What's new in ASP.NET Core 3.1

Article • 09/26/2023

This article highlights the most significant changes in ASP.NET Core 3.1 with links to relevant documentation.

Partial class support for Razor components

Razor components are now generated as partial classes. Code for a Razor component can be written using a code-behind file defined as a partial class rather than defining all the code for the component in a single file. For more information, see Partial class support.

Component Tag Helper and pass parameters to top-level components

In Blazor with ASP.NET Core 3.0, components were rendered into pages and views using an HTML Helper (`Html.RenderComponentAsync`). In ASP.NET Core 3.1, render a component from a page or view with the new Component Tag Helper:

CSHTML

```
<component type="typeof(Counter)" render-mode="ServerPrerendered" />
```

The HTML Helper remains supported in ASP.NET Core 3.1, but the Component Tag Helper is recommended.

Blazor Server apps can now pass parameters to top-level components during the initial render. Previously you could only pass parameters to a top-level component with RenderMode.Static. With this release, both RenderMode.Server and RenderMode.ServerPrerendered are supported. Any specified parameter values are serialized as JSON and included in the initial response.

For example, prerender a `Counter` component with an increment amount (`IncrementAmount`):

CSHTML

```
<component type="typeof(Counter)" render-mode="ServerPrerendered"
    param-IncrementAmount="10" />
```

For more information, see Integrate components into Razor Pages and MVC apps.

Support for shared queues in HTTP.sys

HTTP.sys supports creating anonymous request queues. In ASP.NET Core 3.1, we've added the ability to create or attach to an existing named HTTP.sys request queue. Creating or attaching to an existing named HTTP.sys request queue enables scenarios where the HTTP.sys controller process that owns the queue is independent of the listener process. This independence makes it possible to preserve existing connections and enqueued requests between listener process restarts:

C#

```
public static IHostBuilder CreateHostBuilder(string[] args) =>
     Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder =>
        {
            // ...
            webBuilder.UseHttpSys(options =>
            {
                options.RequestQueueName = "MyExistingQueue";
                options.RequestQueueMode =
RequestQueueMode.CreateOrAttach;
            });
        });
```

Breaking changes for SameSite cookies

The behavior of SameSite cookies has changed to reflect upcoming browser changes. This may affect authentication scenarios like AzureAd, OpenIdConnect, or WsFederation. For more information, see Work with SameSite cookies in ASP.NET Core.

Prevent default actions for events in Blazor apps

Use the `@on{EVENT}:preventDefault` directive attribute to prevent the default action for an event. In the following example, the default action of displaying the key's character in the text box is prevented:

razor

```
<input value="@_count" @onkeypress="KeyHandler" @onkeypress:prevent-
```

```
Default />
```

For more information, see Prevent default actions.

Stop event propagation in Blazor apps

Use the `@on{EVENT}:stopPropagation` directive attribute to stop event propagation. In the following example, selecting the checkbox prevents click events from the child `<div>` from propagating to the parent `<div>`:

razor

```
<input @bind="_stopPropagation" type="checkbox" />

<div @onclick="OnSelectParentDiv">
    <div @onclick="OnSelectChildDiv" @onclick:stopPropagation="_stop-
Propagation">
        ...
    </div>
</div>

@code {
    private bool _stopPropagation = false;
}
```

For more information, see Stop event propagation.

Detailed errors during Blazor app development

When a Blazor app isn't functioning properly during development, receiving detailed error information from the app assists in troubleshooting and fixing the issue. When an error occurs, Blazor apps display a gold bar at the bottom of the screen:

- During development, the gold bar directs you to the browser console, where you can see the exception.
- In production, the gold bar notifies the user that an error has occurred and recommends refreshing the browser.

For more information, see Handle errors in ASP.NET Core Blazor apps.

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

Open a documentation issue

Provide product feedback