Appendix A

Answers to the Test Your Knowledge Questions

This appendix has the answers to the questions in the *Test your knowledge* section at the end of each chapter.

Chapter 1 – Introducing Real-World Web Development Using .NET

Exercise 1.3 – Test your knowledge

Suggested answers to these questions:

1. What was the name of Microsoft's first dynamic server-side-executed web page technology, and why is it still useful to know this history today?

   Answer: Active Server Pages (**ASP**). The name ASP.NET Core derives from ASP, and it is still used in new features like Tag Helpers, as shown in the following markup:

   ```
   <a asp-controller="Home" asp-action="Index">Home</a>
   ```

2. What are the names of two Microsoft web servers?

   Answer: Kestrel (cross-platform) and **Internet Information Services** (**IIS**), which is Windows-only.

3. What are some differences between a microservice and a nanoservice?

   Answer: A microservice implements more than one function grouped into a small domain and is always running. A nanoservice implements a single function and is not always running, i.e., it can be "serverless."

4. What is Blazor?

   Answer: Blazor is a .NET-based technology for implementing web user interfaces. It is designed to be used instead of JavaScript-based **single-page applications** (**SPAs**) like React, Angular, and Vue.

5. What was the first version of ASP.NET Core that could not be hosted on .NET Framework?

   Answer: ASP.NET Core 3.0 requires .NET Standard 2.1, which is not supported by .NET Framework.

6. What is a user agent?

   Answer: A user agent is a client to a web server, for example, a web browser or a search engine web crawler.

7. What impact does the HTTP request-response communication model have on web developers?

   Answer: Website dynamic code resides on and executes on a web server. The server code cannot trigger communication. A web browser must make an HTTP request to trigger code on the server, which can then generate an HTTP response. This makes updating a web page difficult because it is under the control of the client, not the server, when requests for more data are made.

8. Name and describe four components of a URL.

   Answer: The *scheme* determines if you are using HTTP or HTTPS. The *domain* is the unique address of the computer (or a web farm of servers acting as a single logical computer). The *port number* is the port on which the server(s) listen (usually `80` for HTTP and `443` for HTTPS). The *path* is the relative path to a resource like a folder or file. The *query string* is for optional parameters passed along with the request. The *fragment* is an identified element within a resource, like a web page.

9. What capabilities does Developer Tools give you?

   Answer: Developer Tools allows you to see every HTTP request and response, allows you to view client-side application data like cookies, sessions, and local storage, provides a console for logging, and so on.

10. What are the three main client-side web development technologies, and what do they do?

    Answer: HTML5 (structure), CSS3 (styles), and JavaScript (execute actions).

Know your webbreviations

What do the following web abbreviations stand for, and what do they do?

1. **URI (Uniform Resource Identifier)** is a unique sequence of characters that identifies a resource.
2. **URL (Uniform Resource Locator)** is the address of a unique resource.

3. **WCF (Windows Communication Foundation)** is a framework for building service-oriented applications. It is part of .NET Framework, and an open-source implementation named WCF Core is available for modern .NET.
4. **TLD (Top-Level Domain)** is one of the domains at the highest level in the hierarchical **DNS (Domain Name System)** of the internet, for example, `packt.com`.
5. **API (Application Programming Interface)** is a mechanism for a computer system to allow other computer systems to communicate with it. APIs should be well-documented.
6. **SPA (Single-Page Application)** is a web app that loads only a single web page and then updates the content dynamically via JavaScript APIs when needed. SPA frameworks include Angular, React, Vue, and Blazor.
7. **CMS (Content Management System)** is a software application that allows users to build and manage a website without having to write code.
8. **Wasm (WebAssembly)** is a binary instruction format and is designed as a compilation target for programming languages like C#, for deployment on the web, client, and server.
9. **SASS (Syntactically Awesome Style Sheets)** is a CSS preprocessor. SASS has features that don't exist in CSS yet, like nesting, mixins, and inheritance, that help you write maintainable CSS.
10. **REST (REpresentational State Transfer)** is an architectural style for distributed hypermedia systems. Roy Fielding presented it in 2000 in his famous dissertation.

In case you missed it (ICYMI): Acronyms and initialisms are types of abbreviation. Learn more at the following link: `https://www.rd.com/article/acronym-vs-abbreviation-whats-the-difference/`.

Chapter 2 – Building Websites Using ASP.NET Core MVC

Exercise 2.3 – Test your knowledge

Suggested answers to these questions:

1. What do the files with the special names `_ViewStart` and `_ViewImports` do when created in the `Views` folder?

   Answer:

   - A `_ViewStart` file contains a block of statements that are executed when the `View` method is executed, when a controller action method passes a model to a view, for example, to set a default layout.
   - A `_ViewImports` file contains `@using` statements to import namespaces for all views, to avoid having to add the same import statements at the top of all views.

2. What are the names of the three segments defined in the default ASP.NET Core MVC route, what do they represent, and which are optional?

   Answer:

   - `{controller}`: For example, `/shippers` represents a controller class to instantiate, for example, `ShippersController`. It is optional because it can use the default value: `Home`.
   - `{action}`: For example, `/privacy` represents an action method to execute, for example, `Privacy`. It is optional because it can use the default value: `Index`.
   - `{id}`: For example, `/5` represents a parameter in the action method, for example, `int id`. It is optional because it is suffixed with `?`.

3. What does `UseMigrationsEndPoint` do?

   Answer: In ASP.NET Core, the `UseMigrationsEndPoint` extension method is part of the EF Core tooling. It enables a specialized endpoint that provides a web interface to trigger and manage database migrations directly from a deployed ASP.NET Core application. Primarily, this method is intended for use in development environments. It allows for testing migrations without needing to use command-line tools (like `dotnet ef database update`), streamlining the process by providing a web interface accessible through the development server. Rather than running CLI commands, you can visit the migrations endpoint (at `/ApplyDatabaseMigrations` if using default settings) in your browser to apply the latest migration directly to your database.

4. In a shared layout file like `_Layout.cshtml`, how do you output the content of the current view?

   Answer: To output the content of the current view in a shared layout, call the `RenderBody` method, as shown in the following markup:

   ```
   @RenderBody()
   ```

5. In a shared layout file like `_Layout.cshtml`, how do you output a section that the current view can supply content for, and how does the view supply the contents for that section?

   Answer: To output the content of a section in a shared layout, call the `RenderSection` method, specifying a name for the section if it is required, as shown in the following markup:

   ```
   @RenderSection("Scripts", required: false)
   ```

   To define the contents of the section in the view, create a named section, as shown in the following markup:

   ```
   @section Scripts
   {
   <script>
     alert('Hello, Mr. Page!');
   </script>
   }
   ```

6. When calling the `View` method inside a controller's action method, what paths are searched for the view by convention?

   Answer: When calling the `View` method inside a controller's action method, three paths are searched for the view by default, based on combinations of the names of the controller and action method and a special `Shared` folder, as shown in the following example output:

   ```
   InvalidOperationException: The view 'Index' was not found. The following
   locations were searched:
   /Views/Home/Index.cshtml
   /Views/Shared/Index.cshtml
   /Pages/Shared/Index.cshtml
   ```

   This can be generalized as follows:

   - `/Views/[controller]/[action].cshtml`
   - `/Views/Shared/[action].cshtml`
   - `/Pages/Shared/[action].cshtml`

7. How does cache busting with Tag Helpers work?

   Answer: When `asp-append-version` is specified with a `true` value in a `<link>`, `<img>`, or `<script>` element, the Tag Helper for that tag type is invoked. They bust the cached version of whatever is referenced by the link, image, or script by automatically appending a query string value, named `v`, that is generated from a SHA256 hash of the referenced source file.

8. Why might you enable Razor Pages even if you are not creating any yourself?

   Answer: If you have used features like ASP.NET Core Identity UI, then they require Razor Pages.

9. How does ASP.NET Core MVC identify classes that can act as controllers?

   Answer: ASP.NET Core MVC identifies classes that can act as controllers by looking to see if the class (or a class that it derives from) is decorated with the `[Controller]` attribute.

10. In what ways does ASP.NET Core MVC make it easier to test a website?

    Answer: The **Model-View-Controller** (**MVC**) design pattern separates the technical concerns. These consist of:

    - The shape of the data (model)
    - The executable statements to process the incoming request and outgoing response (controller)
    - The generation of the response in a format requested by the user agent like HTML or JSON (view)

    This makes it easier to write unit tests. ASP.NET Core also makes it easy to implement the **Inversion-of-Control** (**IoC**) and **dependency injection** (**DI**) design patterns to remove dependencies when testing a component like a controller.

Chapter 3 – Model Binding, Validation, and Data Using EF Core

Exercise 3.3 – Test your knowledge

1. What does the default model binder do, and what data types can it handle?

   Answer: The default model binder sets parameters in the action method. It can handle the following data types:

   - Simple types like `int`, `string`, and `DateTime`, including nullable types
   - Complex types like `Person` and `Product`
   - Collection types like `ICollection<T>` or `IList<T>`

2. How do you specify validation rules for ASP.NET Core model binding?

   Answer: ASP.NET Core uses data annotations and custom validation attributes on model properties to specify validation rules. For example, `[Required]`, `[StringLength(100)]`, and `[Range(18, 99)]` are common data annotations that enforce validation rules like non-nullability, maximum string length, and numeric ranges, respectively.

3. Where can you find any validation errors caused during model binding?

   Answer: When a model is bound, ASP.NET Core automatically checks if it meets the specified validation rules. If the model is invalid, the framework adds error messages to the `ModelState`, which can then be checked in the controller to determine how to proceed.

4. What is over-posting, aka a mass assignment attack?

   Answer: Over-posting, also known as mass assignment, is a vulnerability where an attacker sends unexpected data in an HTTP request that might map to properties of your model that were not meant to be exposed or modified. This can happen when an attacker manipulates form fields or query parameters to include additional data not meant to be bound.

5. When should you return a `401 Unauthorized` status code, and when should you return a `403 Forbidden` status code?

   Answer: A `401 Unauthorized` status code indicates that the client needs to authenticate to access the resource. It's typically used when an authentication mechanism is required but not provided or invalid. A `403 Forbidden` status code is used when the client is authenticated but does not have the necessary permissions to access the resource.

6. To allow a user to insert an entity like a `Customer`, why do you typically define a pair of action methods, `GET` and `POST`, and what do they do?

   Answer: First, create a `GET` action method and view to show a web page containing a form where the user can enter details for a new customer, and then a button to submit the data. Second, create a `POST` action method to perform the insert operation, and if successful, redirect to the page of customers to review the inserted entity, or, if an error occurs, then back to the new customer page to try again. (You would also use a combination of `GET` and `POST` to perform update or delete operations.)

7. How does a CSRF attack work?

   Answer: A CSRF attack tricks an authenticated user into performing actions on a site without their knowledge. For instance, if a user is logged in to their bank's website and unknowingly clicks on a malicious link, a request might be sent to the bank, performing an action like transferring funds. Since the request comes from the user's authenticated session, the server processes it, thinking it is legitimate.

8. What should you do to prevent CSRF attacks in an ASP.NET Core project?

   Answer: Use the `@Html.AntiForgeryToken()` method to generate a pair of tokens. One token is stored in an HTTP cookie (on the client side, named `.AspNetCore.AntiForgery`). This cookie is tied to the user's session and is sent along with every HTTP request made to the server. Another token is generated and inserted into the form as a hidden field. This token is embedded within the HTML of the form and gets sent to the server when the form is submitted. Both tokens are cryptographically linked and validated against each other on the server. If the tokens do not match, the request is rejected, preventing CSRF attacks.

9. What are two ways to get data into a view?

   Answer: Define a model to pass data directly into a view, or store data in the `ViewData` aka `ViewBag` dictionary.

10. How do you make an action method asynchronous?

    Answer: Decorate the action method with the `async` keyword. Change the return type to `Task<T>`. Call asynchronous methods like `ToListAsync` or `SingleOrDefaultAsync` in the method implementation.

Chapter 4 – Building and Localizing Web User Interfaces

Exercise 4.3 – Test your knowledge

Suggested answers to these questions:

1. What is the advantage of declaring a strongly typed Razor View, and how do you do it?

   Answer: The advantage of declaring a strongly typed Razor View is IntelliSense, which will help you write the markup and code, and compiler warnings and errors at design time. If you do not specify a model type, then the Razor View assumes `dynamic` for the type, the use of `Model` cannot show IntelliSense, and errors will not occur until runtime. You specify a model type by adding a `@model` directive with the type you want to use at the top of the Razor View. In the markup, use `Model` to represent the object. For example:

   ```
   @model Customer
   ...
   <h1>@Model.CompanyName</h1>
   ...
   ```

2. How do you enable Tag Helpers in a view?

   Answer: At the top of the Razor View, or in `Views\_ViewImports.cshtml`, add the following statement:

   ```
   @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
   ```

3. What are the pros and cons of HTML Helper methods compared to Tag Helpers?

   Answer: Tag Helpers are generally easier to work with so should be used in most cases, but there are notable situations where Tag Helpers cannot be used, like in Razor components. In those cases, HTML Helper methods should be used instead. HTML Helper methods can do everything you need but require messier code.

4. How can a browser request a preferred language for localization?

   Answer: A browser can request what culture it prefers by adding a query string parameter (for example, `?culture=en-US&ui-culture=en-US`), sending a cookie (for example, `c=en-US|uic=en-US`), or setting an HTTP header (for example, `Accept-Language: en-US,en;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-GB;q=0.6`).

5. How do you localize text in a view?

   Answer: Create resource files (`*.resx`) for each language with the text values that need to be localized. Inject the registered dependency service for `IViewLocalizer` into the Razor View. Retrieve each text value using the unique key that you assigned to it. In `Program.cs`, call the `AddLocalization` method, and specify the folder that contains the `.resx` files. After the call to `AddControllersWithViews`, call the `AddViewLocalization` method. Call the `UseRequestLocalization` method while building the HTTP pipeline with an array of ISO culture codes that you want to support.

6. What is the prefix for attributes recognized by Tag Helpers?

   Answer: `asp-`. For example, `asp-action` and `asp-controller` in an anchor tag element:

   ```
   <a asp-action="Privacy" asp-controller="Home">View our privacy policy.</
   a>
   ```

7. How can you pass a complex object as a query string parameter?

   Answer: Create a method that returns a dictionary with string values for both the key and value. Call the method to set the `asp-all-route-data` attribute of an anchor tag element, as shown in the following markup:

   ```
   <a asp-controller="Home" asp-action="Shipper"
      asp-all-route-data="await GetShipperData()"
      class="btn btn-outline-primary">Shipper</a>
   ```

8. How can you control how long the contents of the `<cache>` element are cached for?

   Answer: The `<cache>` element has the following attributes that control how long the contents are cached for:

   - `expires-after`: A `TimeSpan` value to expire after. The default is `00:20:00`, meaning 20 minutes.
   - `expires-on`: A `DateTimeOffset` value to expire at. No default.
   - `expires-sliding`: A `TimeSpan` value to expire after if the value has not been accessed during that time. No default.

9. What is the `<environment>` element used for?

   Answer: The Environment Tag Helper renders its content only if the current environment matches one of the values in a comma-separated list of names. This allows you to only show content that is needed during development, or only show content on the production website.

10. What is Bootstrap, and what is a good reason to use it?

    Answer: Bootstrap is the world's most popular framework for building responsive, mobile-first websites. It combines CSS stylesheets with JavaScript libraries to implement its functionality. It is a good choice for prototyping a website UI, although, before going public, you might want to hire a web designer to build a custom Bootstrap theme or replace it with a completely custom set of CSS stylesheets to give your website a distinct brand.

Chapter 5 – Authentication and Authorization

Exercise 5.3 – Test your knowledge

Suggested answers to these questions:

1. How many characters long should a password be, and should it contain special characters?

   Answer: SHALL require passwords to be a minimum of eight characters in length. SHOULD require passwords to be a minimum of 15 characters in length. SHALL NOT impose composition rules like requiring mixtures of different character types.

2. How frequently should passwords be changed?

   Answer: SHALL NOT require users to change passwords periodically. SHALL force a password change if there is evidence of compromise of the authenticator.

3. What are three authentication schemes?

   Answer: Three authentication schemes are: cookies, JWT, and OAuth.

4. Which external authentication providers are supported by ASP.NET Core Identity?

   Answer: ASP.NET Core Identity supports external authentication providers like Google, Facebook, and Microsoft.

5. How can you significantly reduce the risk of XSS attacks being able to steal session cookies?

   Answer: Make the cookie unreadable to JavaScript, only send it over HTTPS, scope it tightly, and make it hard to abuse even if an attacker can run the script. In ASP.NET Core that means using `HttpOnly`, `Secure`, strict `SameSite` where possible, the `__Host-` prefix, short lifetimes, and a solid CSP.

6. What are some of the security vulnerabilities of cookies?

   Answer: XSS, CSRF, cookie theft, session hijacking, and persistent cookies.

7. Why might you enable Razor Pages even if you are not creating any yourself?

   Answer: The `Microsoft.AspNetCore.Identity.UI` package implements its user registration, login, and profile user interface using Razor Pages.

8. How can you customize the user interface provided by ASP.NET Core Identity, for example, the login form?

   Answer: The official documentation shows how to do this at the following link: `https://learn.microsoft.com/en-us/aspnet/core/security/authentication/scaffold-identity#create-full-identity-ui-source`.

9. What dependency service should you use to programmatically register a visitor to your website?

   Answer: The `UserManager<IdentityUser>` instance and its `CreateAsync` method will register a user account for a visitor.

10. What are the five types of ASP.NET Core filters, and in what order do they execute?

    Answer: The order of filter execution is determined by its type, as described in the following list:

    1. **Authorization Filters**: A class that implements `IAuthorizationFilter`
    2. **Resource Filters:** A class that implements `IResourceFilter`
    3. **Action Filters**: A class that implements `IActionFilter`
    4. **Exception Filters:** A class that implements `IExceptionFilter`
    5. **Result Filters**: A class that implements `IResultFilter`

Chapter 6 – Performance and Scalability Optimization Using Caching

Exercise 6.3 – Test your knowledge

Suggested answers to these questions:

1. How can you instruct the visitor's browser to cache the response for 24 hours?

   Answer: To instruct the visitor's browser to cache the response for 24 hours, decorate the controller class or action method with the `[ResponseCache]` attribute, and set the `Duration` parameter to `86400` seconds and the `Location` parameter to `ResponseCacheLocation.Client`.

2. What is the difference between the `max-age=<seconds>` and `s-maxage=<seconds>` directives?

   Answer: `max-age=<seconds>`: This defines the maximum amount of time (in seconds) that a resource is considered fresh. Once the `max-age` has passed, the client will revalidate the resource with the server. For example, `Cache-Control: max-age=3600` says that the resource will be considered fresh for one hour (3,600 seconds). `s-maxage=<seconds>`: Similar to `max-age`, but applies only to shared caches, like proxies or CDNs. If both are specified, this overrides `max-age` for these caches. For example, this option is useful for having different caching rules for public CDNs compared to browsers.

3. When is it important to use the `NoStore` parameter on the `[ResponseCache]` attribute?

   Answer: This completely prevents the client or any intermediary from caching the response. It will not store it at all. This is good practice for resources that include sensitive data, like bank account information, where you don't want it to be stored anywhere.

4. How do you set the default expiration time for output caching?

   Answer: When you call `AddOutputCache` to add output caching to the dependency services collection, configure options including the default expiration time span, as shown in the following code:

   ```
   builder.Services.AddOutputCache(options =>
   {
     options.DefaultExpirationTimeSpan =
       TimeSpan.FromSeconds(DurationInSeconds.TenSeconds);
   });
   ```

5. How do you enable output caching for an endpoint defined using `MapGet`?

   Answer: After the call to `MapGet`, call the `CacheOutput` method, as shown in the following code:

   ```
   app.MapGet("/cached", () => DateTime.Now.ToString())
     .CacheOutput();
   ```

6. Why might you need to set both a sliding and an absolute expiration for object caching?

   Answer: If you only set a sliding expiration, an object may stay in the cache forever, so you might also want to set the `AbsoluteExpirationRelativeToNow` property to a `TimeSpan` further in the future, after which the object would definitely be evicted. Choose this if you want the "best of both worlds."

7. What are some benefits of distributed object caching compared to in-memory object caching?

   Answer: Objects stored in a distributed cache:

   - Are consistent across requests to multiple servers.
   - Survive server restarts and service deployments.
   - Do not waste local server memory.
   - Are stored in a shared area; so, in a server farm scenario with multiple servers, you do not need to enable sticky sessions.

8. What is hybrid object caching?

   Answer: The HybridCache API addresses some limitations found in the `IDistributedCache` and `IMemoryCache` APIs. `HybridCache` efficiently manages most tasks related to storing and retrieving data from the cache. It provides a single interface for both in-process and out-of-process caching. `HybridCache` can seamlessly replace any existing `IDistributedCache` and `IMemoryCache` usage. It always uses the in-memory cache initially, and when an `IDistributedCache` implementation is available, `HybridCache` leverages it for secondary caching.

9. What are some non-caching techniques to improve the scalability of a website?

   Answer: Asynchronous programming, load balancing, and database optimization all contribute to making an ASP.NET Core application more scalable. Combining these strategies with cloud-native tools, like auto-scaling and CDNs, ensures your application can grow with demand while maintaining high performance.

10. What is a CDN, and what are its benefits for a website?

    Answer: A **Content Delivery Network** (**CDN**) will offload static content like images, CSS, and JavaScript libraries to edge servers that are geographically closer to users. This reduces the load on your web server and improves performance for end users.

Chapter 7 – Web User Interface Testing Using Playwright

Exercise 7.3 – Test your knowledge

Suggested answers to these questions:

1. What browsers does Playwright use when running its tests?

   Answer: Playwright uses open-source Chromium builds. Playwright can operate against branded browsers available on your computer. In particular, the current Playwright version will support Stable and Beta channels of these browsers. Playwright's Firefox version uses the most recent Firefox Stable build. Playwright's WebKit version uses the most recent WebKit trunk build, before it is used in Apple Safari and other WebKit-based browsers. Playwright doesn't work with the branded version of Firefox or Safari since they rely on patches.

2. What are the main interfaces that represent important objects when writing tests for Playwright?

   Answer: The main interfaces that represent important objects when writing tests for Playwright are: `IPlaywright`, `IBrowser`, `IBrowserContext`, `IResponse`, `IPage`, and `ILocator`.

3. Using Playwright, what are some methods to get one or more elements on a web page?

   Answer: Playwright provides some useful methods to get one or more elements on a web page, including: `GetByRole`, `GetByLabel`, `GetByPlaceholder`, `GetByTestId`, `GetByText`, `GetByTitle`, and `GetByAltText`.

4. What happens if more than one element matches and you call a method that implies a single DOM element, like `ClickAsync`?

   Answer: All operations on locators that imply a single DOM element will throw an exception if more than one element matches. For example, if you have a locator that matches all buttons and call `ClickAsync`, then an exception is thrown.

5. What does the Playwright Inspector do?

   Answer: The Playwright Inspector tool allows you to create comprehensive test scripts in a fraction of the time it would take to write them manually. It captures user interactions with high precision, reducing the chance of errors that can occur when writing tests by hand.

Chapter 8 – Configuring and Containerizing ASP.NET Core Projects

Exercise 8.3 – Test your knowledge

1. What is the main idea of DI?

   Answer: The main idea of dependency injection is to decouple the creation of an object's dependencies from its own behavior, which allows for more modular, testable, and maintainable code. Instead of objects creating dependencies themselves, they are injected with their dependencies at runtime, often by an external framework or container.

2. In .NET, you can register dependency services with different lifetimes. What are they?

   Answer: The dependency service lifetimes are:

   - **Transient**: These services are created each time they're requested. Transient services should be lightweight and stateless.
   - **Scoped**: These services are created once per client request and are disposed of when the response returns to the client.
   - **Singleton**: These services are usually created the first time they are requested and then shared, although you can provide an instance at the time of registration too.

3. What makes containers lightweight compared to virtual machines?

   Answer: Containers run on a single machine's OS kernel and share that kernel with other containers. They're lightweight because they don't need the extra load of a hypervisor that manages virtual machines. Containers run directly within the host machine's kernel. This makes them more efficient, faster, and less resource-intensive than traditional VMs that require a full-blown operating system for each VM.

4. What is the relationship between a Docker registry, a Docker image, and a Docker container?

   Answer: A Docker image is a static snapshot of a Docker container's configuration. This image includes everything needed to run an application – the code, a runtime, libraries, environment variables, and configuration files – all packaged together. Images are used to create Docker containers.

   A Docker container is a runtime instance of a Docker image. When you run an image, Docker creates a container from that image. Containers are isolated environments that contain their own filesystems, which are provided by the image. A container runs an application as defined by the image and can be started, stopped, moved, and deleted.

   A Docker registry is a storage and content delivery system, holding named Docker images, available in different tagged versions. Users interact with a registry by using Docker push and pull commands to upload and download images. Docker Hub is a popular public registry that anyone can use, and there are other private registry options for storing images, such as those provided by Amazon **Elastic Container Registry** (**ECR**) or **Azure Container Registry** (**ACR**).

5. In a `Dockerfile`, how do you specify the base image?

   Answer: The `FROM` instruction starts with a base image containing the .NET runtime, as shown in the following code:

   ```
   FROM mcr.microsoft.com/dotnet/aspnet:10.0
   ```

6. What are the three main types of Aspire resource?

   Answer: The built-in types of resource are:

   - **Project:** A .NET project. For example, an ASP.NET Core web service or website.
   - **Container:** A container image. For example, a Docker image of Redis or RabbitMQ.
   - **Executable:** An executable file.

7. What role do the `AppHost` project and the `ServiceDefaults` project play in an Aspire solution?

   Answer: The `AppHost` project is a console app that starts all the other projects and ensures the correct configuration for all resources and endpoints. The `AppHost` project defines resources that make up the application, including .NET projects, containers, executables, and cloud resources. The `AppHost` project maps how the different resources communicate with each other.

   The `ServiceDefaults` project is a class library that centralizes the configuration of all the Aspire resources, including components like databases and .NET projects.

8. What are the benefits of referencing an Aspire component package instead of the usual package for a component like Redis?

   Answer: Aspire components are wrapper class libraries that configure a feature like Redis to operate well in a cloud-native environment. Aspire components are designed to solve the biggest roadblock to getting started with cloud-native development. The roadblock is that there is too much configuration that you must get right, and it often is not obvious what path to start with. Aspire helps with this by being opinionated about what a component needs to provide, mandating that all components at a minimum provide resiliency defaults, health checks, and setup telemetry and integrate with DI.

9. What container technologies are supported by Aspire?

   Answer: Docker and Podman. Docker is the default.

10. What does the `AddProject` method do?

    Answer: The `AddProject` method adds a .NET project resource to the Aspire application model. This dynamically scans the referenced project for configuration in `launchSettings.json` like URLs. For example, you would call the `AddProject` method to add an ASP.NET Core Web API service project, as shown in the following code:

    ```
    var apiService = builder.AddProject
      <Projects.AspireStarter_ApiService>("apiservice");
    ```

Chapter 9 – Building Web Services Using ASP.NET Core Web API

Exercise 9.3 – Test your knowledge

Suggested answers to these questions:

1. Which class should you inherit from to create a controller class for an ASP.NET Core Web API service?

   Answer: To create a controller class for an ASP.NET Core Web API service, you should inherit from `ControllerBase`. Do not inherit from `Controller` as you would in MVC, because this class includes methods like `View` that use Razor files to render HTML that is not needed for a web service.

2. What must you do to specify which controller action method will be executed in response to an HTTP request?

   Answer: To specify which controller action method will be executed in response to a request, you must decorate the action method with an attribute. For example, to respond to an HTTP `POST` request, decorate the action method with `[HttpPost]`.

3. What must you do to specify what responses should be expected when calling an action method?

   Answer: To specify what responses should be expected when calling an action method, decorate the action method with the `[ProducesResponseType]` attribute, as shown in the following code:

   ```
   // GET: api/customers/[id]
   [HttpGet("{id}", Name = nameof(Get))] // named route
   [ProducesResponseType(200, Type = typeof(Customer))]
   [ProducesResponseType(404)]
   public IActionResult Get(string id)
   {
   ```

4. List three methods that can be called to return responses with different status codes.

   Answer: Three methods that can be called to return responses with different status codes include:

   - `Ok`: This returns the `200` status code and the object passed to this method in the body.
   - `CreatedAtRoute`: This returns the `201` status code and the object passed to this method in the body.
   - `NoContentResult`: This returns the `204` status code and an empty body.

   Other methods that you could have listed:

   - `BadRequest`: This returns the `400` status code and an optional error message.
   - `NotFound`: This returns the `404` status code and an optional error message.

5. List four ways that you can try out a web service.

   Answer: Four ways that you can test a web service include:

   - Using a browser to test simple HTTP `GET` requests
   - Installing the REST Client extension for VS Code or using the HTTP Editor in Visual Studio
   - Installing the Swagger NuGet package in your web service project, enabling Swagger, and then using the Swagger testing user interface
   - Installing the Postman tool from the following link: `https://www.postman.com`

I listed the four above because they are techniques that I covered or mentioned in the chapter. There are, of course, many other ways to test a web service, for example, `curl`. Give yourself an extra point for any valid additional methods beyond the ones I have listed.

Chapter 10 – Building Clients for Web Services

Exercise 10.3 – Test your knowledge

Suggested answers to these questions:

1. When configuring an HTTP client, how do you specify the format of data that you prefer in the response from the web service?

   Answer: When configuring an HTTP client, you specify the format of the response that you prefer by adding an `Accept` header to the HTTP request that specifies the document format you prefer, and if you specify multiple and want to give them different weights, then set quality values between `0.0` and `1.0`, as shown highlighted in the following code:

```
builder.Services.AddHttpClient(name: "Northwind.WebApi",
  configureClient: options =>
  {
    options.BaseAddress = new Uri("https://localhost:5002/");
    options.DefaultRequestHeaders.Accept.Add(
      new MediaTypeWithQualityHeaderValue(
      mediaType: "application/json", quality: 1.0));
  });
```

2. Why should you not wrap your use of `HttpClient` in a `using` statement to dispose of it when you are finished, even though it implements the `IDisposable` interface, and what should you use instead?

   Answer: `HttpClient` is shared, reentrant, and partially thread-safe, so it is tricky to use correctly in many scenarios. You should use `HttpClientFactory`, which was introduced in .NET Core 2.1.

3. What is CORS?

   Answer: CORS is an HTTP-header-based feature that asks the browser to disable its same-origin security policy in specific scenarios. The HTTP headers indicate which origins should be allowed in addition to the same origin. CORS is not about strengthening security; it is about weakening security to allow the sharing of resources across different origins!

4. What is a JWT?

   Answer: JWT is a standard that defines a compact and secure method to transmit information as a JSON object. The JSON object is digitally signed so it can be trusted. The most common scenario for using JWT is authorization.

5. What are the three parts of a JWT?

   Answer: JWTs consist of three parts separated by dots. These parts are the *header*, *payload*, and *signature*, as shown in the following format: `aaa.bbb.ccc`. The header and payload are Base64 encoded.

Chapter 11 – Testing and Debugging Web Services

Exercise 11.3 – Test your knowledge

Suggested answers to these questions:

1. What is a MUT?

   Answer: **Method under test** (**MUT**) is a method within a SUT being tested. **System under test** (**SUT**) is a type like a class being tested. You often create a test class with multiple test methods to group all the test methods for the SUT.

2. What is a test double?

   Answer: A test double is the umbrella term for any fake dependency in a test. They are used in tests in place of real dependencies that would be harder to set up consistently than a double. There are multiple types of double. The most common are mocks and stubs:

   - Mocks are doubles for outgoing interactions. For example, the test could call a mocked dependency that fakes sending an email during user registration. The state could be changed in the external system. Mocks are usually created using a mocking framework. When they are manually created, they are sometimes called spies.
   - Stubs are doubles for incoming interactions. For example, the test could call a stubbed dependency that retrieves product information from a database. No state is changed in the external system. When the dependency does not yet exist, for example, if using TDD, then a stub is known as a fake. When a stub is a simple value, and does not affect the outcome, then it is known as a dummy.

3. What is the most dangerous test outcome from the following: TN, TP, FN, and FP? Why?

   Answer: **False Negatives** (**FNs**) are much more dangerous than the others because they allow real defects to go unnoticed. These can cause functional problems, security vulnerabilities, and production failures, making them more costly and risk-laden in the long term. FPs only waste time and resources. TNs ("there are no bugs") and TPs ("there is a bug here") are both good results, as long as you then fix the bugs found, of course!

4. For integration testing, what types of external systems should you test, and which should you mock?

   Answer: External systems come in two types: ones under your control and ones outside your control. External systems under your control include data stores that only your project can access. No other system updates the data. External systems outside your control include email systems and public services like weather or government systems. You should directly use external systems under your control but mock external systems that you don't control.

5. What are the benefits of automating tests?

   Answer: Automated tests run consistently with the same inputs, reducing the likelihood of human error and ensuring repeatable results across multiple test cycles. Running automated functional tests is faster than manual testing, allowing for frequent and quick feedback during development. Test automation enables the execution of thousands of tests quickly, something that's impractical manually.

6. What should you consider when integration testing with data stores?

   Answer: The schema for your data stores should be treated like code and be tracked in a source control system like Git. This allows you to keep all the schema changes over time in sync with changes to code that works on data in that structure.

7. How are tests in xUnit configured?

   Answer: Tests in xUnit are configured using .NET attributes, which makes the test code easy to read and understand. It uses `[Fact]` for standard test cases and `[Theory]` with `[InlineData]`, `[ClassData]`, or `[MemberData]` for parameterized tests, enabling data-driven testing.

8. Why would you use the `WebApplicationFactory` class in an integration test?

   Answer: Integration tests for ASP.NET Core projects require a test web host for the SUT and use a test server client to handle requests and responses with the SUT. Use the `WebApplicationFactory` class to streamline bootstrapping the SUT with the `TestServer` class.

9. Using NSubstitute, how do you configure the return value of a faked method?

   Answer: Using NSubstitute, you configure the return value of a faked method by calling the `Returns` method on the object that will substitute, as shown in the following code:

   ```
   ICalculator calc = Substitute.For<ICalculator>();
   calc.Add(2, 3).Returns(5);
   ```

10. What are dev tunnels and how are they useful to .NET developers?

    Answer: Dev tunnels allow developers to expose their local development environment to the wider internet securely. This setup is especially useful when you need to share your work-in-progress with colleagues, clients, or third-party services without deploying it to a public staging environment.

Chapter 12 – Building Web Services Using ASP.NET Core OData

Exercise 12.3 – Test your knowledge

Suggested answers to these questions:

1. What transport protocol does an OData service use?

   Answer: OData uses the **Hypertext Transport Protocol** (**HTTP**).

2. Why is an OData service more flexible than a traditional ASP.NET Core Web API service?

   Answer: OData uses query strings for its queries that enable the client to control what is returned, minimizing round trips. A traditional web API defines all the methods and what gets returned.

3. What must you do to an action method in an OData controller to enable query strings to customize what it returns?

   Answer: You must decorate an action method in an OData controller with the `[EnableQuery]` attribute to enable query strings to customize what it returns.

4. What URL path would return customers in Germany who have made more than one order?

   Answer:

   ```
   /ordersystem/customers
     ?$select=CustomerId,CompanyName,City,Country
     &$filter=(Country eq 'Germany') and (Orders/$count gt 1)
   ```

   You can learn more about using `count` in filters at the following link: `https://devblogs.microsoft.com/odata/adding-support-for-count-segment-in-filter-collections-in-odata-webapi/`.

5. How do you get related entities?

   Answer: Use `$expand` to specify the name of the navigation property, as shown in the following example:

   ```
   /ordersystem/customers
     ?$select=CustomerId,CompanyName,Orders
     &$expand=Orders
   ```

Chapter 13 – Building Web Services Using FastEndpoints

Exercise 13.3 – Test your knowledge

Suggested answers to these questions:

1. Why is a web service built using FastEndpoints “fast”?

   Answer: FastEndpoints is “fast” due to several optimizations and design choices, including mapping requests directly to endpoints, which reduces the complexity and time involved in routing and handling requests, reducing the number of middleware components involved in the request processing pipeline, and minimizing the use of reflection.

2. How do you define an endpoint that has complex request and response models using FastEndpoints?

   Answer: Define a class that inherits from `Endpoint<TRequest, TResponse>` so that it defines an endpoint with both a request model and a response model. Optionally, use a class that inherits from `Endpoint<TRequest, TResponse, TMapper>` if it needs to handle complex mappings.

3. If you do not call the `Verbs` method in the overridden `Configure` method of a FastEndpoints endpoint class, which HTTP methods does the endpoint respond to?

   Answer: Without a call to the Verbs method in the overridden Configure method of a FastEndpoints endpoint class, an endpoint will default to handling all HTTP methods.

4. Why would you call the `Authorize` method rather than the `Roles` method to control access to a FastEndpoints endpoint?

   Answer: The `Roles` method restricts access to the endpoint based on membership of user roles. The `Authorize` method enforces more complex and flexible authorization policies on the endpoint. You can specify a policy that needs to be satisfied for access to be granted. It requires callers to pass specific authorization rules, allowing more fine-grained access control than just role-based checks.

5. Why might you call the `PreProcessors` and `PostProcessors` methods?

   Answer: These are methods that can be used to inject logic before or after the request is processed by the `HandleAsync` method. They allow you to perform common tasks such as validation, logging, and so on. For example:

   ```
   PreProcessors(new MyRequestValidator());
   PostProcessors(new MyResponseLogger());
   ```

Chapter 14 – Web Content Management Using Umbraco CMS

Exercise 14.3 – Test your knowledge

Suggested answers to these questions:

1. What are some basic features that every CMS would have?

   Answer: Any decent basic CMS will include the following core features:

   - A user interface that allows non-technical content owners to log in and manage their content
   - Media asset management of images, videos, documents, and other files
   - Sharing and reuse of pieces of content, often named blocks
   - Saved drafts of content that are hidden from website visitors until they are published
   - Search engine-optimized URLs, page titles and related metadata, sitemaps, and so on
   - Authentication and authorization including management of users, groups, and their access rights to content
   - A content delivery system that converts the content from simple data into one or more formats, like HTML and JSON

2. For how long are Umbraco LTS releases supported?

   Answer: Umbraco has a **Long-Term Support** (**LTS**) release every two years that aligns with .NET LTS releases. Like .NET, LTS releases are supported for three years. Umbraco CMS 17 (LTS) is aligned with .NET 10. Umbraco CMS 21 (LTS) will be aligned with .NET 12.

3. How do you make Umbraco CMS available as a .NET project template?

   Answer: To make Umbraco CMS available as a .NET project template you must install the templates, as shown in the following command:

   ```
   dotnet new install Umbraco.Templates
   ```

4. What does the `BootUmbracoAsync` method do?

   Answer: The `BootUmbracoAsync` method performs actions like initiating the database connection to verify connectivity and confirming that the database is ready for CRUD operations. It might also include running any outstanding database migrations to ensure schema compatibility with the code.

5. What is the relative path to the Umbraco back office?

   Answer: The Umbraco backoffice is accessed through the `/umbraco` relative path.

6. What sections of the Umbraco back office can a member of the editors user group access?

   Answer: Editors can only access the Content and Media sections.

7. In Umbraco, what is a document type, and what are its three main components or aspects?

   Answer: A document type is a blueprint for content on a site. It defines the structure and properties of a piece of content, such as a page or a component of a page. Document types allow you to manage and organize content consistently across a website by setting up rules and templates that content editors follow when creating new pages or elements.

8. In the settings for a content type, why might you select the **Allow vary by culture** checkbox?

   Answer: Selecting the **Allow vary by culture** check box enables CMS users to translate text on the website to supported human languages like English, French, and Spanish.

9. What syntax is used to define an Umbraco content template?

   Answer: Razor syntax is used to define an Umbraco content template so editors can mix HTML, CSS, and C# code to render their content.

10. What are two ways of organizing media?

    Answer: You can organize media using folders to define a hierarchy like a filesystem. You can also use tags. Administrators should establish a set of tags relevant to the content and then editors should use them consistently to improve the searchability of assets across the team.

Chapter 15 – Customizing and Extending Umbraco CMS

Exercise 15.3 – Test your knowledge

Suggested answers to these questions:

1. Why might you build a custom property editor for Umbraco CMS?

   Answer: If a built-in property editor is too basic then you might want to build a custom one. For example, the built-in media picker does not support features like advanced metadata tagging, file categorization, and asset previews right within the editor.

2. What file types are disallowed for uploads by default?

   Answer: Common dynamically executable file types like ASPX and CSHTML, as shown in the following configuration:

```
"DisallowedUploadFiles": ["ashx", "aspx", "ascx", "config", "cshtml",
"vbhtml", "asmx", "air", "axd", "xamlx"],
```

3. By default, does Umbraco CMS follow the modern security good practice of enforcing password minimum length without enforcing special characters?

   Answer: Yes, it forces a minimum number of characters of 10 and does not require special characters, as shown in the following configuration:

```
"UserPassword": {
  "RequiredLength": 10,
  "RequireNonLetterOrDigit": false,
    "RequireDigit": false,
    "RequireLowercase": false,
    "RequireUppercase": false,
    "HashAlgorithmType": "PBKDF2.ASPNETCORE.V3",
    "MaxFailedAccessAttemptsBeforeLockout": 5
  },
```

4. What is the difference between the `KeepAllVersionsNewerThanDays` and `KeepLatestVersionPerDayForDays` settings?

   Answer: The `KeepAllVersionsNewerThanDays` setting defaults to `7` (days) and it retains all versions created within the last seven days, preserving recent version history. The `KeepLatestVersionPerDayForDays` setting defaults to `90` (days) and so content versions older than 90 days are deleted.

5. What are the two most commonly rendered properties of a published content item?

   Answer: `Name` and `Url`.

6. What does the Models Builder feature allow you to do?

   Answer: The Models Builder feature allows you to work with strongly typed models in your views. For example, instead of `@Model.Value("bodyText")` you can use `@Model.BodyText`.

7. How can you programmatically load media?

   Answer: To load a media item, for example, an image or document by its ID, you can use the `TypedMedia` method, as shown in the following code:

```
@{
  IPublishedContent? media = _umbracoHelper.Media(5678); // 5678 is the
media ID.
}

@if (media is not null)
{
  <img src="@media.Url()" alt="@media.Name" />
}
else
{
  <p>Media not found.</p>
}
```

8. How can you programmatically render content using a template?

   Answer: You can use the `RenderTemplate` method to render a specific template for a given content item, as shown in the following code:

```
@{
  IPublishedContent? content = _umbracoHelper.Content(1234);
  IHtmlEncodedString renderedTemplate =
    await _umbracoHelper.RenderTemplateAsync(1234);
}

@if (!string.IsNullOrEmpty(renderedTemplate))
{
  @Html.Raw(renderedTemplate)
}
else
{
  <p>Template could not be rendered.</p>
}
```

9. What are dictionary values used for?

   Answer: For multilingual sites, Umbraco allows you to define dictionary items. You can retrieve their values using `GetDictionaryValue`, which will fetch the dictionary value for the key `WelcomeText` based on the current language, as shown in the following code:

```
@{
  string? welcomeText = _umbracoHelper.GetDictionaryValue("WelcomeText");
}

<p>@welcomeText</p>
```

10. In Umbraco CMS, what is a member?

    Answer: Members are registered visitors or users of the website or CMS.

This is the end of *Appendix A*.