Folke Core Documentation

Release 1.3.0

Arnaud Castaner, Sidoine de Wispelaere

Dec 06, 2017

Contents

Folke is an open-source Web Application/REST API framework written in C# fully compatible with .NET Core. This documentation explains how to build an application using this framework. Folke Core uses several related projects also developped by the Folke team.

- Project page
- File a bug or ask a question

Contents:

CHAPTER 1

Getting Started

Folke.Core is a framework. As such, some design and concept decisions have been made by the developers. For instance, the framework uses Folke.Elm which is an ORM (Object-Relational Mapping) engine built by the team. It replaces EntityFramework or NHibernate. Folke.Core also uses Folke.Identity and Folke.Identity.Server which are implementations of Microsoft.AspNet.Identity and the accompanying controllers.

If you prefer, you can use each of these components separately (ie: use only Folke.Elm) for your own projects. But if you're going to use Folke.Core and save a lot of time by not re-implementing yourself a lot of code we already handled, you will have to use these modules.

These are the modules and technologies Folke.Core uses:

- Folke.Elm
- Folke.Identity
- Folke.Identity.Server
- Webpack
- KnockoutJS

The choices that are up to you are the following:

- The database engine
- The HTML and CSS code (you are free to use any framework you like, such as Bootstrap)

Folke.Core does not support Angular since we use KnockoutJS. We made this choice to keep the framework as light as possible.

1.1 project.json

Create an new empty ASP.NET Core project using your favorite tool (yeoman, Visual Studio, etc.). In the project.json file, add a reference to `Folke.Core` and `Folke.Elm.MySql`:

```
{
 "version": "1.0",
 "title": "Folke",
 "description": "Folke Homepage",
 "dependencies": {
  "Microsoft.NETCore.App": {
  "version": "1.0.0",
  "type": "platform"
  },
  "Microsoft.AspNetCore.Diagnostics": "1.0.0",
  "Microsoft.AspNetCore.Server.Kestrel": "1.0.0",
  "Microsoft.Extensions.Logging.Console": "1.0.0",
  "Folke.Core": "1.3.0.1",
  "Folke.Elm.Mysql": "1.3.0"
 },
 .....
```

Note: Note that if you prefer to use another database driver, Elm also supports PostgreSQL, Microsoft SQL Server and SQLite.

Note: Elm is the ORM developped by the Folke team.

1.2 Startup.cs

Now in the `Startup.cs` file we need to register the various Folke services. This is done in one line in the `ConfigureServices` method:

```
public void ConfigureServices(IServiceCollection services)
{
   services.AddFolkeCore<MySqlDriver>(options =>
   {
       options.ConnectionString = Configuration["Data:ConnectionString"];
   });
}
```

The `MySqlDriver` requires that you use the `Folke.Elm.MySql` namespace. If you need to use another kind of database you must reference the appropriate driver. The `ConnectionString` option is a simple string. In this example it comes from an `IConfigurationRoot` property defined earlier in the code:

```
public IConfigurationRoot Configuration { get; }

public Startup(IHostingEnvironment env)
{
   var builder = new ConfigurationBuilder()
       .SetBasePath(env.ContentRootPath)
       .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
       .AddJsonFile("appsettings.Local.json", optional: true)
       .AddEnvironmentVariables();
   Configuration = builder.Build();
}
```

For reference, this is what the appsettings.json file looks like:

```
{
   "Logging": {
       "IncludeScopes": false,
       "LogLevel": {
       "Default": "Debug",
       "System": "Information",
       "Microsoft": "Information"
       }
   },
   "Data": {
       "ConnectionString": "Server=localhost;Database=someDb;Uid=somUser;
↪Pwd=somePassword;",
       "DefaultAdministratorUserName": "admin@folke.co",
       "DefaultAdministratorPassword": "ThisShouldBeChanged!"
   }
}
```

We now need to add Folke to the request pipeline. For this we need to modify the `Configure` method and add all the types that the framework needs:

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env,␣
↪ILoggerFactory loggerFactory, IFolkeConnection connection,
           RoleManager<Role> roleManager, UserManager<User> userManager,␣
↪ApplicationPartManager applicationPartManager)
       {
           loggerFactory.AddConsole();
           app.UseMvc();

           app.UseFolkeCore(connection, env, roleManager, userManager,␣
↪applicationPartManager, options =>
           {
               options.AdministratorEmail = Configuration[
↪"Data:DefaultAdministratorUserName"];
               options.AdministratorPassword = Configuration[
↪"Data:DefaultAdministratorPassword"];
           });

           if (env.IsDevelopment())
           {
               app.UseDeveloperExceptionPage();
           }

           app.Run(async (context) =>
           {
               await context.Response.WriteAsync("Hello World!");
           });
       }
```

The part of this code that is relevant to Folke is the `app.UseFolkeCore()` block. The section below describes the required parameters.

```
IApplicationBuilder.UseFolkeCore(IFolkeConnection connection, IHostingEnvironment env,
↪ RoleManager<Role> roleManager, UserManager<User> userManager,␣
↪ApplicationPartManager applicationPartManager)
```

1.2.1 IFolkeConnection

This object manages the database connection and transactions state. This is a compoment of Folke.Elm, the ORM the framework uses.

1.2.2 IHostingEnvironment

This object holds the hosting environment options (path, development or production environment, etc).

1.2.3 RoleManager<Role>

This class is used to manage roles like modifying one or creating a new one. The `Role` class inherits from `IdentityRole` of Folke.Identity.Elm which is an implementation of Microsoft.AspNet.Identity developped to work with Folke.Elm.

1.2.4 UserManager<User>

This class is used to manage users like modifying one or creating a new one. The `User` class inherits from `IdentityUser` of Folke.Identity.Elm which is an implementation of Microsoft.AspNet.Identity developped to work with Folke.Elm.

1.2.5 ApplicationPartManager

This class is used to easily register the Folke.Identity.Server controllers (such as AuthenticationController) in your application.