How to use security headers in ASP.NET Core MVC 5

Take advantage of security headers in ASP.NET Core MVC 5 to protect your website against cross-site scripting, code injection, clickjacking, and other attacks.

By Joydip Kanjilal
Columnist, InfoWorld
JAN 11, 2021 3:00 AM PST

ASP.NET Core MVC 5 is a lightweight, open source, highly testable framework built on top of the ASP.NET Core 5 runtime and based on the model-view-controller (MVC) architecture. Part of the new .NET 5, the ASP.NET Core MVC 5 framework combines the capabilities of .NET Core, MVC, and Web API.

Security headers are a technique that can be used to improve the security of a web application. There are several ways in which you can specify security headers in your ASP.NET Core MVC application. This article talks about these ways with code examples wherever appropriate.

[Also on InfoWorld: Understanding Microsoft .NET 5]

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 5 project in Visual Studio 2019

First off, let's create an ASP.NET Core project in Visual Studio 2019. Following these steps should create a new ASP.NET Core 5 project in Visual Studio 2019.

1. Launch the Visual Studio IDE.
2. Click on “Create new project.”
3. In the “Create new project” window, select “ASP.NET Core Web App (Model-View-Controller)” from the list of templates displayed.
4. Click Next.
5. In the “Configure your new project” window, specify the name and location for the new project.
6. Optionally check the “Place solution and project in the same directory” check box, depending on your preferences.
7. Click Next.
8. In the “Additional Information” window shown next, select .NET 5.0 as the target framework from the drop-down list at the top. Leave the “Authentication Type” as None (default).
9. Ensure that the check boxes “Enable Docker,” “Configure for HTTPS,” and “Enable Razor runtime compilation” are unchecked as we won’t be using any of those features here.
10. Click Create.

A new ASP.NET Core MVC 5 project will be created. We’ll use this project in the subsequent sections in this article.

Specify headers in middleware in ASP.NET Core 5

Middleware components are used to inspect, route, or modify the request and response messages that flow through the pipeline. To specify headers in the middleware you can either create a new middleware class or take advantage of the Configure method pertaining to the Startup class as shown in the code snippet given below.

```
app.Use(async (context, next) =>
{
    context.Response.Headers.Add("Header-Key", "Header-Value");
    await next();
});
```

When you run the application, a new header with the name specified will be added to all responses.

Specify headers in web.config in ASP.NET Core 5

When working with ASP.NET Core or ASP.NET Core MVC 5 you no longer need a web.config file. However, using a web.config file is perfectly valid if you're hosting your application in IIS. The following code snippet shows how you can add custom headers in the web.config file.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.webServer>
    <httpProtocol>
      <customHeaders>
        <add name="Header-Key" value="Header-Value" />
      </customHeaders>
    </httpProtocol>
  </system.webServer>
</configuration>
```

When you run the above application and browse the GET endpoint using Postman, you should see the new header listed as shown in the screen image (Figure 1) below.

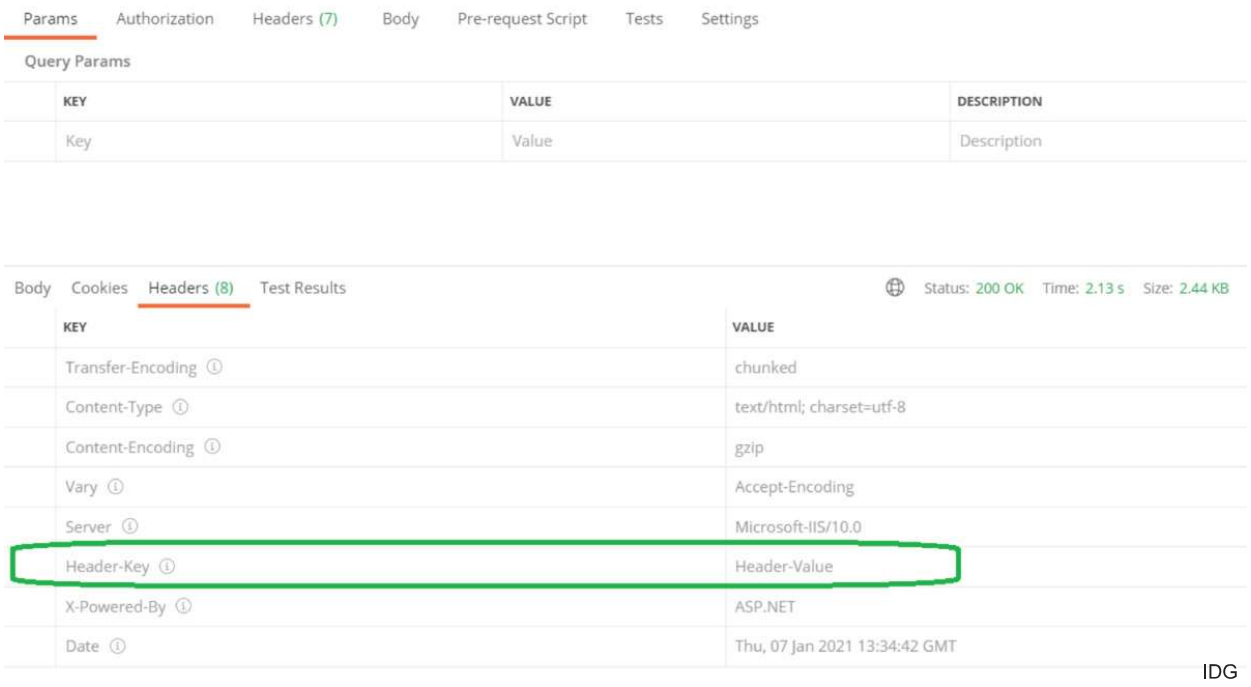


Figure 1.

Security headers in ASP.NET Core MVC 5

You can set certain HTTP header values to improve the security of web applications developed in ASP.NET Core MVC 5. These security headers when used properly can help protect an application.

The following is a list of some of the most widely used headers.

HTTP Strict-Transport-Security (HSTS)

You should take advantage of the HTTP Strict-Transport-Security header to prevent web pages from being served over plain HTTP — i.e., you can ensure that web pages will be transmitted only over HTTPS. It should be noted that ASP.NET Core MVC 5 framework contains a built-in middleware named HSTS. The following code snippet illustrates how we can take advantage of this middleware to impose this security restriction.

```
services.AddHsts(options =>
    {
        options.IncludeSubDomains = true;
        options.MaxAge = TimeSpan.FromDays(365);
    });
```

X-Frame-Options

The X-Frame-Options header prevents framing — i.e., it prevents browsers from rendering your web page within another web page, and thus prevents other websites from using your content. X-Frame-Options can be added using the following piece of code.

```
context.Response.Headers.Add("X-Frame-Options", "DENY");
```

X-Xss-Protection

The X-Xss-Protection header will cause modern-day browsers to stop loading the web page when they detect a cross-site scripting attack. The following code snippet shows how this header can be added.

```
context.Response.Headers.Add("X-Xss-Protection", "1; mode=block");
```

In the preceding code snippet, the value “1” implies enabled and the mode of “block” implies the web browser.

X-Content-Type-Options

The X-Content-Type-Options header is used to indicate that the MIME types specified in the Content-Type headers are deliberately configured and should not be changed by the browser. This header prevents MIME sniffing, which can be used by attackers to turn non-executable MIME types into executable ones.

```
app.UseXContentTypeOptions();
```

Referrer-Policy

When you click on a link in the website you're currently browsing, the control is transferred to the linked site. In addition, referrer data such as the URL could also be passed. If this URL includes the path and query string, then user privacy or security could be compromised. You can disable this behavior using the Referrer-Policy header as shown in the code snippet given below.

```
context.Response.Headers.Add("Referrer-Policy", "no-referrer");
```

X-Permitted-Cross-Domain-Policies

This header can be used to indicate if Adobe products are allowed to render the web page from a different domain than yours. In other words, like X-Frame-Options above, this header protects you against website spoofing or unauthorized use of your content. As an example, if you're using Flash in your application, you can prevent clients from making cross-site requests using the X-Permitted-Cross-Domain-Policies header using the following code snippet.

```
context.Response.Headers.Add("X-Permitted-Cross-Domain-Policies", "none");
```

X-Powered-By

The X-Powered-By header is added to the web.config file to identify the server technology (e.g., IIS) being used. You can remove this header if you've used a web.config file that has the X-Powered-By specified.

Feature-Policy

UNITED STATES

The Feature-Policy header is used to specify all of the features your application needs.

```
context.Response.Headers.Add("Feature-Policy", camera 'none'; geolocation 'none'; micro
```

Content-Security-Policy

Content Security Policy is a security policy that is used to control the resources that a web page is allowed to load. It represents an extra layer of security that is implemented via a Content-Security-Policy header in an HTTP response. Content-Security-Policy is used to detect and mitigate certain types of attacks such as cross-site scripting attacks and data injection attacks.

The following code snippet illustrates how this header can be used.

```
app.Use(async (ctx, next) =>
{
    ctx.Response.Headers.Add("Content-Security-Policy",
    "default-src 'self'; report-uri /idgreport");
    await next();
});
```

Security headers are fundamental to the security of a website. They can be used to help protect a website against the types of attacks your website will likely encounter such as cross-site scripting, code injection, and clickjacking. You can validate if you've set the security headers for your website properly at this link.

How to do more in ASP.NET Core:

- How to handle unknown actions in ASP.NET Core 5 MVC
- How to overload action methods in ASP.NET Core 5 MVC
- How to use multiple implementations of an interface in ASP.NET Core
- How to use IHttpClientFactory in ASP.NET Core
- How to use the ProblemDetails middleware in ASP.NET Core

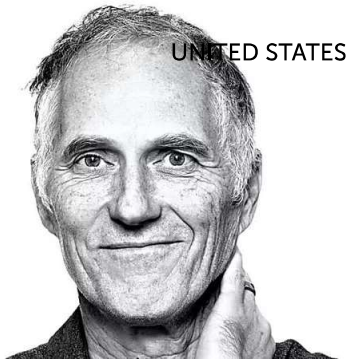
- How to create route constraints in ASP.NET Core
- How to manage user secrets in ASP.NET Core
- How to build gRPC applications in ASP.NET Core
- How to redirect a request in ASP.NET Core
- How to use attribute routing in ASP.NET Core
- How to pass parameters to action methods in ASP.NET Core MVC
- How to use API Analyzers in ASP.NET Core
- How to use route data tokens 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.

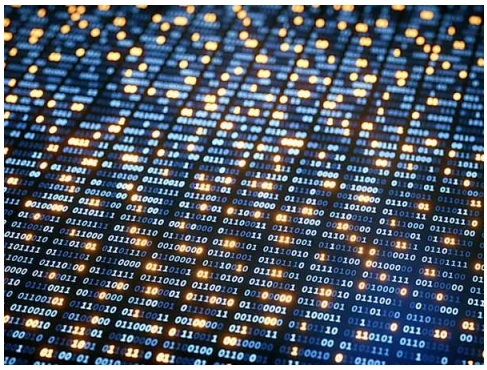
Follow



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

2 common cloud computing predictions for 2021 are wrong

8 databases supporting in-database machine learning



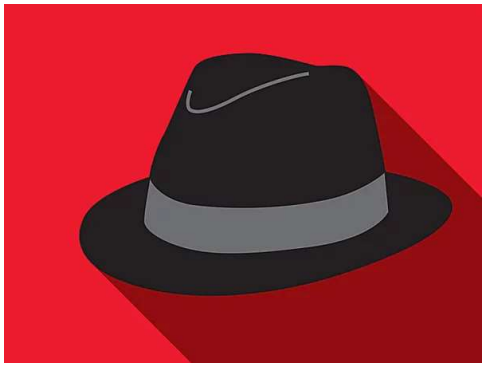
What is a computational storage drive? Much-needed help for CPUs

The decline of Heroku

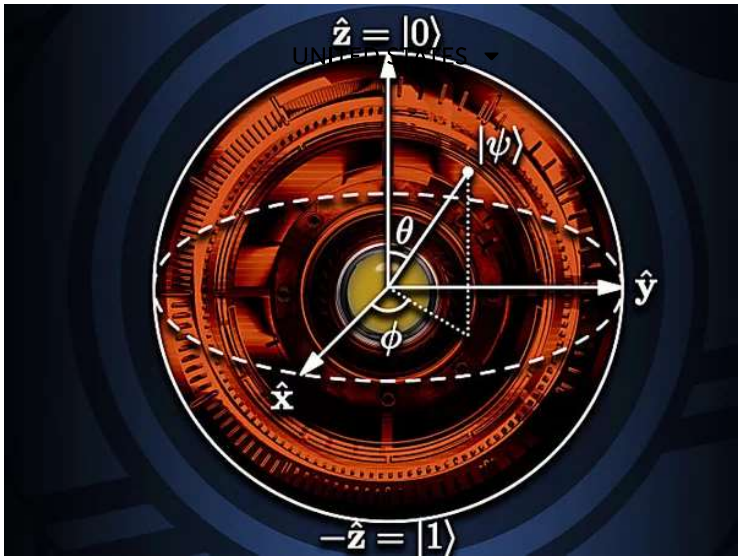
The shifting market for PostgreSQL

Edge computing can be a data cache for public

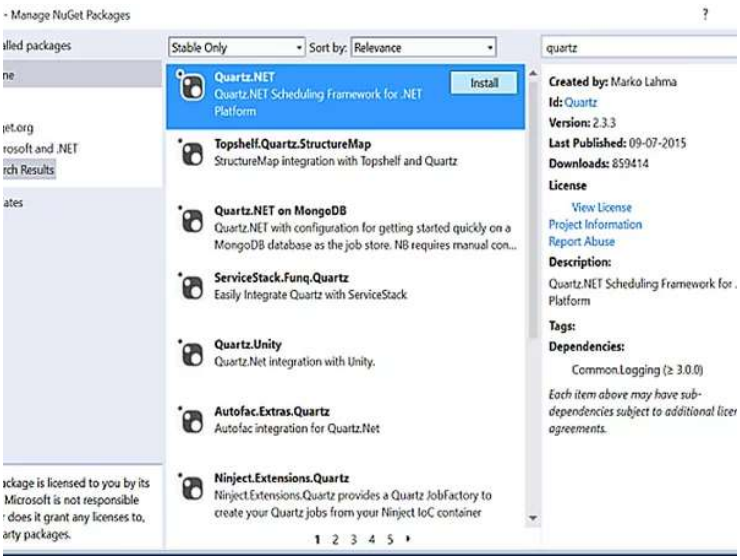
3 enterprise AI success stories



Red Hat OpenShift ramps up security and



Amazon Braket: Get started with quantum computing



How to work with Quartz.Net in C#