iscte
INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UPskill

Contents

What will this module be about?

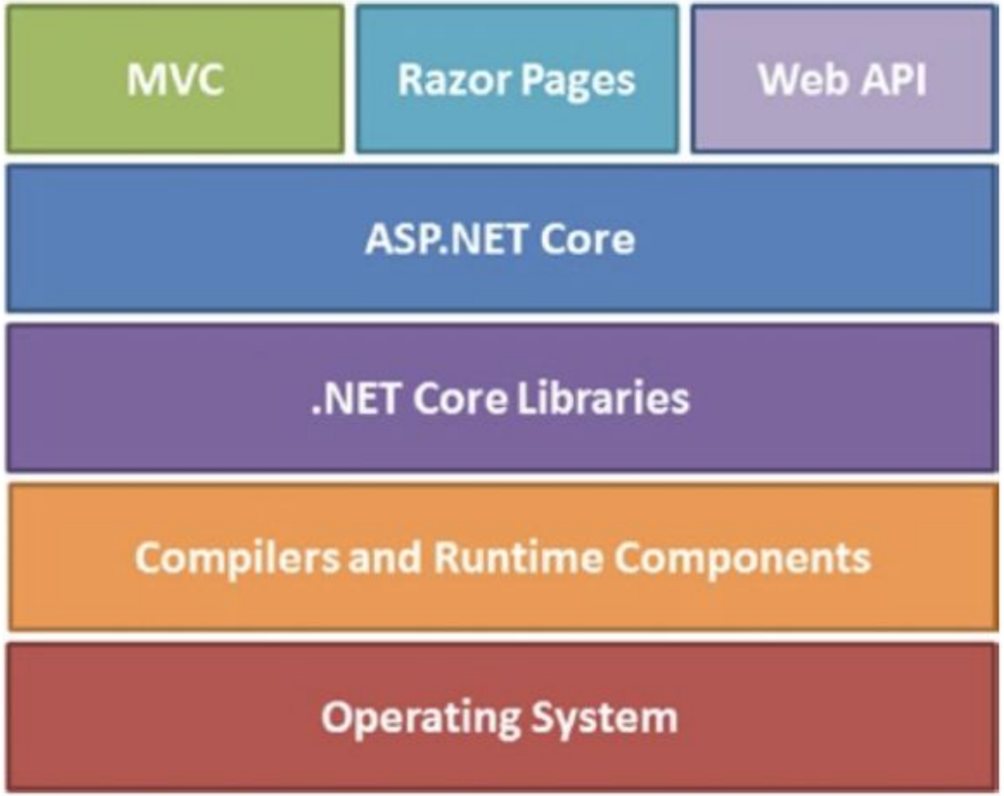
ASP .NET Core Basics

Overview of ASP.NET Core

- Framework for building modern web applications and services
- Cross-platform
  - Windows, Linux, or macOS
- Open source
- What does this mean?

Overview of ASP.NET Core

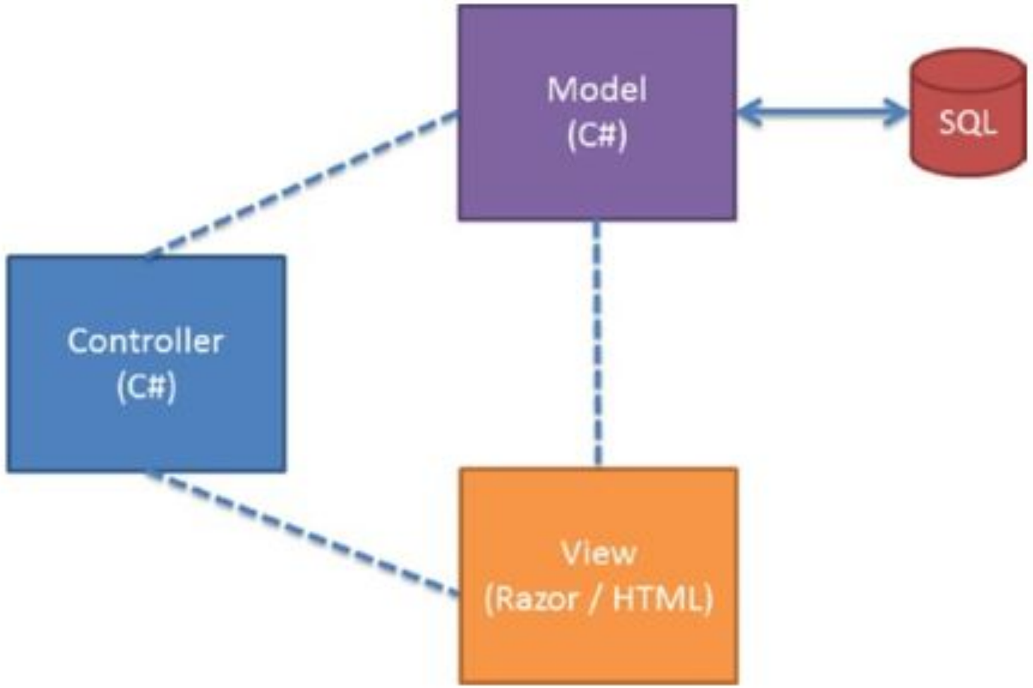
- ASP .NET core:
  - includes a built-in web server: Kestrel
  - Compatible with IIS, Nginx, or Apache
- Multiple development options for UI
  - MVC, Razor Pages, and Blazor
- Unified programming model for MVC web applications and Web APIs
- Built considering modularity
  - Nuget packages


MVC
Razor Pages
Web API
ASP.NET Core
.NET Core Libraries
Compilers and Runtime Components
Operating System

ASP.NET Core MVC

- Building web applications using the Model-View-Controller (MVC) pattern
- A good development choice when your program flow is complex and involves multiple models and views.
- Preferred development option for Ajax-based scenarios and Single Page Applications (SPAs).

ASP.NET Core MVC



Model

- Represents the application’s data and could be anything from a primitive type to a complex object

View

- Houses the application’s user interface (UI) and usually displays data held by the model
- Accepts user input and commands
- Exists as a .cshtml file and primarily contains markup and Razor code

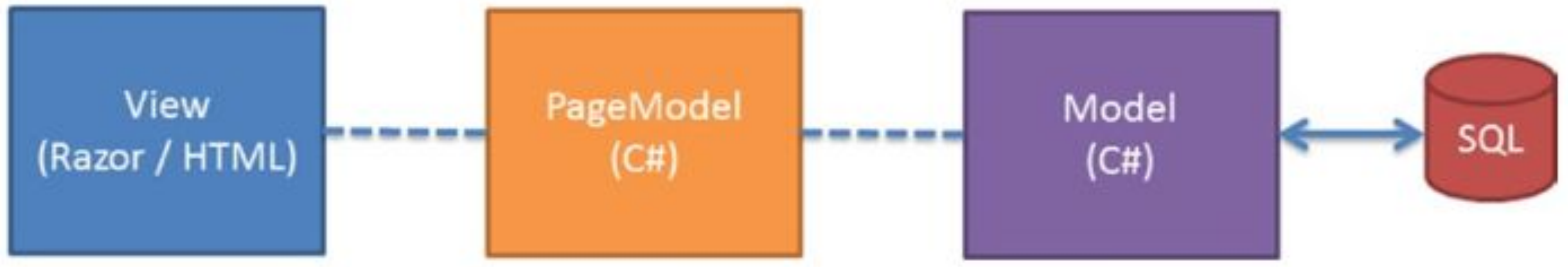
Controller

- Mediates between a model and a view
- Prepares the model required for a view and also to act upon the user input and commands as captured by the view
- Decides the program flow by deciding which model and view to send next to the user
- An end user never deals with a model and a view directly; rather, it invokes the controller.
- C# class and typically contains one or more methods called actions

ASP.NET Core Razor Pages

- Building web applications that use a Model-View-ViewModel (MVVM)-like pattern
- Preferred development option for page-focused scenarios with a simple program flow.
- The lack of a controller class also makes their code organization simpler.

ASP.NET Core Razor Pages



ASP.NET Core Razor Pages - PageModel

- Difference between MVC and MVVM is the absence of a separate controller class
- The ViewModel is closely related to a view and is responsible for view-specific things such as data binding, UI event handling, and UI notifications.
- It encapsulates view-specific data and behavior. It also updates the underlying data model whenever necessary.

Blazor

- Can we use C# on the server side as well as on the client side?
- Blazor is a framework for building rich and interactive client-side web user interfaces with ASP.NET Core
- UI components using C#, HTML, and CSS
- Depending on where the application code runs, Blazor offers two hosting models: client-side (Blazor WebAssembly) and server-side (Blazor Server).

Client Side (Blazor WebAssembly)

- WebAssembly (often abbreviated as Wasm) is a compact binary format that gives good performance (almost like a native execution) for web applications.
- Currently, all the major browsers support WebAssembly, so your application is not tied to a particular browser.
- Also, refer to WebAssembly

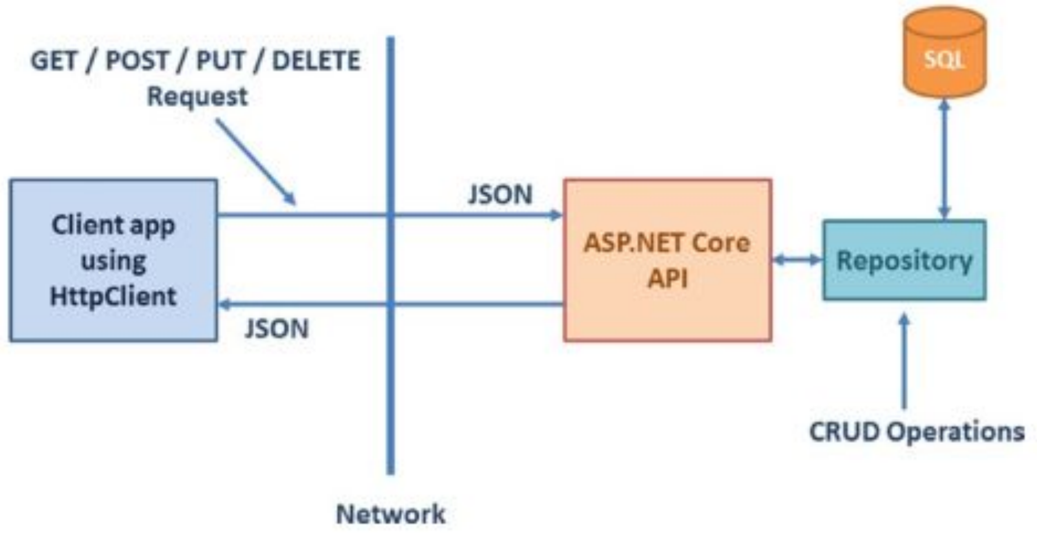
Client Side (Blazor WebAssembly)

- Deciding on the hosting model to use should be done prior to beginning your development
- Points to consider:
  - Good overall performance
  - Limitations of a browser
  - Longer initial load time
  - The browser needs to support WebAssembly

Server Side (Blazor Server)

- Blazor application is executed on a web server inside an ASP.NET Core application
- Changes to DOM and event handling are performed using a SignalR connection.
- Points to consider:
  - Can use ASP.NET Core capabilities
  - Every user interaction requires server communication
  - Faster initial load time
  - Not adequate to a very huge user base

ASP .NET Core Web API

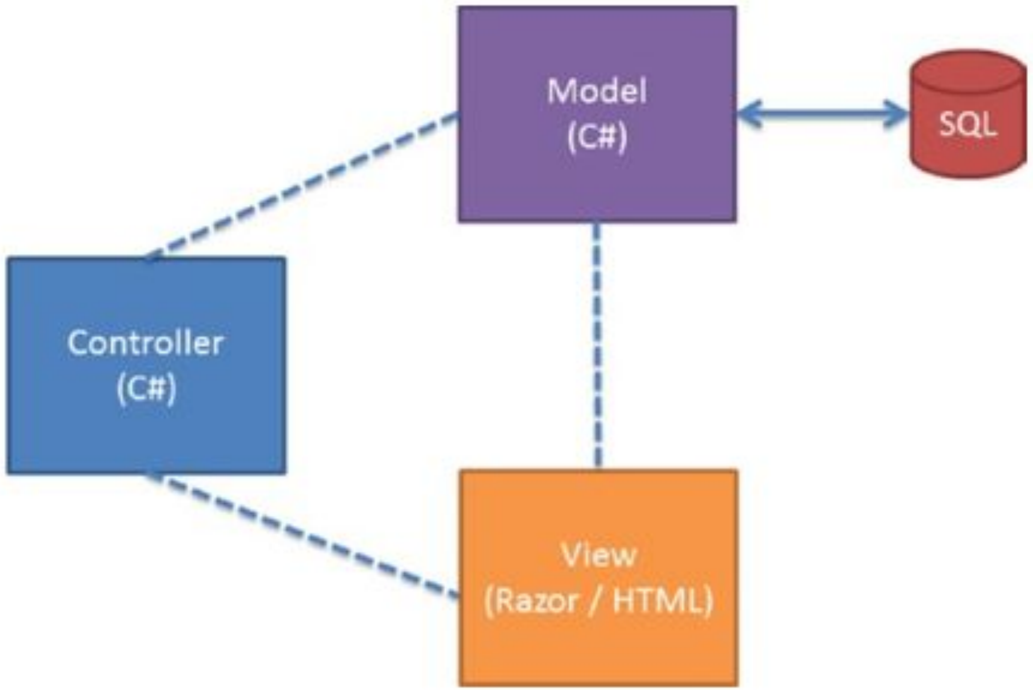


ASP .NET Core MVC

ASP.NET Core MVC

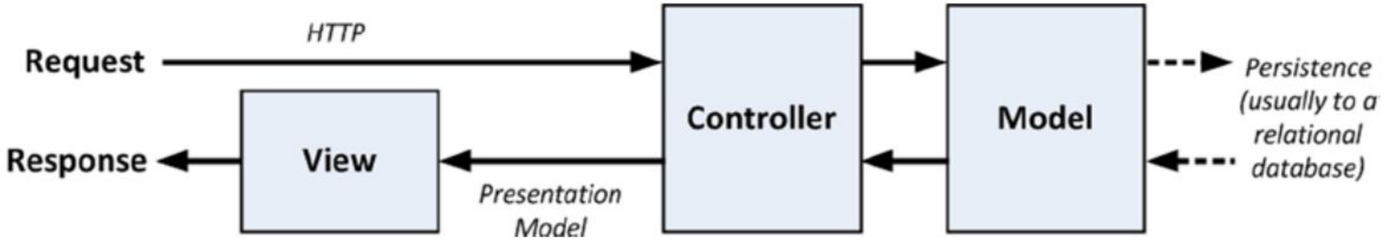
- Building web applications using the Model-View-Controller (MVC) pattern
- A good development choice when your program flow is complex and involves multiple models and views.
- Preferred development option for Ajax-based scenarios and Single Page Applications (SPAs).

ASP.NET Core MVC



The ASP.NET Implementation of MVC

- *Controllers* are C# classes, derived from the *System.Web.Mvc.Controller*
    - *Each public method in a class derived from Controller is an* ***action method****, which is associated with a configurable URL through the ASP.NET routing system.*



[Freeman, 2013]

Model

- Represents the application’s data and could be anything from a primitive type to a complex object

View

- Houses the application’s user interface (UI) and usually displays data held by the model
- Accepts user input and commands
- Exists as a .cshtml file and primarily contains markup and Razor code

Controller

- Mediates between a model and a view
- Prepares the model required for a view and also to act upon the user input and commands as captured by the view
- Decides the program flow by deciding which model and view to send next to the user
- An end user never deals with a model and a view directly; rather, it invokes the controller.
- C# class and typically contains one or more methods called actions

Debugging and Testing

Debugging MVC Applications

- Visual Studio prepares new projects for debugging automatically
  - Debug Attribute in the Web.config File
- Select Debug Configuration

Debugging MVC Applications

- A *breakpoint* is an instruction that tells the debugger to halt execution of the application and hand control to the programmer

- Local variables, scopes and execution

Getting Started with Automated Testing

- Web application developers today focus on two kinds of automated testing:
  - Unit Testing
    - *a way to specify and verify the behavior of individual classes (or other small units of code) in isolation from the rest of the application.*
  - Integration Testing
    - *a way to specify and verify the behavior of multiple components working together, up to and including the entire Web application*

Using TDD and the Red-Green-Refactor Workflow

- With test-driven development (TDD), you use unit tests to help design your code
    1. Determine that you need to add a new feature or method to your application
    2. Write the test that will validate the behavior of the new feature when it is written
    3. Run the test and get a red light
    4. Write the code that implements the new feature
    5. Run the test again and correct the code until you get a green light
    6. Refactor the code if required
    7. Run the test to confirm that your changes have not changed the behavior

Understanding Integration Testing

- Common approach: *UI automation*
  - simulating or automating a Web browser by reproducing the actions that a user would perform, e.g., pressing buttons, following links, and submitting forms

- Best-known open source browser options are:
  - Selenium RC (http://seleniumhq.org/)
  - WatiN (http://watin.org)

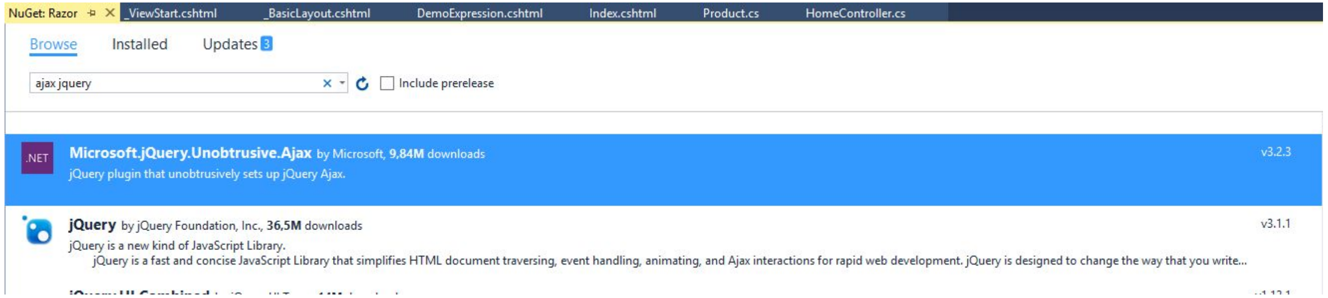
Understanding Integration Testing

- integration testing lets you create tests that are client-focused, recreating the actions of a user

- it can highlight problems that come from the interaction between components

URL and Ajax Helper Methods

Install NuGet Packages



ajax jquery

Microsoft.jQuery.Unobtrusive.Ajax by Microsoft, **9,84M** downloads v3.2.3
jQuery plugin that unobtrusively sets up jQuery Ajax.

jQuery by jQuery Foundation, Inc., **36,5M** downloads v3.1.1
jQuery is a new kind of JavaScript Library.
jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write...

Alternatively:

- Select **Package Manager Console** from the Visual Studio Tools ➤ Library Package Manager menu and enter the following command:
  - Install-Package jQuery –version 3.1.1
  - Install-Package Microsoft.jQuery.Unobtrusive.Ajax –version 3.2.3

Creating Basic Links and URLs

- One of the most fundamental tasks
- Can be done using HTML Helpers that render URLs

Creating Basic Links and URLs

Description	Example	Output
Application-relative URL	Url.Content("~/Content/Site.css")	/Content/Site.css
Link to named action or controller	Html.ActionLink("My Link", "Index", "Home")	<a href="/">My Link</a>
URL for action	Url.Action("GetPeople", "People")	/People/GetPeople
URL using route data	Url.RouteUrl(new {controller = "People", action="GetPeople"})	/People/GetPeople
Link using route data	Html.RouteLink("My Link", new {controller = "People", action="GetPeople"})	<a href="/People/GetPeople">My Link</a>
Link to named route	Html.RouteLink("My Link", "FormRoute", new {controller = "People", action="GetPeople"})	<a href="/app/forms/People/GetPeople"> My Link</a>

Using MVC Unobtrusive* Ajax

- Ajax (or, if you prefer, AJAX) is shorthand for *Asynchronous JavaScript and XML*
- Model for requesting data from the server in the background, without having to reload the Web page
- The MVC Framework contains built-in support for unobtrusive Ajax
  - Based on *jQuery*
  - Use helper methods to define your Ajax features, rather than having to add blocks of code throughout your views

** not conspicuous or attracting attention*

Preparing the Project for Unobtrusive Ajax

- Set up in two places in the application
    - Web.config file > configuration/appSettings set UnobtrusiveJavaScriptEnabled to true
    - Add references to the jQuery JavaScript libraries from previously added NuGet packages
        - <script src="~/Scripts/jquery-1.10.2.js"></script>
        - <script src="~/Scripts/jquery.unobtrusive-ajax.js"></script>

Creating the Ajax Form

```
@using (Html.BeginForm("<Action>", "<Controller>", FormMethod.Post))
```

Property	Description
Confirm	Sets a message to be displayed to the user in a confirmation window before making the Ajax request
HttpMethod	Sets the HTTP method that will be used to make the request—must be either Get or Post
InsertionMode	Specifies the way in which the content retrieved from the server is inserted into the HTML. The three choices are expressed as values from the InsertionMode enum: InsertAfter, InsertBefore and Replace (which is the default).
LoadingElementId	Specifies the ID of an HTML element that will be displayed while the Ajax request is being performed
LoadingElementDuration	Specifies the duration of the animation used to reveal the element specified by LoadingElementId
UpdateTargetId	Sets the ID of the HTML element into which the content retrieved from the server will be inserted
Url	Sets the URL that will be requested from the server

Working with Ajax Callbacks

- The AjaxOptions class defines a set of properties that specify JavaScript functions that will be called at various points in the Ajax request lifecycle

Property	jQuery Event	Description
OnBegin	beforeSend	Called immediately prior to the request being sent
OnComplete	complete	Called if the request is successful
OnFailure	error	Called if the request fails
OnSuccess	success	Called when the request has completed, irrespective of whether the request succeeded or failed

Model Binding and Validation

- *Server and Client-side validations*

Model Binding

- Model binding is the process of creating .NET objects using the data sent by the browser in an HTTP request

- Model binding is an elegant bridge between the HTTP request and the C# methods that define actions

Model Validation

- Users will often enter data that isn’t valid and cannot be used
- Model validation is the process of ensuring the data received by the application is suitable for binding to the model
  - when this is not the case, provide useful information to the user that will help explain the problem

Model Validation

```
namespace ModelValidation.Models {
    public class Appointment {

        public string ClientName { get; set; }

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

        public bool TermsAccepted { get; set; }
    }
}
```

```
[HttpPost]
public ViewResult MakeBooking(Appointment appt) {

    if (string.IsNullOrEmpty(appt.ClientName)) {
        ModelState.AddModelError("ClientName", "Please enter your name");
    }

    if (ModelState.IsValidField("Date") && DateTime.Now > appt.Date) {
        ModelState.AddModelError("Date", "Please enter a date in the future");
    }

    if (!appt.TermsAccepted) {
        ModelState.AddModelError("TermsAccepted", "You must accept the terms");
    }

    if (ModelState.IsValid) {
        // statements to store new Appointment in a
        // repository would go here in a real project
        return View("Completed", appt);
```

Model Validation

```
namespace ModelValidation.Models {
    public class Appointment {

        [Required]
        public string ClientName { get; set; }

        [DataType(DataType.Date)]
        [Required(ErrorMessage="Please enter a date")]
        public DateTime Date { get; set; }

        [Range(typeof(bool), "true", "true", ErrorMessage = "You must accept the terms")]
        public bool TermsAccepted { get; set; }
    }
}
```

Displaying Validation Messages

Overloaded Method	Description
Html.ValidationSummary()	Generates a summary for all validation errors
Html.ValidationSummary(bool)	If the bool parameter is true, then only model-level errors are displayed (see the explanation after the table). If the parameter is false, then all errors are shown
Html.ValidationSummary(string)	Displays a message (contained in the string parameter) before a summary of all the validation errors
Html.ValidationSummary(bool, string)	Displays a message before the validation errors. If the bool parameter is true, only model-level errors will be shown

The Built-in Validation Attributes

Attribute	Example	Description
Compare	[Compare("MyOtherProperty")]	Two properties must have the same value. This is useful when you ask the user to provide the same information twice, such as an e-mail address or a password.
Range	[Range(10, 20)]	A numeric value (or any property type that implement IComparable) must not lie beyond the specified minimum and maximum values. To specify a boundary on only one side, use a MinValue or MaxValue constant—for example, [Range(int.MinValue, 50)].
RegularExpression	[RegularExpression("pattern")]	A string value must match the specified regular expression pattern. Note that the pattern has to match the entire user-supplied value, not just a substring within it. By default, it matches case sensitively, but you can make it case insensitive by applying the (?i) modifier—that is, [RegularExpression("(?i)mypattern")].
Required	[Required]	The value must not be empty or be a string consisting only of spaces. If you want to treat whitespace as valid, use[Required(AllowEmptyStrings = true)].
StringLength	[StringLength(10)]	A string value must not be longer than the specified maximum length. You can also specify a minimum length:[StringLength(10, MinimumLength=2)].

Using Client-Side Validation

- Web.config
  - <add key="ClientValidationEnabled" value="true" />
- Adding the NuGet Packages
  - jQuery, jQuery.Validation, Microsoft.jQuery.Unobtrusive.Validation
- Add script elements to _Layout:
  - <script src="~/Scripts/jquery-1.10.2.js"></script>
  - <script src="~/Scripts/jquery.validate.js"></script>
  - <script src="~/Scripts/jquery.validate.unobtrusive.js"></script>

Deployment

- *Windows Azure*

What is the Azure SDK for .NET?

- Set of Visual Studio tools, command-line tools, runtime binaries, and client libraries that help you develop, test, and deploy apps that run in Azure

- Azure Web Apps provides a highly scalable, self-patching web hosting service.

https://docs.microsoft.com/en-us/azure/dotnet-sdk

How to ?

- Publish to Azure with SQL Database
    - In the **Solution Explorer,** right-click your **project** and select **Publish**
    - Make sure that Microsoft Azure App Service is selected and click **Publish**

- Sign in to Azure
    - In the Create App Service dialog, click Add an account, and then sign in to your Azure subscription
    - Configure the web app name
    - Create a resource group
    - Create an App Service plan
    - Create a SQL Server instance

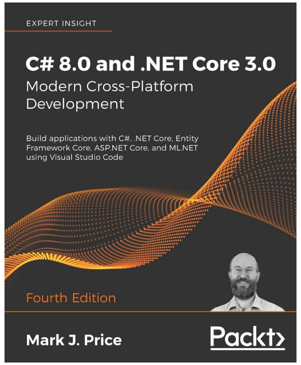
https://docs.microsoft.com/en-us/azure/app-service-web/app-service-web-tutorial-dotnet-sqldatabase

Azure App Service

- Azure App Service is an Azure service that works on a platform as a service (PaaS) model
- It can run and manage web applications, mobile applications, APIs, and business logic applications.
- Managed hosting environment for your ASP.NET Core web applications

References

- Price, Mark J. "*C# 8.0 AND .NET CORE 3.0: Modern Cross-Platform Development*", PACKT Publishing, 2019
- B. Joshi, Beginning Database Programming Using ASP.NET Core 3, Apress, 2019
- [ASP.NET documentation]()
- https://github.com/dotnet/aspnetcore



EXPERT INSIGHT

C# 8.0 and .NET Core 3.0

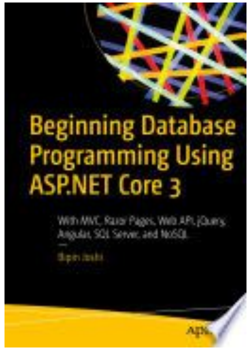
Modern Cross-Platform Development

Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code

Fourth Edition

Mark J. Price

Packt>



Beginning Database Programming Using ASP.NET Core 3

With MVC, Razor Pages, Web API, jQuery, Angular, SQL Server, and NoSQL

Bipin Joshi