What's new in ASP.NET Core 2.1

Article • 02/08/2023

This article highlights the most significant changes in ASP.NET Core 2.1, with links to relevant documentation.

SignalR

SignalR has been rewritten for ASP.NET Core 2.1.

ASP.NET Core SignalR includes a number of improvements:

- A simplified scale-out model.
- A new JavaScript client with no jQuery dependency.
- A new compact binary protocol based on MessagePack.
- Support for custom protocols.
- A new streaming response model.
- Support for clients based on bare WebSockets.

For more information, see ASP.NET Core SignalR.

Razor class libraries

ASP.NET Core 2.1 makes it easier to build and include Razor-based UI in a library and share it across multiple projects. The new Razor SDK enables building Razor files into a class library project that can be packaged into a NuGet package. Views and pages in libraries are automatically discovered and can be overridden by the app. By integrating Razor compilation into the build:

- The app startup time is significantly faster.
- Fast updates to Razor views and pages at runtime are still available as part of an iterative development workflow.

For more information, see Create reusable UI using the Razor Class Library project.

Identity UI library & scaffolding

ASP.NET Core 2.1 provides ASP.NET Core Identity as a Razor Class Library. Apps that include Identity can apply the new Identity scaffolder to selectively add the source code contained in the Identity Razor Class Library (RCL). You might want to generate source code so you can modify the code and change the behavior. For example, you could instruct the scaffolder to generate the code used in registration. Generated code takes precedence over the same code in the Identity RCL.

Apps that do **not** include authentication can apply the Identity scaffolder to add the RCL Identity package. You have the option of selecting Identity code to be generated.

For more information, see Scaffold Identity in ASP.NET Core projects.

HTTPS

With the increased focus on security and privacy, enabling HTTPS for web apps is important. HTTPS enforcement is becoming increasingly strict on the web. Sites that don't use HTTPS are considered insecure. Browsers (Chromium, Mozilla) are starting to enforce that web features must be used from a secure context. GDPR requires the use of HTTPS to protect user privacy. While using HTTPS in production is critical, using HTTPS in development can help prevent issues in deployment (for example, insecure links). ASP.NET Core 2.1 includes a number of improvements that make it easier to use HTTPS in development and to configure HTTPS in production. For more information, see Enforce HTTPS.

On by default

To facilitate secure website development, HTTPS is now enabled by default. Starting in 2.1, Kestrel listens on `https://localhost:5001` when a local development certificate is present. A development certificate is created:

- As part of the .NET Core SDK first-run experience, when you use the SDK for the first time.
- Manually using the new `dev-certs` tool.

Run `dotnet dev-certs https --trust` to trust the certificate.

HTTPS redirection and enforcement

Web apps typically need to listen on both HTTP and HTTPS, but then redirect all HTTP traffic to HTTPS. In 2.1, specialized HTTPS redirection middleware that intelligently redirects based on the presence of configuration or bound server ports has been introduced.

Use of HTTPS can be further enforced using HTTP Strict Transport Security Protocol (HSTS). HSTS instructs browsers to always access the site via HTTPS. ASP.NET Core 2.1

adds HSTS middleware that supports options for max age, subdomains, and the HSTS preload list.

Configuration for production

In production, HTTPS must be explicitly configured. In 2.1, default configuration schema for configuring HTTPS for Kestrel has been added. Apps can be configured to use:

- Multiple endpoints including the URLs. For more information, see Kestrel web server implementation: Endpoint configuration.
- The certificate to use for HTTPS either from a file on disk or from a certificate store.

GDPR

ASP.NET Core provides APIs and templates to help meet some of the EU General Data Protection Regulation (GDPR) requirements. For more information, see GDPR support in ASP.NET Core. A sample app shows how to use and lets you test most of the GDPR extension points and APIs added to the ASP.NET Core 2.1 templates.

Integration tests

A new package is introduced that streamlines test creation and execution. The Microsoft.AspNetCore.Mvc.Testing package handles the following tasks:

- Copies the dependency file (**.deps*) from the tested app into the test project's *bin* folder.
- Sets the content root to the tested app's project root so that static files and pages/views are found when the tests are executed.
- Provides the WebApplicationFactory<TEntryPoint> class to streamline bootstrapping the tested app with TestServer.

The following test uses xUnit to check that the Index page loads with a success status code and with the correct Content-Type header:

C#

```
public class BasicTests
    : IClassFixture<WebApplicationFactory<RazorPagesProject.Startup>>
{
    private readonly HttpClient _client;

    public
BasicTests(WebApplicationFactory<RazorPagesProject.Startup> factory)
```

```
    {
        _client = factory.CreateClient();
    }

    [Fact]
    public async Task GetHomePage()
    {
        // Act
        var response = await _client.GetAsync("/");

        // Assert
        response.EnsureSuccessStatusCode(); // Status Code 200-299
        Assert.Equal("text/html; charset=utf-8",
            response.Content.Headers.ContentType.ToString());
    }
}
```

For more information, see the Integration tests topic.

[ApiController], ActionResult<T>

ASP.NET Core 2.1 adds new programming conventions that make it easier to build clean and descriptive web APIs. `ActionResult<T>` is a new type added to allow an app to return either a response type or any other action result (similar to IActionResult), while still indicating the response type. The `[ApiController]` attribute has also been added as the way to opt in to Web API-specific conventions and behaviors.

For more information, see Build Web APIs with ASP.NET Core.

IHttpClientFactory

ASP.NET Core 2.1 includes a new `IHttpClientFactory` service that makes it easier to configure and consume instances of `HttpClient` in apps. `HttpClient` already has the concept of delegating handlers that could be linked together for outgoing HTTP requests. The factory:

- Makes registering of instances of `HttpClient` per named client more intuitive.
- Implements a Polly handler that allows Polly policies to be used for Retry, CircuitBreakers, etc.

For more information, see Initiate HTTP Requests.

Kestrel libuv transport configuration

With the release of ASP.NET Core 2.1, Kestrel's default transport is no longer based on Libuv but instead based on managed sockets. For more information, see Kestrel web server implementation: Libuv transport configuration.

Generic host builder

The Generic Host Builder (`HostBuilder`) has been introduced. This builder can be used for apps that don't process HTTP requests (Messaging, background tasks, etc.).

For more information, see .NET Generic Host.

Updated SPA templates

The Single Page Application templates for Angular and React are updated to use the standard project structures and build systems for each framework.

The Angular template is based on the Angular CLI, and the React template is based on create-react-app.

For more information, see:

- Use Angular with ASP.NET Core
- Use React with ASP.NET Core

Razor Pages search for Razor assets

In 2.1, Razor Pages search for Razor assets (such as layouts and partials) in the following directories in the listed order:

1. Current Pages folder.
2. */Pages/Shared/*
3. */Views/Shared/*

Razor Pages in an area

Razor Pages now support areas. To see an example of areas, create a new Razor Pages web app with individual user accounts. A Razor Pages web app with individual user accounts includes */Areas/Identity/Pages*.

MVC compatibility version

The SetCompatibilityVersion method allows an app to opt-in or opt-out of potentially breaking behavior changes introduced in ASP.NET Core MVC 2.1 or later.

For more information, see Compatibility version for ASP.NET Core MVC.

Migrate from 2.0 to 2.1

See Migrate from ASP.NET Core 2.0 to 2.1.

Additional information

For the complete list of changes, see the ASP.NET Core 2.1 Release Notes.