Leveraging and Validating Complex Enterprise Data Models

Gill Cleeren
ARCHITECT

@gillcleeren www.snowball.be

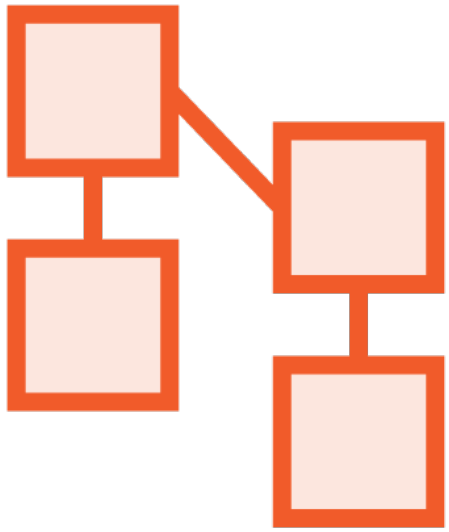
Overview

Model binding

Validating data

Model Binding

Model binding allows us to create .NET objects from the data in an HTTP request, used as parameters for action methods



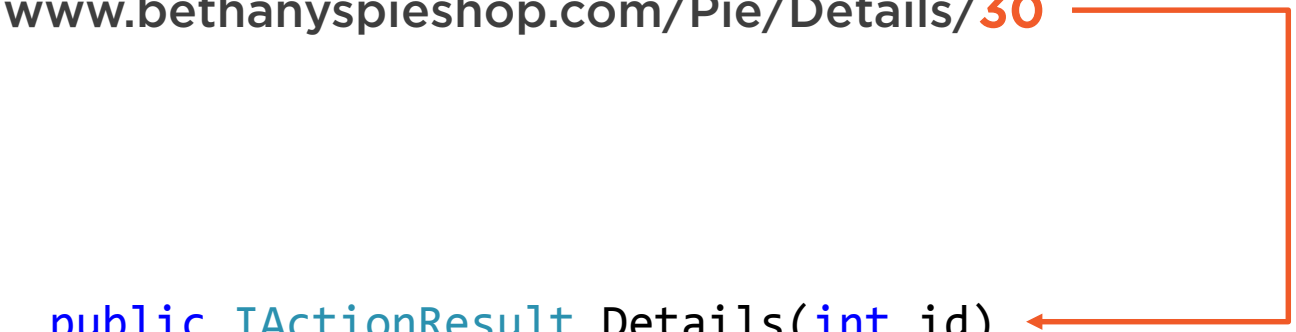
Binding the values is abstracted away

Works with

- Simple types
- Complex types
- Arrays

Model Binding

www.bethanyspieshop.com/Pie/Details/30



```
public IActionResult Details(int id)
```

Model binders

- Form values
- Routing values
- Query string values

Binding Simple Types

www.bethanyspieshop.com/Pie/Details/**30**

```
public IActionResult Details(int id)
```

Works for numeric, Boolean, string & date values

www.bethanyspieshop.com/Pie/Details/**ABC**

```
public IActionResult Details(int id)
                              id = 0

public IActionResult Details(int? id)
```

Binding Complex Types

```
public IActionResult EditPie(Pie pie)
```

```
<form asp-controller="PieManagement"
  asp-action="EditPie" method="post" >
   <label asp-for="Name"></label>
   <input asp-for="Name"/>
</form>
```

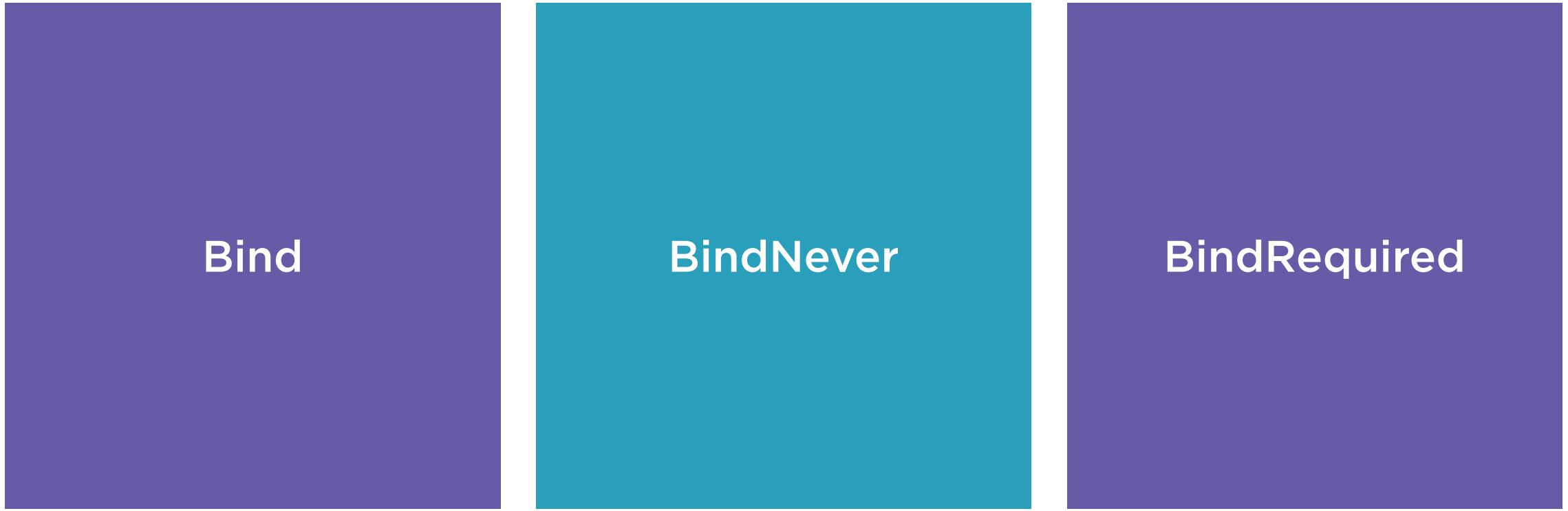
Binding Complex Types

```
public IActionResult EditPie(PieEditViewModel pieEditViewModel)
```

```
<form asp-controller="PieManagement"
  asp-action="EditPie" method="post" >
   <label asp-for="Pie.Name"></label>
   <input asp-for="Pie.Name"/>
</form>
```

```
public class PieEditViewModel
{
    public Pie Pie { get; set; }
    public int CategoryId { get; set; }
}
```

Binding Attributes



```
public IActionResult EditPie
    ([Bind("Pie")] PieEditViewModel pieEditViewModel)
{
    ...
}
```

The Bind Attribute

```
public class Order
{
    [BindNever]
    public int OrderId { get; set; }

}
```

The BindNever Attribute

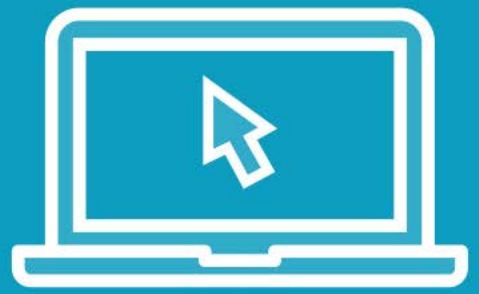
Binding to Collections

```
@model List<Pie>
<form asp-action="BulkEditPies" method="post">

@for (var i = 0; i < Model.Count; i++)
{
    <label asp-for="@Model[i].PieId"></label>

    <input asp-for="@Model[i].PieId" readonly="readonly" />
}

</form>
```

Binding to Collections

```
[HttpPost]
public IActionResult BulkEditPies(List<Pie> pies)
{

    //Do awesome things with the pie here
    return View();
}
```

Demo



Binding complex types

Binding a list

Specifying the Binding Source

FromBody

FromQuery

FromHeader

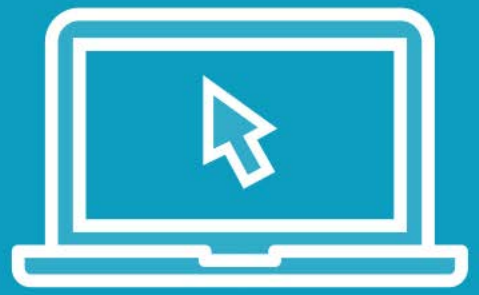
FromRoute

FromForm

```
public IActionResult EditPie([FromQuery]int pieId,
[FromHeader(Name = "Accept-Language")] string accept)
{

    ...

}
```

Specifying the Binding Source

Demo



Specifying the Binding Source

A question from Bethany

- My staff shouldn't be able to enter incorrect data into my application. How can we fix this?

Validating Data

```
public IActionResult AddPie(PieEditViewModel
pieEditViewModel)
{
    if (pieEditViewModel.Pie.Price < 0)
      ModelState.AddModelError
        (nameof(pieEditViewModel.Pie.Price),
        "The price of the pie should be higher than 0");

}
```

Validating the Model Manually

```
if (ModelState.IsValid)
{
    _pieRepository.CreatePie(pieEditViewModel.Pie);
    return RedirectToAction("Index");
}

return View(pieEditViewModel);
```

Model Validation

Validation Attributes

Required

StringLength

Range

RegularExpression

DataType

- Phone
- Email
- Url

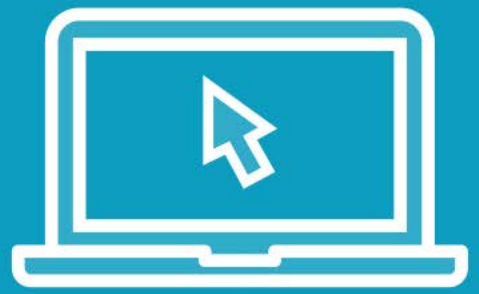
Validation Attributes

```
public class Order
{
    [Required(ErrorMessage = "Please enter your first name")]
    [StringLength(50)]
    public string FirstName { get; set; }

    [Required(ErrorMessage = "Please enter your phone number")]
    [StringLength(25)]
    [DataType(DataType.PhoneNumber)]
    public string PhoneNumber { get; set; }
}
```

Displaying Validation Errors

```
<form asp-controller="PieManagement" asp-action="EditPie"
method="post" class="form-horizontal" role="form">

    <h4>Update the pie details below</h4>

    <div asp-validation-summary="All"
        class="text-danger"></div>

</form>
```

Demo



Model validation

Displaying error messages

The Need for Custom Validation Attributes

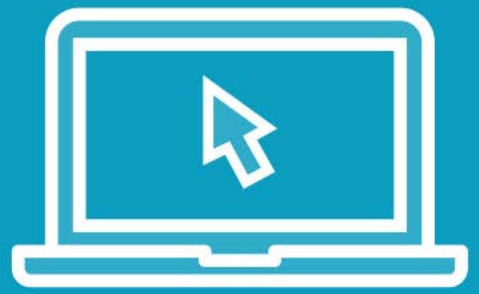
[Required]
[Range]
[Phone]
...

Custom validation rule?
→ Custom attribute

```
public class ValidUrlAttribute : Attribute, IModelValidator
{
    IEnumerable<ModelValidationResult>
        IModelValidator.Validate
            (ModelValidationContext context)

    {...}
}
```

Custom Validation Attribute

Demo



Creating a custom validation attribute

Client-side validation

Including the Correct Bower Packages

```
{
    "name": "asp.net",
    "private": true,
    "dependencies": {
       "bootstrap": "v3.3.7",
       "jquery": "2.2.4",
       "jquery-validation": "1.16.0",
       "jquery-validation-unobtrusive": "v3.2.6“
     }
}
```

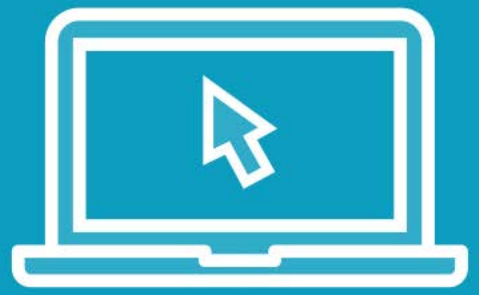
```
<script asp-src-include="lib/jquery/dist/jquery.js">
</script>

<script asp-src-include="lib/jquery-validation/dist/jquery.validate.min.js">
</script>

<script asp-src-include="lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js">
</script>
```

Including the Correct Scripts

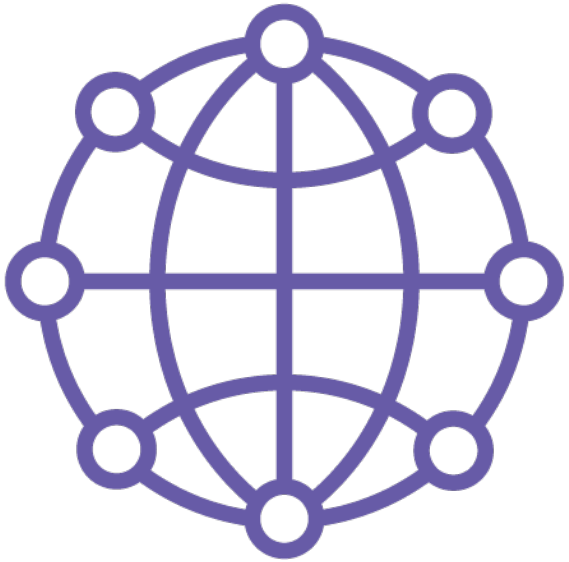
```
<input
    class="form-control" type="text"
    data-val="true"
    data-val-length="The field Last name must be a string
        with a maximum length of 50."
    data-val-length-max="50"
    data-val-required="Please enter your last name"
    id="LastName" name="LastName" value="" />
```

Generated data- Attributes

Demo



Validating data on the client



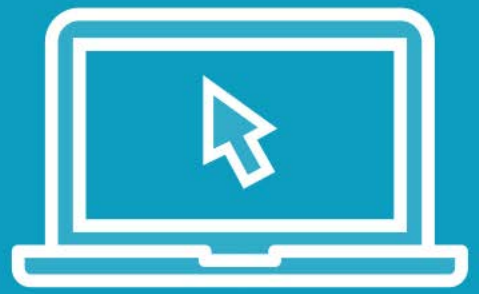
Remote validation

- Using [Remote] attribute
- Invoke action method

```
[Remote("CheckIfPieNameAlreadyExists", "PieManagement",
    ErrorMessage = "That name already exists")]
public string Name { get; set; }
```

Remote Validation

Demo



Using remote validation

Summary

- Model binding automates a lot of tedious work
- Strong combination with built-in validation features
- Validation mechanism can be extended

Up next:
Clean view code