 10
    return Ok($"Student ID: {id}");
}
```

Data Annotations for Validations

3. Data Annotations for Validation

ASP.NET Core provides attributes for **server-side validation**.

Common Data Annotations:

- `[Required]` – field must be provided
- `[StringLength(50)]` – restricts string length
- `[Range(18, 60)]` – numeric range
- `[EmailAddress]` – valid email format
- `[RegularExpression()]` – regex validation
- `[Compare("Password")]` – compare two fields

Example

```
public class RegisterModel
{
    [Required]
    public string Username { get; set; }

    [EmailAddress]
    public string Email { get; set; }

    [Range(18, 60)]
    public int Age { get; set; }
}
```

4. ModelState.IsValid Check

- Ensures the received model passes all validation rules.

```
[HttpPost]
public IActionResult Register(RegisterModel model)
{
    if (ModelState.IsValid)
    {
        return Ok("Registration successful");
    }
    return BadRequest(ModelState);
}
```

Custom Validations

5. Custom Validation

When built-in annotations are not sufficient.

Option 1: Custom Attribute

```
public class EvenNumberAttribute : ValidationAttribute
{
    public override bool IsValid(object value)
    {
        if (value is int num)
            return num % 2 == 0;
        return false;
    }
}

public class NumberModel
{
    [EvenNumber(ErrorMessage = "Only even numbers are allowed.")]
    public int Value { get; set; }
}
```

6. Real-Time Use Cases

- **User Registration Form** – model binding from form + validation rules
- **Search/Filter API** – query string model binding
- **REST APIs** – binding route values (`/api/products/{id}`)
- **Scheduling System** – custom validation for date ranges
- **E-commerce Checkout** – validating credit card, shipping address, etc.

Quiz Time

Sample Interview Questions

1. What is model binding and why is it important in ASP.NET Core?
2. What are the default data sources for model binding?
3. How do you validate a model in ASP.NET Core?
4. What is the difference between `ModelState.IsValid` and data annotations?
5. Can you create custom validation attributes? Give an example.
6. How does model binding handle complex types vs. simple types?
7. How do route parameters, query strings, and form data differ in binding?