



MICROSOFT ARCHITECT

By Joydip Kanjilal, Columnist, InfoWorld
SEP 28, 2020 3:00 AM PDT

How to manage user secrets in ASP.NET Core

Take advantage of user secrets management in ASP.NET Core to prevent the sharing of sensitive application data in your projects

When working with ASP.NET web applications, you will want to protect certain pieces of application data, called user secrets, that should not be shared with others. Your user secrets might include a database connection string that also contains the user ID and the password for the database. You might also want to refrain from sharing information such as access keys, API keys, and connection information details for cloud services such as Azure or AWS.

However, when you share your project with others this secret information also will be shared. How can we prevent this? A feature in ASP.NET Core named User Secrets allows you to store user secrets outside your project tree in a JSON file, and can even be managed using a command-line tool called the Secrets Manager. This article talks about how you can work with the User Secrets API in ASP.NET Core.

[[Also on InfoWorld: What's new in 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 project in Visual Studio 2019

First off, let's create an ASP.NET Core project in Visual Studio 2019. 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.



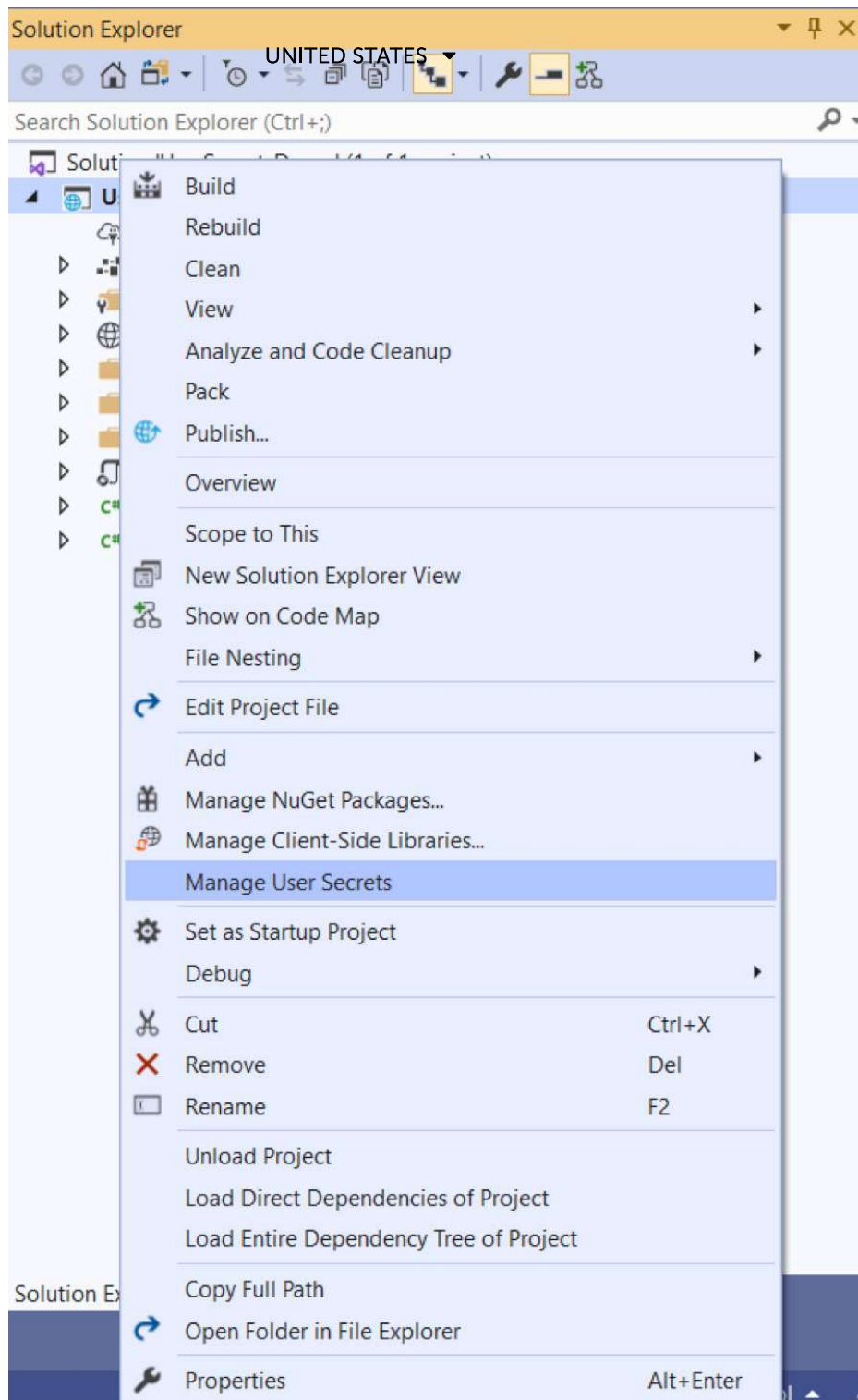
1. Launch the Visual Studio IDE.
2. Click on "Create new project."
3. In the "Create new project" window, select "ASP.NET Core Web Application" from the list of templates displayed.
4. Click Next.
5. In the "Configure your new project" window, specify the name and location for the new project.
6. Optionally check the "Place solution and project in the same directory" check box, depending on your preferences.
7. Click Create.
8. In the "Create a New ASP.NET Core Web Application" window shown next, select .NET Core as the runtime and ASP.NET Core 3.1 (or later) from the drop-down list at the top.
9. Select "Web Application (Model-View-Controller)" as the project template to create a new ASP.NET Core MVC application.

10. Ensure that the check boxes “Enable Docker Support” and “Configure for HTTPS” are unchecked as we won’t be using those features here.
11. Ensure that Authentication is set to “No Authentication” as we won’t be using authentication either.
12. 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 sections below to illustrate how we can manage user secrets in our ASP.NET Core 3.1 projects.

Add user secrets management to your project

Adding user secrets management to your project is fairly straightforward. All you need to do is select the project in the Solution Explorer window, right-click on the project, and then select Manage User Secrets as shown in Figure 1 below.



IDG

Figure 1

This will open the secrets.json file in your Visual Studio IDE. Here's where you can add your secrets as shown below.

RECOMMENDED WHITEPAPERS



HPE Ezmeral Ecosystem: Deploying AI/ML workloads into production in hybrid cloud environments



Eight Things Seasoned NAS Buyers Know

```
{
  "ConnectionString": "This is a test connection string",
  "APIKey": "This is s secret key",
  "AppSettings": {
    "GlobalSettings": {
      "GlobalAccessKey": "This is a global access key!"
    }
  }
}
```

The secret.json file is created at the following location:

C:\Users\joydip\AppData\Roaming\Microsoft\UserSecrets\e4f51d14-ddc1-48f4-bb34-84c114e3d6d0

When you open the .csproj file of your project, you'll notice that a UserSecretsId element has been added as shown in the code snippet given below.

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>netcoreapp3.1</TargetFramework>
    <UserSecretsId>e4f51d14-ddc1-48f4-bb34-84c114e3d6d0</UserSecretsId>
  </PropertyGroup>
</Project>
```

Use the Secret Manager tool in .NET Core

The Secret Manager tool is a command-line tool available in .NET Core for managing your configuration and secret data. In this section we'll examine how we can work with this tool.

Enable user secrets

Type the following command at the command prompt:

UNITED STATES ▼

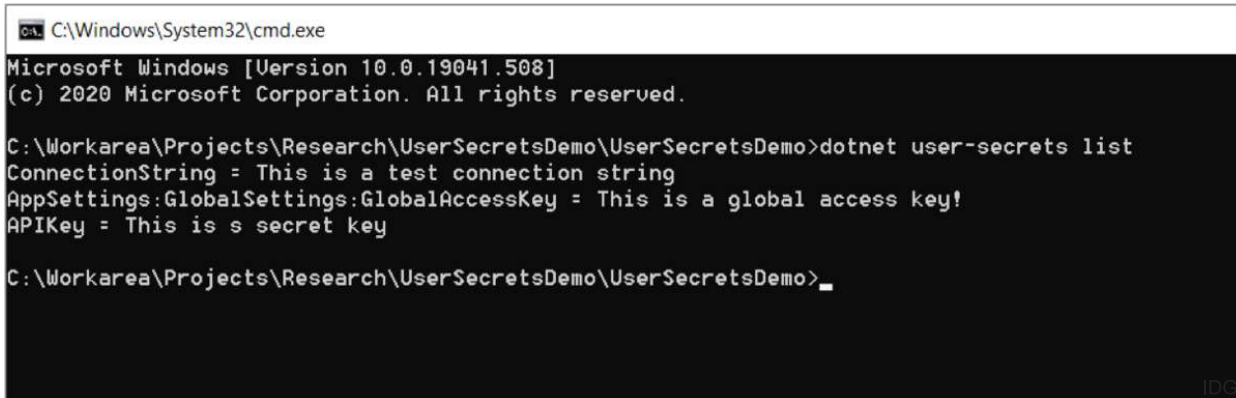
```
dotnet user-secrets init
```

Set a secret

To list the available secrets, you can use this command:

```
dotnet user-secrets list
```

Figure 2 below shows the list of keys we created earlier.

A screenshot of a Windows command prompt window. The title bar shows 'C:\Windows\System32\cmd.exe'. The window content shows the following text: 'Microsoft Windows [Version 10.0.19041.508] (c) 2020 Microsoft Corporation. All rights reserved. C:\Workarea\Projects\Research\UserSecretsDemo\UserSecretsDemo>dotnet user-secrets list ConnectionString = This is a test connection string AppSettings:GlobalSettings:GlobalAccessKey = This is a global access key! APIKey = This is s secret key C:\Workarea\Projects\Research\UserSecretsDemo\UserSecretsDemo>'. The prompt is a white cursor on a black background. There is a small 'IDG' logo in the bottom right corner of the terminal window.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Workarea\Projects\Research\UserSecretsDemo\UserSecretsDemo>dotnet user-secrets list
ConnectionString = This is a test connection string
AppSettings:GlobalSettings:GlobalAccessKey = This is a global access key!
APIKey = This is s secret key

C:\Workarea\Projects\Research\UserSecretsDemo\UserSecretsDemo>_
```

Figure 2

You can use the following command to set a key:

```
dotnet user-secrets set "AuthorApiKey" "xyz1@3"
```

Access a secret

To access user secrets programmatically, you can take advantage of the Configuration API in ASP.NET Core. Let's update the HomeController class to be able to access the configuration data. At first glance, the HomeController class would look like this:

```

public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    public HomeController(ILogger<HomeController> logger)
    {
        _logger = logger;
    }
    //Action methods go here - this is done for brevity
}

```

The following code snippet illustrates how we can update the HomeController class and leverage dependency injection to be able to inject an instance of IConfiguration using constructor injection.

```

public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    private readonly IConfiguration _config;
    public HomeController(ILogger<HomeController> logger,
        IConfiguration config)
    {
        _logger = logger;
        _config = config;
    }
    //Action methods go here - this is done for brevity
}

```

Remove a secret

To remove a key, you can use the following command:

```
dotnet user-secrets remove "AuthorApiKey"
```

If you want to remove all keys, you can use the following command instead:

```
dotnet user-secrets clear
```

Here is a variation on the same command that removes GlobalSettings from the stored secrets:

```
dotnet user-secrets remove "AppSettings:GlobalSettings"
```

The ability to configure, manage, and secure configuration data has been redefined in ASP.NET Core. User Secrets is a great feature in ASP.NET Core that is an excellent alternative to using environment variables. User Secrets ensures that there is no sensitive data included in the source code. Instead, the user secrets are stored outside of the project folder — inside the user's profile folder in the file system.

However, one downside is that the data that is stored by User Secrets is not encrypted. I will discuss other options for protecting user secrets such as Azure application settings and Azure key vault in a future article [here](#).

How to do more 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

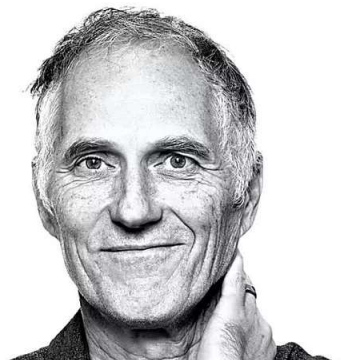
Joydip Kanjilal is a Microsoft MVP in ASP.Net, as well as a speaker and author of several books and articles. He has more than 20 years of experience in IT including more than 16 years in Microsoft .Net and related technologies.

Follow     

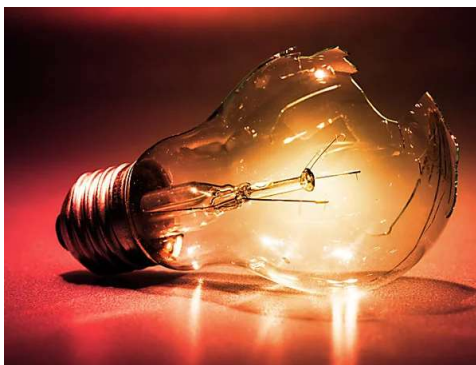
- Stay up to date with InfoWorld's newsletters for software developers, analysts, database programmers, and data scientists.
- Get expert insights from our member-only Insider articles.

YOU MAY ALSO LIKE

Recommended by



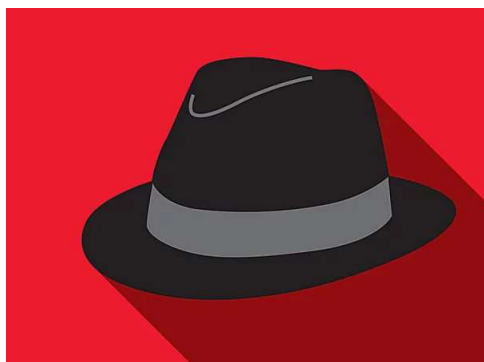
Tim O'Reilly: the golden age of the programmer is over



The decline of Heroku



10 top-notch libraries for C++ programming



Red Hat OpenShift ramps up security and manageability with Platform Plus



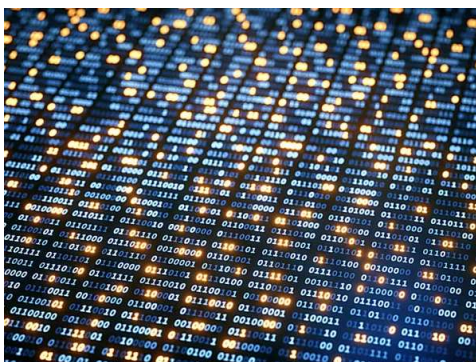
The shifting market for PostgreSQL



Review: 7 Python IDEs go to the mat



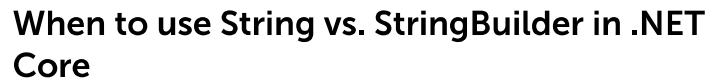
3 enterprise AI success stories



What is a computational storage drive? Much-



Red Hat Enterprise Linux takes aim at edge



dtSearch® instantly searches terabytes of files, emails, databases, web data. See site for hundreds of reviews; enterprise & developer evaluations

Want lightning fast analytics? See why the Incorta data analytics platform is changing enterprise data forever.

2020 was a year of rapid progression of digital transformation for businesses. The following is a snapshot of the digital transformation advancements made across all facets of business.

DDoS extortion attacks are real. Don't Negotiate. Mitigate with NETSCOUT. Learn more.