MICROSOFT ARCHITECT
By Joydip Kanjilal, Columnist, InfoWorld
JUL 13, 2020 3:00 AM PDT

How to use route data tokens in ASP.NET Core

Take advantage of data tokens in ASP.NET Core to attach additional information to a route and retrieve it programmatically when needed

ASP.NET Core is an open source, cross-platform, lean, and modular framework for building high-performance web applications. Routing in ASP.NET Core is managed by built-in routing middleware, which maps the incoming requests to the respective route handlers.

The routing infrastructure in ASP.NET Core provides support for data tokens, a useful feature that allows you to pass arbitrary data with routes and use them programmatically in the route handling pipeline. This article talks about how we can work with data tokens in ASP.NET Core.

[Also on InfoWorld: What is Jamstack? The static website revolution upending web development]

To work with the code examples provided in this article, you should have Visual Studio 2019 installed in your system. If you don't already have a copy, you can download Visual Studio 2019 here.

Create an ASP.NET Core MVC project

First off, let's create an ASP.NET Core project in Visual Studio. Assuming Visual Studio 2019 is installed in your system, follow the steps outlined below to create a new ASP.NET Core project in Visual Studio.

1. Launch the Visual Studio IDE.
2. Click on “Create new project.”
3. In the “Create new project” window, select “ASP.NET Core Web Application” from the list of templates displayed.
4. Click Next.
5. In the “Configure your new project” window shown next, specify the name and location for the new project.
6. Click Create.
7. In the “Create a New ASP.NET Core Web Application” window, select .NET Core as the runtime and ASP.NET Core 3.1 (or later) from the drop-down list at the top.
8. Select “Web Application (Model-View-Controller)” as the project template to create a new ASP.NET Core MVC application.
9. Ensure that the check boxes “Enable Docker Support” and “Configure for HTTPS” are unchecked as we won’t be using those features here.
10. Ensure that Authentication is set to “No Authentication” as we won’t be using authentication either.
11. Click Create.

Following these steps will create a new ASP.NET Core MVC project in Visual Studio 2019. We’ll use this project in the subsequent sections of this article.

Why use route data tokens?

In ASP.NET Core MVC, not only can you use URL segments to determine whether a route matches a request, but you can also associate additional information with a route. You can take advantage of data tokens in ASP.NET Core MVC to attach this additional information to a route and retrieve it programmatically when needed. Note that a data token is just another key-value pair and that a route can have any number of these key-value pairs, i.e., data tokens.

Data tokens are a useful feature of the ASP.NET Core MVC routing system, as they allow you to map routes to arbitrary values. Let's understand this with an example. Consider the following code snippet that takes advantage of the UseEndpoints extension method to define a route and associate it with a data token.


```
app.UseEndpoints(routes =>
{
   routes.MapControllerRoute(name: "All",
   pattern: "{*url}",
   defaults: new { controller = "Home", action = "Index" },
   constraints: new { },
   dataTokens: new { routeName = "All" });
});
```

Note how the "All" route has been mapped to a controller/action pair. You can take advantage of this route to show the home page if a more specific URL is not available.

Add data tokens to a route in ASP.NET Core

You can specify data tokens using the UseEndpoints or UseMvc extension methods. The following code snippet illustrates how you can add data tokens to a route in an ASP.NET Core MVC application.

```
app.UseEndpoints(endpoints =>
{
   endpoints.MapControllerRoute(
   name: "default",
   dataTokens: new { Name = "This is a sample data." },
   pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

Note the usage of the UseEndpoints extension method. Endpoint routing is a feature introduced in ASP.NET Core 3.0 MVC that enables you to decouple the route matching logic from the MVC middleware.

Before the introduction of endpoint routing, routing resolution in ASP.NET Core MVC was performed at the end of the request processing pipeline. As a result, route information (such as which action method to execute) was unknown to any middleware processing a request before the MVC middleware in the request processing pipeline. You can learn more about endpoint routing in ASP.NET Core 3.0 MVC from my article here.

If you are using a previous version of ASP.NET Core such as ASP.NET Core 2.0 or 2.1, you will need to use the UseMvc extension method instead as shown in the code snippet given below.

```
app.UseMvc(routes =>
{
  routes.MapRoute(
  name: "default",
  template: "{controller=Home}/{action=Index}/{id?}",
  defaults: null,
  constraints: null,
  dataTokens: new { Name = "This is a sample data."});
});
```

Access data tokens in action methods in ASP.NET Core

The data tokens that are associated with a route are initialized when a route maps an incoming URL to an action method. Once these data tokens have been set, you can access them from your action method using the RouteData.Tokens property. This property exposes the data tokens as a collection of key-value pairs. The following code snippet illustrates how you can access data tokens in an action method.

```
public IActionResult Index()
{
    var data = (string)RouteData.DataTokens["Name"];
    //Write your code here
    return View();
}
```

Data tokens are additional or arbitrary data that can be passed with routes and that are available for use in the route handling pipeline. You can store any type of data in data tokens (not just strings) and you can access this data from any upstream middleware or from the action methods of your controller.

How to do more in ASP.NET Core:

- How to use API versioning in ASP.NET Core
- How to use Data Transfer Objects in ASP.NET Core 3.1
- How to handle 404 errors in ASP.NET Core MVC
- How to use dependency injection in action filters in ASP.NET Core 3.1
- How to use the options pattern in ASP.NET Core
- How to use endpoint routing in ASP.NET Core 3.0 MVC
- How to export data to Excel in ASP.NET Core 3.0
- How to use LoggerMessage in ASP.NET Core 3.0
- How to send emails in ASP.NET Core
- How to log data to SQL Server in ASP.NET Core
- How to schedule jobs using Quartz.NET in ASP.NET Core

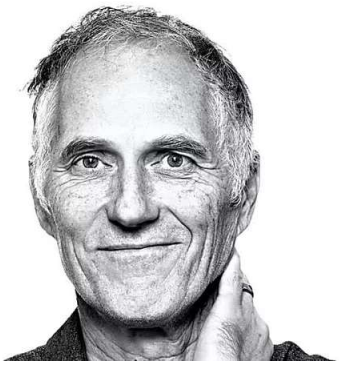
- How to return data from ASP.NET Core Web API

- How to format response data in ASP.NET Core
- How to consume an ASP.NET Core Web API using RestSharp
- How to perform async operations using Dapper
- How to use feature flags in ASP.NET Core
- How to use the FromServices attribute in ASP.NET Core
- How to work with cookies in ASP.NET Core
- How to work with static files in ASP.NET Core
- How to use URL Rewriting Middleware in ASP.NET Core
- How to implement rate limiting in ASP.NET Core
- How to use Azure Application Insights in ASP.NET Core
- Using advanced NLog features in ASP.NET Core
- How to handle errors in ASP.NET Web API
- How to implement global exception handling in ASP.NET Core MVC
- How to handle null values in ASP.NET Core MVC
- Advanced versioning in ASP.NET Core Web API
- How to work with worker services in ASP.NET Core
- How to use the Data Protection API in ASP.NET Core
- How to use conditional middleware in ASP.NET Core
- How to work with session state in ASP.NET Core
- How to write efficient controllers in ASP.NET Core

Joydip Kanjilal is a Microsoft MVP in ASP.Net, as well as a speaker and author of several books and articles. He has more than 20 years of experience in IT including more than 16 years in Microsoft .Net and related technologies.

Follow

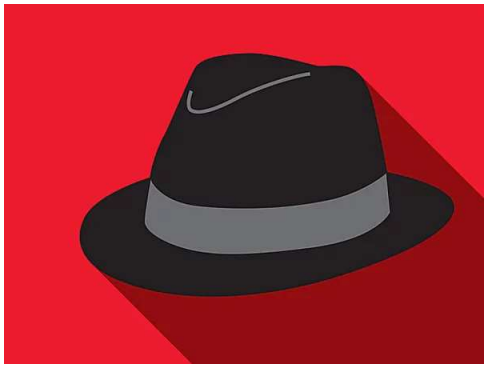
YOU MAY ALSO LIKE



Tim O'Reilly: the golden age of the programmer is over

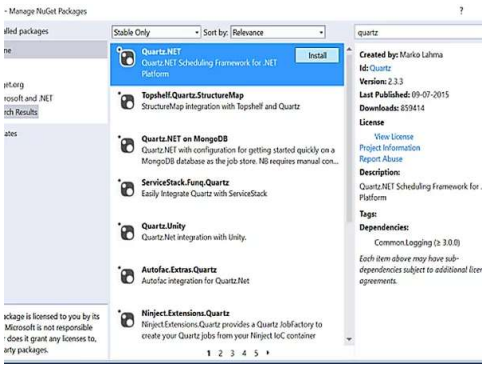
2 common cloud computing predictions for 2021 are wrong

Deno Company forms to back Node.js rival



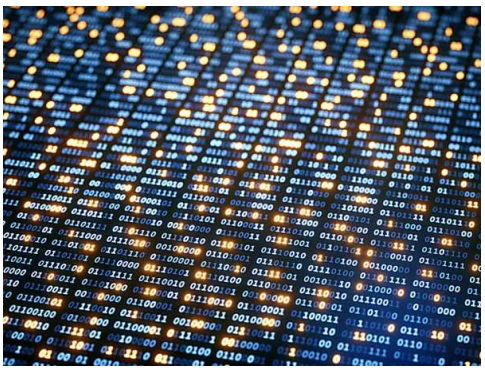
Red Hat OpenShift ramps up security and manageability with Platform Plus

What's new in Kubernetes 1.20



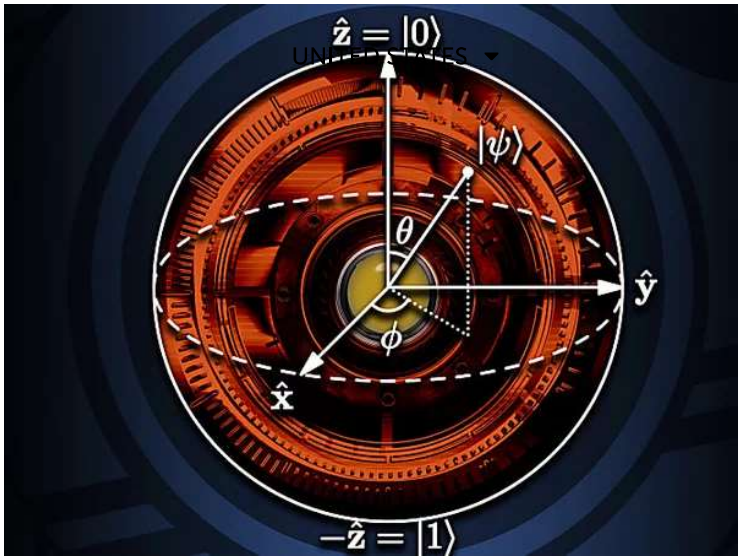
How to work with Quartz.Net in C#

Pulling devops and multicloud together



What is a computational storage drive? Much-

Red Hat Enterprise Linux takes aim at edge



Amazon Braket: Get started with quantum computing

10 top-notch libraries for C++ programming