¿Qué es un DTO con validaciones?

Un DTO (*Data Transfer Object*) es una clase que define **qué datos recibe o envía una API**.

Las **validaciones** en un DTO sirven para evitar:

- datos vacíos
- campos inválidos
- formatos incorrectos
- valores fuera de rango

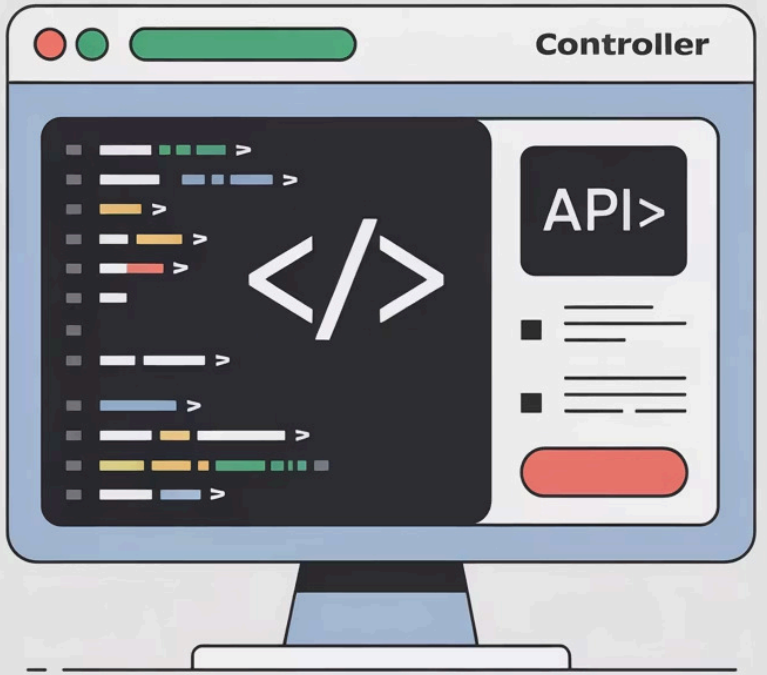
Y se ejecutan **antes de que el controlador procese la petición**.

✅ Decoradores más usados (Data Annotations)

Atributo	Qué valida
[Required]	El campo no puede ser nulo ni vacío
[StringLength]	Longitud máxima y mínima
[MaxLength], [MinLength]	Controlan longitud
[Range]	Valores numéricos en un rango permitido
[EmailAddress]	Formato de correo válido
[Phone]	Formato de teléfono
[Url]	Formato de URL
[RegularExpression]	Validación personalizada
[Compare]	Comparar campos (ej. contraseña y confirmación)

🎯 Ejemplo básico de un DTO con validaciones

```
using System.ComponentModel.DataAnnotations;

public class CustomerDto
{
 [Required(ErrorMessage = "El nombre es obligatorio")]
 [StringLength(50, MinimumLength = 3,
 ErrorMessage = "Debe tener entre 3 y 50 caracteres")]
 public string Name { get; set; } = "";

 [Required(ErrorMessage = "El correo es obligatorio")]
 [EmailAddress(ErrorMessage = "Correo no válido")]
 public string Email { get; set; } = "";

 [Range(18, 100,
 ErrorMessage = "La edad debe estar entre 18 y 100 años")]
 public int Age { get; set; }

 [Phone(ErrorMessage = "Teléfono no válido")]
 public string Phone { get; set; } = "";

 [Required]
 public int CategoryId { get; set; }
}
```

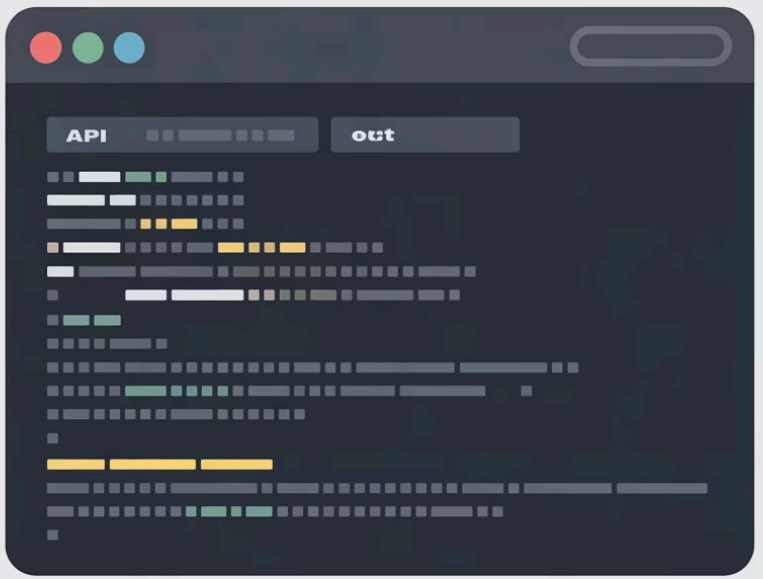


🚀 Cómo se usa en el controlador

ASP.NET Core valida el DTO automáticamente al recibirlo.

```
[HttpPost]
public IActionResult Create(CustomerDto dto)
{
 if (!ModelState.IsValid)
 return BadRequest(ModelState);
 // Devuelve errores de validación

 // Aquí continuarías guardando en la base de datos
 return Ok("Cliente creado");
}
```

🧪 Ejemplo de error de validación devuelto por la API




```
{
 "Name": [
 "Debe tener entre 3 y 50 caracteres"
 ],
 "Email": [
 "Correo no válido"
 ]
}
```

🛠️ Validación personalizada (extra)

```
public class ProductDto
{
 [Required]
 public string Name { get; set; }

 [Range(1, 100000)]
 public decimal Price { get; set; }

 [RegularExpression(@"^[A-Z0-9]{8}$",
 ErrorMessage = "El código debe tener 8 caracteres alfanuméricos en mayúscula")]
 public string Code { get; set; }
}
```

🧠 Consejo importante

(muy usado en APIs)

Siempre valida en el DTO para evitar que llegue basura al controlador.

Cuando tengas relaciones como CategoryId, sí debes poner:

```
[Required]
public int CategoryId { get; set; }
```