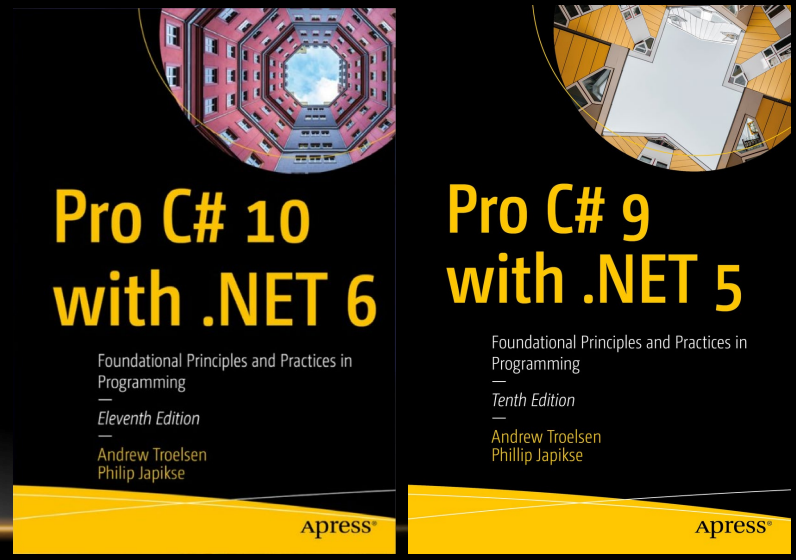

Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS
AUSTIN
Getting Started with ASP.NET Core
Philip Japikse
#VSLIVE
NO CODE LIMITS

GETTING STARTED WITH ASP.NET CORE

Philip Japikse (@skimedic)
skimedic@outlook.com
CTO, Author, Teacher
Microsoft MVP, ASPInsider, PST, PSM II, PSD

Phil.About()

- CTO/Chief Architect, Pintas & Mullins
- Author: Apress.com (http://bit.ly/apressbooks)
- Professional Scrum Trainer (PSF, PSD)
- Speaker: http://www.skimedic.com/blog/page/Abstracts.aspx
- Microsoft MVP, ASPInsider, PST, PSM II, PSD
- Founder, Agile Conferences, Inc.
  - http://www.cincydeliver.org
- President, Cincinnati .NET User's Group



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

.NET (CORE) SUPPORT LIFECYCLES

.NET CORE SUPPORT LIFECYCLES

- Long Term Support (LTS)
  - Only upgraded with critical fixes (patches)
  - Supported for three years after GA release

  or
  - At least one year after the next LTS release.

  - NOTE:
    - 2.1 LTS (support ended 8/21/21)
    - 3.1 LTS (support until 12/03/22)
    - 6.0 LTS (support until ~Q4/2024)

- Current (STS)
  - Minor releases
  - Upgraded more rapidly
  - Supported for ~~three~~ six months after:
    - Next Current or LTS release

  - NOTE:
    - 1.0, 1.1, 2.0, 2.1, 2.2, 3.0 all end of lifed
    - 5.0 Current (support until ~05/10/2022)

https://www.microsoft.com/net/core/support

ASP.NET CORE FUNDAMENTALS

ASP.NET CORE

- ASP.NET Core rebuilt on top of .NET Core
- Single framework for web, services, and microservices
  - WebApi + MVC + Razor Pages = ASP.NET Core
- Cross-platform
  - Not tied to IIS or Windows
- Takes advantage of .NET Core performance
  - Includes a high performance web server (Kestrel) built on LibUV

MVC/SERVICE CONTROLLERS AND ACTIONS

CONTROLLERS AND ACTIONS (MVC WEB APPLICATION)

- All derive from single Controller class (which derives from ControllerBase)
  - Controller, AsyncController, APIController all rolled into one

- All actions return IActionResult (or Task<IActionResult>)

- Non-HttpGet methods must specify HTTP Verb
  - All methods should specify HTTP Verb
  - HttpGet is default, but unmarked action methods also support HttpPost
  - Posts should use AntiForgery Token
- Browsers only capable of Get and Post requests

CONTROLLERS AND ACTIONS (MVC WEB APPLICATION/SERVICES)

- ControllerBase contains helper methods for:
  - Returning HttpStatusCodes - NoContent (204), OK (200), BadRequest (400), etc.)
  - Redirecting to routes, action methods, razor pages

- Controller contains helper methods for UI support
  - Returning Views, PartialViews, etc

THE POST-REDIRECT-GET PATTERN (PRG)

- When a Post action completes, redirect the user to a Get action
- E.g. Create -> Details, Delete -> Index
- Prevents user double posting

RAZOR PAGES

RAZOR PAGES

- Eliminate the Controller class when building web applications
- Page contains markup (*.cshtml) and code (*.cshtml.cs)
- Binding through properties in code
- ViewBag replaced with properties in code
- Still uses MVC style layout, partials, and editors

RESTFUL SERVICES

BUILDING RESTFUL SERVICES WITH ASP.NET CORE

- Follows the Model/Controller pattern
- Controllers inherit from ControllerBase (not Controller)
- ApiController Attribute (applied at the project or controller level)
  - Enables REST-specific behavior for controllers
    - Automatic 400 responses on model validation errors
    - Binding source parameter inference
    - Multipart/form-data inference

ROUTING

ROUTING

- Determines with Controller/Action or Page to execute
- Used to generate URLs with-in the application
- MVC Web Applications/Restful Services
  - Route is independent of file structure or names
- Razor Page Web Applications
  - Route is based on file structure and name

ATTRIBUTE ROUTING (MVC WEB APPLICATIONS, RESTFUL SERVICES)

- First class citizen in ASP.NET Core
- Controllers - Uses Route Attribute
- Actions - Uses Route Attribute or added to HTTP Verbs
- Controller routes are combined with Action routes
  - Actions use attribute routing when controller has route
  - Route attributes with “/” reset the route
- Reserved tokens use “[]” instead of “{}”

RAZOR PAGE WEB APPLICATION ROUTING

- Based on application structure
- Route parameters are added to the @page directive
- Reserved tokens use “{}”

MODEL BINDING AND VALIDATION

MODEL BINDING

- Implicit model binding uses action method parameter
- Sources for data (in order):
  - Form fields
  - The request body (API services)
  - Route data
  - Query string parameters
  - Uploaded files
- Route data and query string values are used only for simple types.

MODEL BINDING

- Can explicitly specify the source:
  - [FromQuery] - query string.
  - [FromRoute] - route data.
  - [FromForm] - posted form fields.
  - [FromBody] - request body.
  - [FromHeader] - HTTP headers.

- Can also explicitly call for model binding with code

VALIDATION

- Data annotations off built in validation
  - Execute server side and emit JavaScript for client-side validation

- Errors are added to ModelState
  - Errors in binding (e.g. data type issues)
  - Errors in validation (required, length, etc.)

- Errors displayed in UI with tag helpers

VIEWS AND LAYOUTS

VIEWS

- Rendered from an action method using View() or PartialView()
- Razor code mixes with markup
- Tag Helpers keep you in the mark up

- Located in Views\[ControllerName] or Views\Shared

PARTIAL VIEWS

- Don’t use a layout
- Render from an action method using PartialView()
- Render from another view using partial tag helper

LAYOUTS

- _ViewStart.cshtml sets default for views (can be configured per view)
- Layout
  - RenderBody renders the view
  - Sections add more control (required || optional)
    - RenderSection/IgnoreSection
- Partial Views don't use a layout

TEMPLATES

- Render automatically if named after a type and located
- Rendered on demand by passing in name

- DisplayTemplates (DisplayFor/DisplayForModel)
  - Located in Views\ControllerName\DisplayTemplates or Views\Shared\DisplayTemplates

- EditorTemplates (EditFor/EditForModel)
  - Located in Views\ControllerName\EditorTemplates or Views\Shared\EditorTemplates

CSS ISOLATION

- Views, Layouts, and Pages can have their own CSS files
- All get bundled into {ProjectName}.styles.css
  - Only bundled while debugging when in Development
  - Bundled when deployed
  - Can force bundling by calling UseStaticWebAssets

TAG HELPERS

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

- Can also create custom tag helpers

AREAS

AREAS

- Partitioned “mini” MVC sites within an application
- Has own controllers, views, ViewStart and ViewImports
  - Can move main ViewStart and ViewImports to top level to control main app and areas
- Used for parts of application that should be partitioned from main app (e.g. admin functions)

SWAGGER/SWASHBUCKLE

SWAGGER/OPENAPI

- Swagger is a language-agnostic spec for describing REST APIs
  - When donated to the OpenAPI Initiative was renamed to OpenAPI
- Provides interactive documentation, client SDK Generation, and API discoverability
  - Swagger.json describes the service, SwaggerUI presents it on the web
- Options for ASP.NET Core
  - Swashbuckle.AspNetCore generates docs
  - Nswag generates docs + client code

Contact Me

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

http://bit.ly/apressbooks

Questions?

Thank You!