What's new in ASP.NET Core 2.0

Article • 06/04/2022

This article highlights the most significant changes in ASP.NET Core 2.0, with links to relevant documentation.

Razor Pages

Razor Pages is a new feature of ASP.NET Core MVC that makes coding page-focused scenarios easier and more productive.

For more information, see the introduction and tutorial:

- Introduction to Razor Pages
- Get started with Razor Pages

ASP.NET Core metapackage

A new ASP.NET Core metapackage includes all of the packages made and supported by the ASP.NET Core and Entity Framework Core teams, along with their internal and 3rd-party dependencies. You no longer need to choose individual ASP.NET Core features by package. All features are included in the Microsoft.AspNetCore.All package. The default templates use this package.

For more information, see Microsoft.AspNetCore.All metapackage for ASP.NET Core 2.0.

Runtime Store

Applications that use the `Microsoft.AspNetCore.All` metapackage automatically take advantage of the new .NET Core Runtime Store. The Store contains all the runtime assets needed to run ASP.NET Core 2.0 applications. When you use the `Microsoft.AspNetCore.All` metapackage, no assets from the referenced ASP.NET Core NuGet packages are deployed with the application because they already reside on the target system. The assets in the Runtime Store are also precompiled to improve application startup time.

For more information, see Runtime store

.NET Standard 2.0

The ASP.NET Core 2.0 packages target .NET Standard 2.0. The packages can be referenced by other .NET Standard 2.0 libraries, and they can run on .NET Standard 2.0-compliant implementations of .NET, including .NET Core 2.0 and .NET Framework 4.6.1.

The `Microsoft.AspNetCore.All` metapackage targets .NET Core 2.0 only, because it's intended to be used with the .NET Core 2.0 Runtime Store.

Configuration update

An `IConfiguration` instance is added to the services container by default in ASP.NET Core 2.0. `IConfiguration` in the services container makes it easier for applications to retrieve configuration values from the container.

For information about the status of planned documentation, see the GitHub issue.

Logging update

In ASP.NET Core 2.0, logging is incorporated into the dependency injection (DI) system by default. You add providers and configure filtering in the `Program.cs` file instead of in the `Startup.cs` file. And the default `ILoggerFactory` supports filtering in a way that lets you use one flexible approach for both cross-provider filtering and specific-provider filtering.

For more information, see Introduction to Logging.

Authentication update

A new authentication model makes it easier to configure authentication for an application using DI.

New templates are available for configuring authentication for web apps and web APIs using Azure AD B2C.

For information about the status of planned documentation, see the GitHub issue.

Identity update

We've made it easier to build secure web APIs using Identity in ASP.NET Core 2.0. You can acquire access tokens for accessing your web APIs using the Microsoft Authentication Library (MSAL).

For more information on authentication changes in 2.0, see the following resources:

- Account confirmation and password recovery in ASP.NET Core
- Enable QR Code generation for authenticator apps in ASP.NET Core
- Migrate Authentication and Identity to ASP.NET Core 2.0

SPA templates

Single Page Application (SPA) project templates for Angular, Aurelia, Knockout.js, React.js, and React.js with Redux are available. The Angular template has been updated to Angular 4. The Angular and React templates are available by default; for information about how to get the other templates, see Create a new SPA project. For information about how to build a SPA in ASP.NET Core, see The features described in this article are obsolete as of ASP.NET Core 3.0.

Kestrel improvements

The Kestrel web server has new features that make it more suitable as an Internet-facing server. A number of server constraint configuration options are added in the `KestrelServerOptions` class's new `Limits` property. Add limits for the following:

- Maximum client connections
- Maximum request body size
- Minimum request body data rate

For more information, see Kestrel web server implementation in ASP.NET Core.

WebListener renamed to HTTP.sys

The packages `Microsoft.AspNetCore.Server.WebListener` and `Microsoft.Net.Http.Server` have been merged into a new package `Microsoft.AspNetCore.Server.HttpSys`. The namespaces have been updated to match.

For more information, see HTTP.sys web server implementation in ASP.NET Core.

Enhanced HTTP header support

When using MVC to transmit a `FileStreamResult` or a `FileContentResult`, you now have the option to set an `ETag` or a `LastModified` date on the content you transmit.

You can set these values on the returned content with code similar to the following:

```
C#
```

```
var data = Encoding.UTF8.GetBytes("This is a sample text from a bina-
ry array");
var entityTag = new
EntityTagHeaderValue("\"MyCalculatedEtagValue\"");
return File(data, "text/plain", "downloadName.txt", lastModified:
DateTime.UtcNow.AddSeconds(-5), entityTag: entityTag);
```

The file returned to your visitors has the appropriate HTTP headers for the `ETag` and `LastModified` values.

If an application visitor requests content with a Range Request header, ASP.NET Core recognizes the request and handles the header. If the requested content can be partially delivered, ASP.NET Core appropriately skips and returns just the requested set of bytes. You don't need to write any special handlers into your methods to adapt or handle this feature; it's automatically handled for you.

Hosting startup and Application Insights

Hosting environments can now inject extra package dependencies and execute code during application startup, without the application needing to explicitly take a dependency or call any methods. This feature can be used to enable certain environments to "light-up" features unique to that environment without the application needing to know ahead of time.

In ASP.NET Core 2.0, this feature is used to automatically enable Application Insights diagnostics when debugging in Visual Studio and (after opting in) when running in Azure App Services. As a result, the project templates no longer add Application Insights packages and code by default.

For information about the status of planned documentation, see the GitHub issue.

Automatic use of anti-forgery tokens

ASP.NET Core has always helped HTML-encode content by default, but with the new version an extra step is taken to help prevent cross-site request forgery (XSRF) attacks. ASP.NET Core will now emit anti-forgery tokens by default and validate them on form POST actions and pages without extra configuration.

For more information, see Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core.

Automatic precompilation

Razor view pre-compilation is enabled during publish by default, reducing the publish output size and application startup time.

For more information, see Razor view compilation and precompilation in ASP.NET Core.

Razor support for C# 7.1

The Razor view engine has been updated to work with the new Roslyn compiler. That includes support for C# 7.1 features like Default Expressions, Inferred Tuple Names, and Pattern-Matching with Generics. To use C# 7.1 in your project, add the following property in your project file and then reload the solution:

XML

```
<LangVersion>latest</LangVersion>
```

For information about the status of C# 7.1 features, see the Roslyn GitHub repository.

Other documentation updates for 2.0

- Visual Studio publish profiles for ASP.NET Core app deployment
- Key Management
- Configure Facebook authentication
- Configure Twitter authentication
- Configure Google authentication
- Configure Microsoft Account authentication

Migration guidance

For guidance on how to migrate ASP.NET Core 1.x applications to ASP.NET Core 2.0, see the following resources:

- Migrate from ASP.NET Core 1.x to ASP.NET Core 2.0
- Migrate Authentication and Identity to ASP.NET Core 2.0

Additional Information

For the complete list of changes, see the ASP.NET Core 2.0 Release Notes.

To connect with the ASP.NET Core development team's progress and plans, tune in to the ASP.NET Community Standup.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

ASP.NET Core feedback

ASP.NET Core is an open source project. Select a link to provide feedback:

- Open a documentation issue
- Provide product feedback