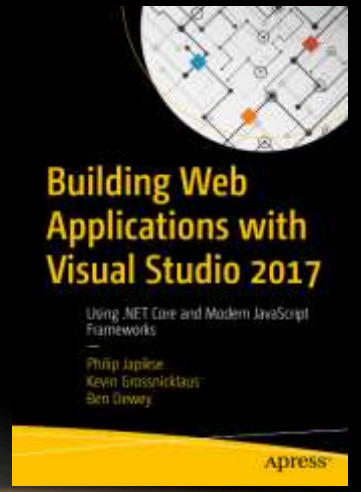
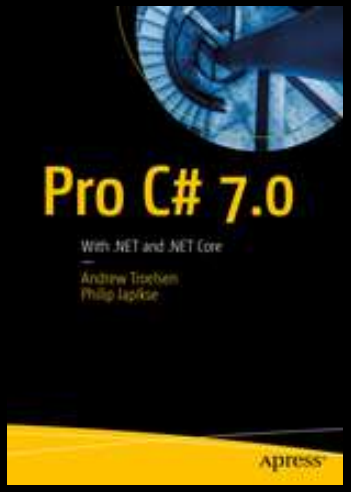
MOVING TO ASP.NET CORE MVC

Philip Japikse (@skimedic)
skimedic@outlook.com
www.skimedic.com/blog
Microsoft MVP, ASPInsider, MCSD, MCDBA, CSM, PSM II, PSD
Consultant, Teacher, Writer

Phil.About()

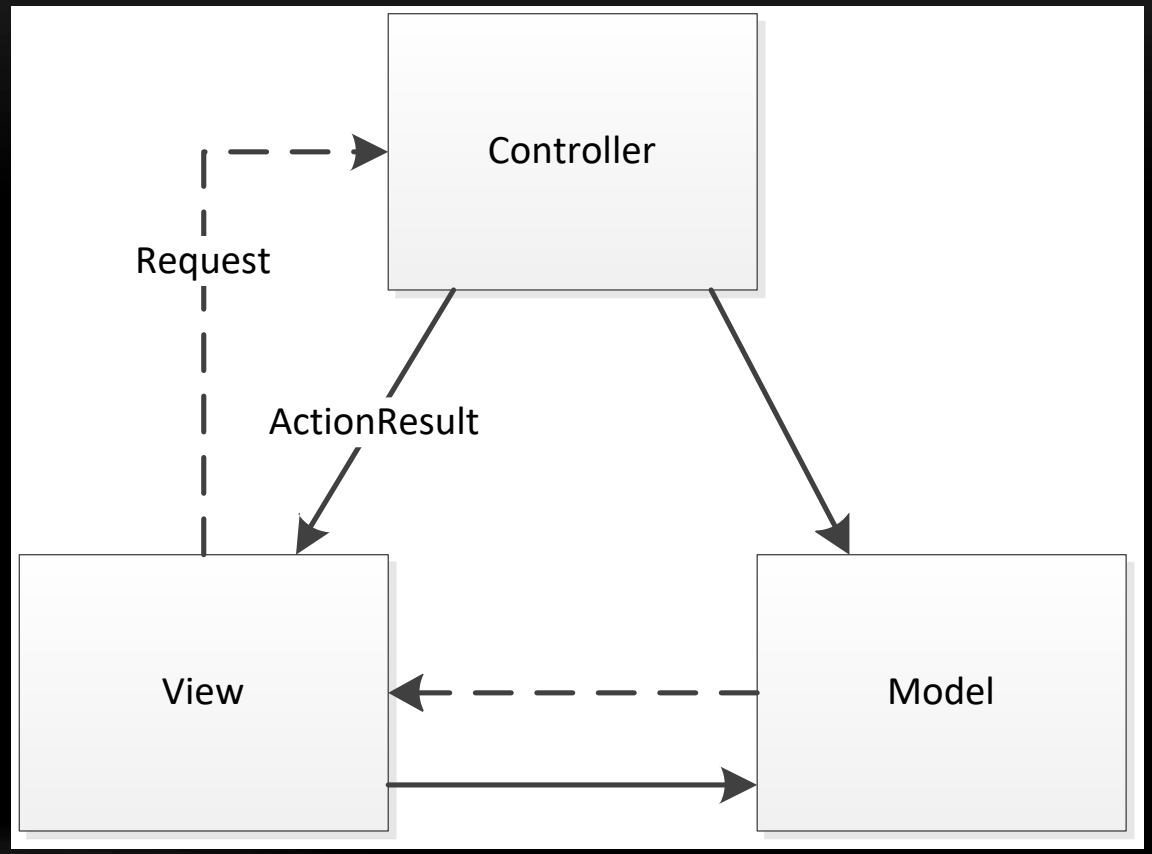
- Director of Consulting/Chief Architect
- Author: Apress.com (http://bit.ly/apressbooks)
- Speaker: http://www.skimedic.com/blog/page/Abstracts.aspx
- Microsoft MVP, ASPInsider, MCSD, MCDBA, CSM, PSM II, PSD
- Founder, Agile Conferences, Inc.
  - http://www.cincydeliver.org
- President, Cincinnati .NET User's Group





WHY MOVE TO MVC OVER WEBFORMS?

- Full Control over HTML and URLS
  - Multiple HTML Forms
- Separation of Concerns
- Extensible
- Testable
- Convention over Configuration
- Dependency Injection Support
- Plus many more reasons!



IT'S THE FUTURE

CONVENTION OVER CONFIGURATION

- Controllers
  - <*name*>Controller
- Views
  - Named for Action
  - Placed in /Views/<ControllerName>
- Layouts
  - Start with "_" (carryover from WebMatrix)
- Project Structure

WHY NOT MVC?

THE MODEL VIEW CONTROLLER PATTERN

MODELS

- The data of the application
- Supports
  - Data Annotations
  - Additional Metadata

VIEW MODELS

- Façade for individual models
- Transport mechanism for models

VIEWS

- Strongly Typed
- Accepts Interactions from User
- Returns results of interactions back to user

CONTROLLERS

- Process Incoming requests
- Perform changes to the model
- Select views to render to the user

CONTROLLERS AND ACTIONS

CONTROLLERS AND ACTIONS

- Controllers live in Controllers folder
- Controller, AsyncController, APIController all rolled into one
- All return IActionResult (or Task<IActionResult>)
- HTTP Verbs
  - HttpGet is default
  - All others must be specified (no long based on name of method)

BASE CONTROLLER

Helper Methods

- Content/NoContent/File
- Redirect[Permanent]
- LocalRedirect[Permanent]
- Ok/Accepted/CreatedAt
- UnAuthorized
- NotFound/BadRequest/Forbid
- StatusCode

Events

- OnActionExecuting
- OnActionExecuted

ACTIONS

- Methods of a controller
  - Each Exposed by unique URLS
- Return an ActionResult
  - View, Redirect, Json, etc.

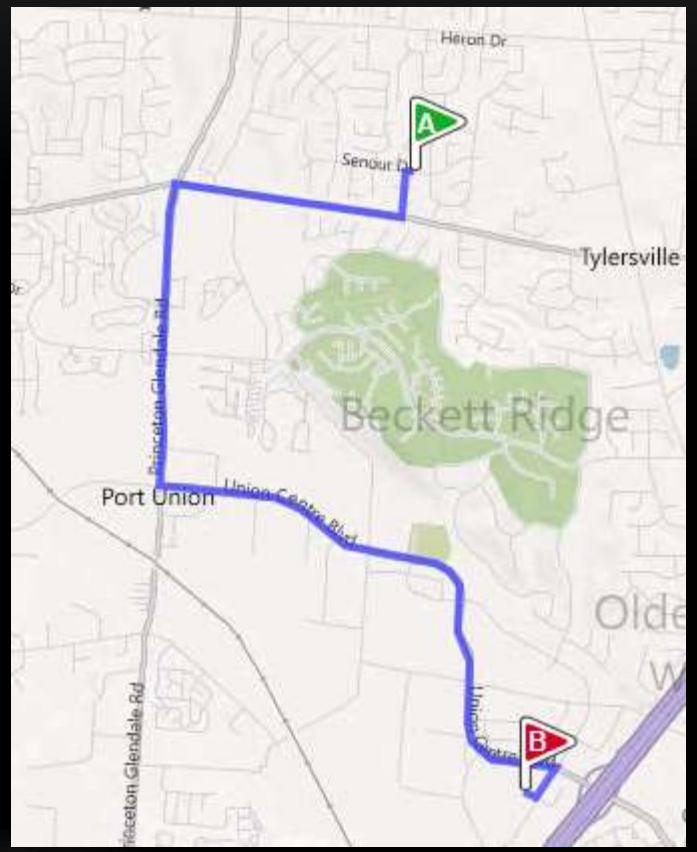
ACTION FILTERS

- Authorization
- Action
- Result
- Exception
- Custom

ROUTING

ROUTING

- Directs Incoming URLS to correct Controller/Actions
- Creates Outgoing URLS
- SEO Optimized



TRADITIONAL ROUTING

- Routes are added to the Route Table
- Routes are added using templates:
  - Area/Controller/action
  - Customer values e.g. {id}
- Parameters can be constrained
  - Optional ?
  - Types int,bool,datetime,decimal, etc
- First matching route wins

```
app.UseMvc(routes =>
{
  routes.MapRoute("default",
  "{controller=Home}/
   {action=Index}/
   {id?}");
});
```

ATTRIBUTE ROUTING

- Uses Attributes on Controllers and Actions
- Can be added to HTTP Verbs
  - [HttpGet(“Home”)]
- Can be named
- Controller routes are combined with Action routes
  - All actions must use attribute routing if controller has a route
  - Route attributes with “/” reset the route
- Reserved tokens use “[]” instead of “{}”

```
public class HomeController : Controller
{
 [Route("")]
 [Route("Home")]
 [Route("Home/Index")]
 public IActionResult Index()
 {
  return View();
 }
 [Route("Home/About")]
 public IActionResult About()
 {
  return View();
 }
 [Route("Home/Contact")]
 public IActionResult Contact()
 {
  return View();
 }
}
```

MODELS, MODEL BINDING, VALIDATION

MODEL BINDING

- Reconstitutes form values into target object using reflection from:
  - Form values
  - Route values
  - Query strings
- Converts types
  - Failures are added to ModelState
- Manual or Automatic

```
[HttpPost]
public async Task<IActionResult> ChangePassword(
  ChangePasswordViewModel viewModel)
{
  if (!ModelState.IsValid)
  {
    return View(viewModel);
  }

  //Omitted for brevity

  bool foo = await TryUpdateModelAsync(viewModel);

  //Omitted for brevity
  return View(viewModel);
}
```

CUSTOMIZE MODEL BINDING

- [BindRequired]: This attribute adds a model state error if binding cannot occur.
- [BindNever]: Tells the model binder to never bind to this parameter.
- [FromHeader], [FromQuery], [FromRoute], [FromForm]: Use these to specify the exact binding source you want to apply.
- [FromServices]: This attribute uses dependency injection to bind parameters from services.
- [FromBody]: Use the configured formatters to bind data from the request body. The formatter is selected based on content type of the request.
- [ModelBinder]: Used to override the default model binder, binding source and name.

MODEL VALIDATION

- Server side occurs after binding using:
  - Conventions
  - Data Annotations
  - Custom annotations
- Server side can be triggered manually
- Client side with JavaScript

```
public class Movie
{
 public int Id { get; set; }

 [Required]
 [StringLength(100)]
 public string Title { get;set;}

 [ClassicMovie(1960)]
 [DataType(DataType.Date)]
 public DateTime ReleaseDate { get; set;}

 [Range(0, 999.99)]
 public decimal Price { get; set; }

 [Required]
 public Genre Genre { get; set; }
}


TryValidateModel(movie);
```

COMMON VALIDATION ATTRIBUTES

- [CreditCard]: Validates the property has a credit card format.
- [Compare]: Validates two properties in a model match.
- [EmailAddress]: Validates the property has an email format.
- [Phone]: Validates the property has a telephone format.
- [Range]: Validates the property value falls within the given range.
- [RegularExpression]: Validates that the data matches the specified regular expression.
- [Required]: Makes a property required.
- [StringLength]: Validates that a string property has at most the given maximum length.
- [Url]: Validates the property has a URL format.

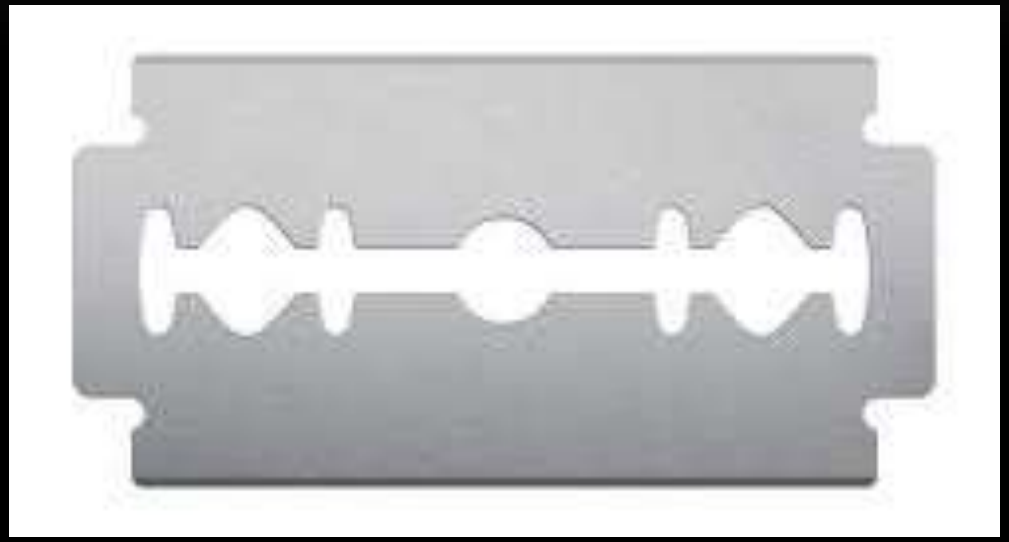
MODEL STATE ERRORS

- Validation occurs:
  - After binding
  - Before action execution
- Errors are added to the ModelState
  - Default max is 200
  - Each property is represented in the collection, valid or not
- If any properties have errors (binding or validation), IsValid = false

RAZOR

THE RAZOR VIEW ENGINE

- Greatly reduces typing
- Mixes HTML and Code cleanly
  - Increases readability
- Compiled at runtime



LESS TYPING, EASIER TO READ

```
@{
    //Code Block
    var foo = "Foo";
    var bar = "Bar";
    var htmlString =
"<ul><li>one</li><li>two</li></ul>";
}
@foo<br />
@htmlString<br />
@foo.@bar<br />
@Html.Raw(htmlString)
```

```
<%
    //Code Block
    var foo = "Foo";
    var bar = "Bar";
    var htmlString =
"<ul><li>one</li><li>two</li></ul>";
%>
<%= foo %><br />
<%: htmlString%><br />
<%=foo %>.<%=bar%><br />
<%=htmlString %>
```

LESS TYPING, EASIER TO READ

```
@for (int i = 0; i <5; i++)
{
  @:Straight Text
  <div>Value:@i</div>
  <text>
  Lines without HTML tag
  </text>
  <br/>
}
----------------------------
<a href="/Products/Detail/@Model.Id">
  @Model.Details.ModelName</a>
```

```
<% for (int i = 0; i < 5; i++)
          {
              %>
              Straight Text
             <div>Value:<%=i%></div>
              Lines Without HTML tag
    <br />
 <% } %>

----------------------------
<href="/Products/Detail/<%=p.ProductID%>">
<%= p.ProductName%></a>
```

SMART @ HANDLING

```
@{ var foo = “Foo”; }
@*
Multiline Comments
Hi.
*@
Email Address Handling:
foo@foo.com = foo@foo.com
test@foo = test@foo
test@(foo) = testFoo
```

Problem Areas:

Generics:

```
var item = GetOrderItem<string>(order);
```

Must be:

```
var item = ( GetOrderItem<string>(order) );
```

Too many “.”:

“/orders/products/@Product.Name.jpg

Must be

“/orders/products/(@Product.Name).jpg

RAZOR FUNCTIONS & DELEGATES

```
@functions {
    public [static] IList<string>
SortList(IList<string> strings)  {
        var list = from s in strings orderby s select s;
        return list.ToList();
    }
}
----------------------------------------------------------
@{ var myList = new List<string>  {
        "C", "A", "Z", "F“ };
       var sortedList = SortList(myList);
//MyFunctions.SortList(myList)
}
@foreach (string s in sortedList) {
    @s 
}
```

```
@{
    Func<dynamic, object>
  b = @<strong>@item</strong>;
}
This will be bold: @b("Foo")
```

HTML HELPERS

- Ecapsulate rendering code
- Examples:
  - ActionLink
  - AntiforgeryToken
  - BeginForm
  - DisplayFor
  - EditorFor

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

@Html.EditorForModel()

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

TAG HELPERS

- Enable server-side code to participate in rendering HTML elements in Razor views
- Reduces the transition between code and markup
  - Tag Helpers Attach to HTML elements
  - HTML Helpers are invoked as methods
- Fully supported with IntelliSense

VIEWS, LAYOUTS, AND TEMPLATES

RAZOR VIEWS

- HTML Template with Razor markup
- Strongly typed
- Located in the Views/[ControllerName] or Views/Shared folder
- Layouts provide consistent sections
- Partial views don’t use a layout or have server-side code
- View Components combine Partial views and server-side code

VIEW LAYOUTS

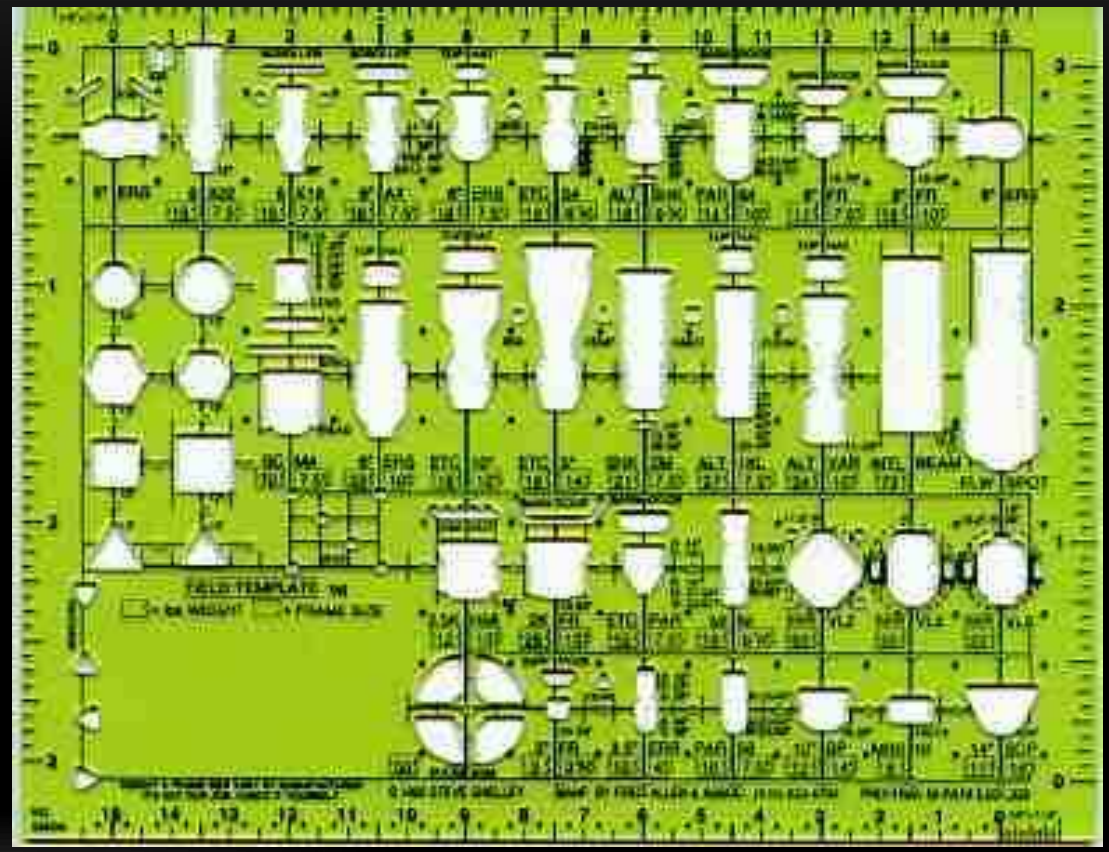
- Provide consistency across Views
- Contains the HTML/Head/Body/Footer of the HTML page
- Default layout is specified in _ViewStart.cshtml
  - View Layout property sets the layout from the default
- Sections provide a way to organize where certain page elements should be placed
  - Sections can be required or optional

PASSING DATA TO THE VIEW

- ViewModels
- ViewData – Data Dictionary
- ViewBag – Dynamic Dictionary
- TempData

TEMPLATES

- Templates
  - DisplayForModel || EditorForModel
  - DisplayFor || EditorFor



CREATING TEMPLATES

- Default templates:
  - <TypeName>.cshtml (e.g. Boolean.cshtml)
  - Add to DisplayTemplates/EditorTemplates folder
    - Under Shared for all
- Named Templates
  - @Html.DisplayFor(x=>x.foo,"Template")
  - @Html.EditorFor(x=>x.foo,"Template")

DISPLAY TEMPLATES

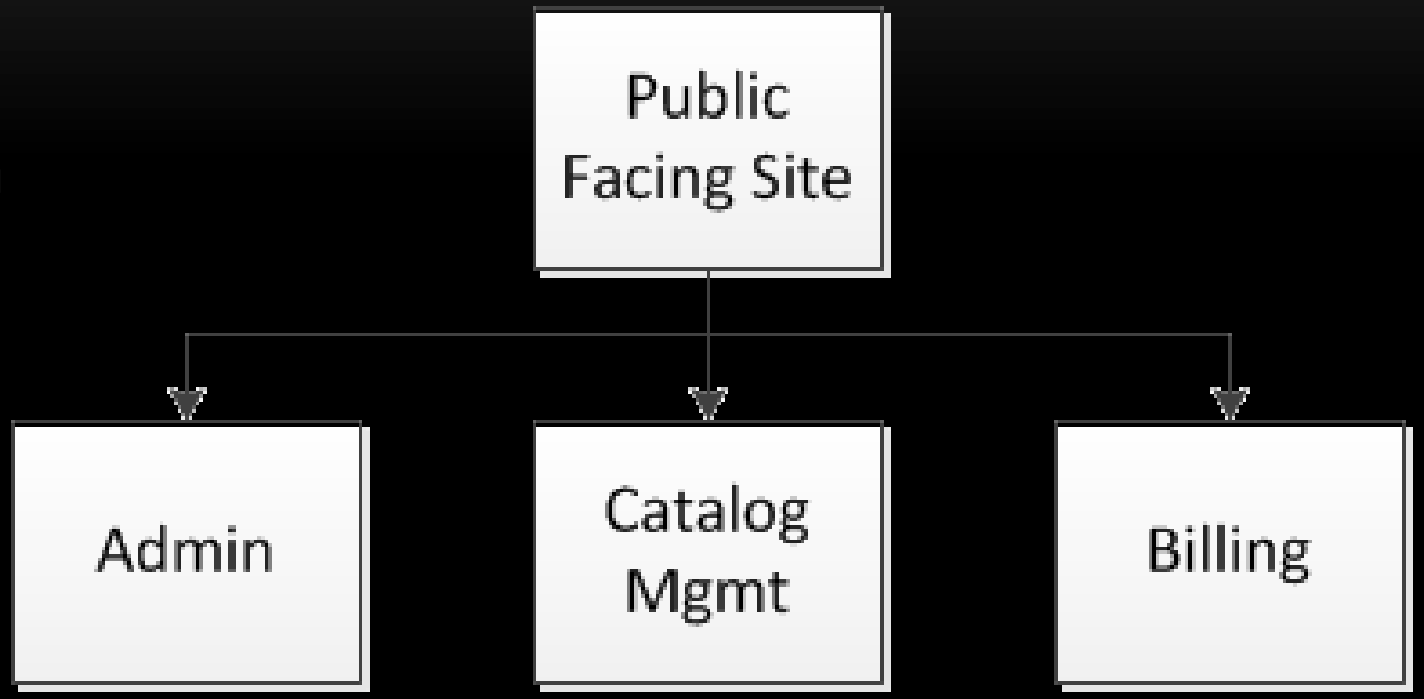
```
@Html.DisplayFor(model => model.[boolean])
----------------------------------------
@model bool?
@{
    if (ViewData.ModelMetadata.IsNullableValueType)
    {
        if (!Model.HasValue) { @:Unknown }
        else if (Model.Value) { @:Yes }
        else { @:No }
    } else {
        if ((bool)Model) { @:Yes }
        else { @:No }
    }
}
```

EDITOR TEMPLATES

```
@Html.EditorFor(model => model.[boolean] )
----------------------------------------
@{
var value = false;
var selectionMade = false;
var list = new List<SelectListItem>();
if (ViewData.ModelMetadata.IsNullableValueType)
{
  list.Add(new SelectListItem
  { Text = "", Value = ""});
  if (!Model.HasValue)
  {
     list[0].Selected = true;
     selectionMade = true;
  }
  else { value = Model.Value;}
}
else
{
  if (Model == null) {value = false;}
  else { value = (bool)Model; }
}
list.Add(new SelectListItem
{
  Text = "Yes",
  Value = "true",
  Selected = (!selectionMade && value == true)
});
list.Add(new SelectListItem
{
  Text = "No",
  Value = "false",
  Selected = (!selectionMade && value == false)
});
}
@Html.DropDownList(
  string.Empty, list, ViewData["htmlAttributes"])
```

AREAS

- Functional Segments of application
- Enhance Separation of Controls
- Fully supported by routing system
  - Add namespace to routing



.NET CORE & ASP.NET CORE

WHAT IS .NET CORE?

- Rewrite of “full” .NET Framework
- Vast performance improvements over prior versions
  - Including native compilation
- Flexible deployment model
  - Windows, Linux, Mac

- Full command line support
- True side by side installation support
- Open source from the start
  - Many improvements and features provided by the community

ASP.NET CORE

- ASP.NET Core is ASP.NET rebuilt on top of .NET Core
- Single, cross-platform framework for web, services, and microservices
  - WebApi + MVC + Web Pages + Razor Pages = ASP.NET Core
- Takes advantage of .NET Core performance
  - Includes a high performance web server (Kestrel) built on LibUV

Contact Me

skimedic@outlook.com
www.skimedic.com/blog
www.twitter.com/skimedic

http://bit.ly/skimediclyndacourses
http://bit.ly/apressbooks

www.hallwayconversations.com

Questions?

Thank You!

Find the code at: https://github.com/skimedic/presentations/tree/master/DOTNETCORE/ASP.NETCore/SpyStore