# How to access views from SQL Server database in ASP.NET Core

## Introduction

This sample demonstrates how to access views from SQL Server database in ASP.NET Core.

## Sample prerequisites

* .NET Core 1.0 or later version(s). [[.NET Core + Visual Studio tooling](http://go.microsoft.com/fwlink/?LinkID=798306)]
* Microsoft Visual Studio 2015 update3 or above. [[Visual Studio 2015](https://www.visualstudio.com/en-us/visual-studio-homepage-vs.aspx)]
* Sql Server 2008 R2 or newer.
* A Sql Server database have under schema and data.

CREATE TABLE [dbo].[Views] (

[Id] UNIQUEIDENTIFIER NOT NULL,

[Path] VARCHAR (200) NOT NULL,

[Content] TEXT NULL,

[CreateTime] DATETIME NOT NULL,

[LastUpdateTime] DATETIME NOT NULL,

[IsDirectory] BIT NOT NULL,

[ParentId] UNIQUEIDENTIFIER NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

INSERT INTO [dbo].[Views] ([Id], [Path], [Content], [CreateTime], [LastUpdateTime], [IsDirectory], [ParentId]) VALUES (N'e1eb6c38-353e-4083-a164-36fd3bbac68c', N'Home', NULL, N'2017-02-23 00:00:00', N'2017-02-23 00:00:00', 1, N'00000000-0000-0000-0000-000000000000')

INSERT INTO [dbo].[Views] ([Id], [Path], [Content], [CreateTime], [LastUpdateTime], [IsDirectory], [ParentId]) VALUES (N'70eb5b67-1c9a-46dc-bde9-54500232ef47', N'Home/Index.cshtml', N'Hello World', N'2017-02-23 00:00:00', N'2017-02-23 00:00:00', 0, N'e1eb6c38-353e-4083-a164-36fd3bbac68c')

## Building the sample

* Deploy you database by previous scripts, and get the connection string.
* Open the sample solution “**CSAccessViewFromSqlServer**” using Visual Studio.
* Open “**appsettings.json**”, local to “ConnectionStrings:ViewsDB” section, and set the value as your database connection string.
* Right click the project “**CSAccessViewFromSqlServer**” and select Restore packages.
* Press **F6 Key** or select **Build -> Build Solution** from the menu to build the sample.

## Running the sample

Open the sample solution in Visual Studio, then press **F5 Key** or select **Debug -> Start Debugging** from the menu.

The sample will be running and loading the view that you have specified.

## Using the code

You should implement interface **IFileProvider** and use it in Startup.cs.

DBViewProvider.cs

public class DBViewProvider : IFileProvider

{

public IDirectoryContents GetDirectoryContents(string subpath)

{

string path = ConvertPath(subpath);

return new DBViewDirectoryContents(path);

}

public IFileInfo GetFileInfo(string subpath)

{

string path = ConvertPath(subpath);

return new DBFileInfo(path);

}

public IChangeToken Watch(string filter)

{

return new NoWatchChangeToken();

}

private string ConvertPath(string path)

{

if (path.StartsWith("/Views/", StringComparison.OrdinalIgnoreCase))

{

path = path.Substring(7);

}

if (path.StartsWith("Views/", StringComparison.OrdinalIgnoreCase))

{

path = path.Substring(6);

}

if (path.StartsWith("/", StringComparison.OrdinalIgnoreCase))

{

path = path.Substring(1);

}

return path;

}

}

DBFileInfo.cs

public class DBFileInfo : IFileInfo

{

private View \_view;

public DBFileInfo(string path)

{

using (ViewDBContext db = new ViewDBContext())

{

this.\_view = db.Views.FirstOrDefault(m => m.Path == path);

}

}

public DBFileInfo(View view)

{

\_view = view;

}

public bool Exists

{

get { return \_view != null; }

}

public bool IsDirectory

{

get { return \_view != null ? \_view.IsDirectory : false; }

}

public DateTimeOffset LastModified

{

get { return \_view != null ? \_view.LastUpdateTime : DateTimeOffset.MinValue; }

}

public long Length

{

get

{

if (\_view != null && \_view.Content != null)

{

byte[] bytes = Encoding.UTF8.GetBytes(\_view.Content);

return bytes.Length;

}

else

{

return 0;

}

}

}

public string Name

{

get

{

if (\_view != null && \_view.Path != null)

{

return \_view.Path;

}

else

{

return string.Empty;

}

}

}

public string PhysicalPath

{

get

{

return string.Empty;

}

}

public Stream CreateReadStream()

{

if (\_view != null && \_view.Content != null)

{

byte[] bytes = Encoding.UTF8.GetBytes(\_view.Content);

MemoryStream ms = new MemoryStream(bytes);

return ms;

}

else

{

return null;

}

}

}

StartUp.cs

public void ConfigureServices(IServiceCollection services)

{

ViewDBContext.ConnectionString = Configuration.GetConnectionString("ViewsDB");

services.AddMvc();

services.Configure<RazorViewEngineOptions>(options =>

{

options.FileProviders.Add(new DBViewProvider());

});

}

## More information

IFileProvider Interface in ASP.NET Core Docs.

<https://docs.microsoft.com/en-us/aspnet/core/api/microsoft.extensions.fileproviders.ifileprovider>