



Module 6

Developing Models



Module Overview

- Creating Models
- Working with Forms
- Validating Data





Lesson 1: Creating Models

- Developing Models
- Passing Models to Page Content
- Binding Models to Pages and Displaying Data
- Demonstration: How to Bind Models To Pages
- What Are Model Binders?
- Adding CRUD Operations to Pages



Developing Models

A model is a collection of classes



The Photo and Comment Model Classes

```
public class Photo {  
    public int PhotoID { get; set; }  
    public string Title { get; set; }  
    public byte[] PhotoFile { get; set; }  
    public string Description { get; set; }  
    public DateTime CreatedDate { get; set; }  
    public string Owner { get; set; }  
    public virtual ICollection<Comment> Comments { get; set; }  
}
```

```
public class Comment {  
    public int CommentID { get; set; }  
    public int PhotoID { get; set; }  
    public string UserName { get; set; }  
    public string Subject { get; set; }  
    public string Body { get; set; }  
    public virtual Photo Photo { get; set; }  
}
```



Passing Models to Page Content

The instances of model classes are usually created in a page handler and passed to the Page Content through Properties



One Way Binding

With one way binding the content can be rendered in the response

```
public class SomePage : PageModel {  
    public SomeModel Model {get; set;}  
    public ActionResult OnGet() {  
        Model = new SomeModel() { Text = "some text" };  
    }  
}
```

Binding Page Content to Model Classes and Displaying Data

```
public class Person
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
}
```

```
<label asp-for="Model.FirstName"/>
<input asp-for="Model.FirstName"/>
<label asp-for="Model.LastName"/>
<input asp-for="Model.LastName"/>
```



First Name

Last Name

Submit my name

▲ Demonstration: How to Bind Model Classes to Page Content

In this demonstration, you will learn how to:

- Add a model to a Web application
- Pass a model from Page Models to Page Content
- Render the model properties using **asp-for="Model.PropertyName"**



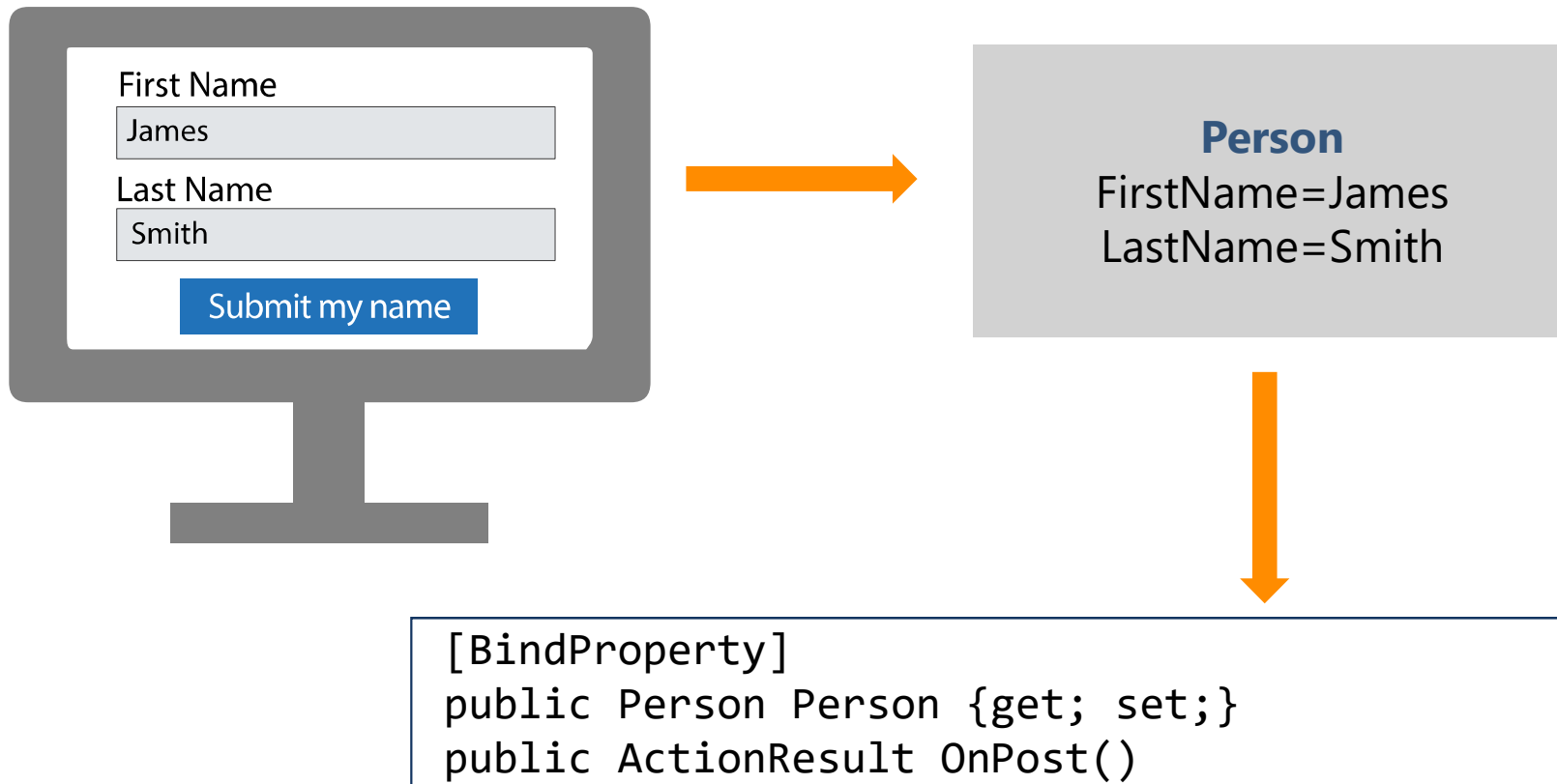


What Are Model Binders?

- The ASP.NET Core runtime uses model binders to determine how parameters are passed to handlers and how PageModel Properties are hydrated
- The default model binder uses the following logic:
 - The binder examines the definition of the handler that it must pass parameters to
 - The binder searches for values in the request that can be passed as parameters
 - If a PageModel Property is decorated with the [BindProperty] attribute, the ModelBinder searches for values in the request that can be used to set the property



Passing Parameters to Actions



Two Way Binding

With two way binding, the ModelBinder updates the property on Postback

```
public class SomePage : PageModel {  
    [BindProperty]  
    public SomeModel Model {get; set;}  
    public ActionResult OnGet() {  
        Model = new SomeModel() { Text = "some text" };  
    }  
    public ActionResult OnPost(){  
        //Model updated  
    }  
}
```



Lesson 2: Working with Forms

- Using Display and Edit Data Annotations
- Using Display Helpers
- Using Editor Helpers
- Using Form Helpers
- Demonstration: How to Use Display and Edit Data Annotations

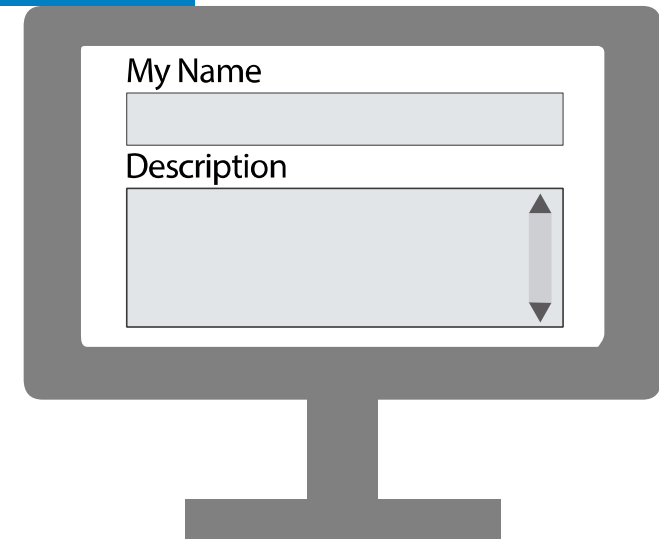


Using Display and Edit Data Annotations

```
public class Person
{
    [Display(Name="My Name")]
    public string Name { get; set; }

    [DataType(DataType.MultilineText)]
    public string Description { get; set; }
}
```

@EditorForModel



Using Display Helpers

- **Html.DisplayNameFor()**

```
@Html.DisplayNameFor(model => model.FirstName)
```

 First Name:

- **Html.DisplayFor()**

```
@Html.DisplayFor(model => model.FirstName)
```

 James

Using Editor Helpers

- **Html.LabelFor()**

```
@Html.LabelFor(model => model.ContactMe)
```



```
<label for="ContactMe">  
  Contact Me  
</label>
```

- **Html.EditorFor()**

```
@Html.EditorFor(model => model.ContactMe)
```



```
<input type="checkbox"  
  name="ContactMe" />
```


Using Editor Helpers (Continued)

- **LabelTagHelper**

```
<label asp-for="ContactMe"></label>
```



```
<label for="ContactMe">  
    Contact Me  
</label>
```

- **InputTagHelper**

```
<input asp-for="ContactMe" />
```



```
<input type="checkbox"  
    name="ContactMe" />
```

Using Form Helpers

- **Html.BeginForm**

```
@using (Html.BeginForm("ShowDetails", "Person")) {  
}
```



```
<form  
  action="/Person/ShowDetails"  
  method="post">  
</form>
```

- **FormTagHelper**



```
<form asp-controller="Person"  
      asp-action="ShowDetails">  
</form>
```



Demonstration: How to Use Display and Edit Data Annotations

In this demonstration, you will learn how to:

- Add data annotations to a model
- Build a form in the view by using form helpers
- Render the properties from the view to the browser by using editor helpers
- Render the properties from the view to the browser by using display helpers





Lesson 3: Validating MVC Application

- Validating User Input with Data Annotations
- Using Validation Helpers
- Demonstration: How to Validate User Input with Data Annotations
- Adding Custom Validations
- Demonstration: How to Add Custom Validations



Validating User Input with Data Annotations

```
public class Person
{
    [Required(ErrorMessage = "Please enter a name.")]
    public string Name { get; set; }

    [Range(0, 150)]
    public int Age { get; set; }


    [Required]
    [RegularExpression("^.+\\@.+.+$")]
    public string EmailAddress { get; set; }

    [DataType(DataType.MultilineText)]
    [StringLength(20)]
    public string Description { get; set; }
}
```

Using Validation Helpers

- **Html.ValidationSummary()**


```
@Html.ValidationSummary()
```



```
<ul>  
  <li>Please enter a name.</li>  
  <li>The EmailAddress field is required</li>  
</ul>
```

- **Html.ValidationMessageFor()**

```
@Html.ValidationMessageFor(model => model.Name)
```



```
Please enter a name.
```

Using Validation Helpers (Continued)

- **ValidationSummaryTagHelper**

```
<div asp-validation-summary="All"></div>
```



```
<ul>  
  <li>Please enter a name.</li>  
  <li>The EmailAddress field is required</li>  
</ul>
```

- **ValidationMessageTagHelper**

```
<span asp-validation-for="Name"></span>
```



Please enter a name.



Demonstration: How to Validate User Input with Data Annotations

In this demonstration, you will learn how to:

- Add validation data annotations to a model
- Use the **ModelState.IsValid** property in a handler
- Use validation helpers in a Page Content



Adding Custom Validations

- Create custom validation data annotations

```
public class AllLettersValidationAttribute : ValidationAttribute {  
    public override bool IsValid(Object value) {  
        return ((string)value).All(Char.IsLetter);  
    }  
}
```

- Use custom validation data annotations

```
[AllLettersValidation(ErrorMessage = "Only letters allowed.")]  
public string Name { get; set; }
```



Demonstration: How to Add Custom Validations

In this demonstration, you will learn how to add custom validations





Lab: Developing Models

- Exercise 1: Adding a Model
- Exercise 2: Working with Forms
- Exercise 3: Adding Validation

Estimated Time: 60 minutes

