Evelyn Documentation

Release 0.0.0

Philip Wood

May 29, 2018

Getting started

CHAPTER 1

Overview

You probably want to see Evelyn in action. In this section we'll describe how to get the code, build and run it. This shouldn't take more than a few minutes.

CHAPTER 2

Getting the code

The Evelyn repository is hosted on Github. You can clone the repo with one of the following commands:

2.1 Using HTTPS

```
git clone https://github.com/binarymash/evelyn.git
```

2.2 Using SSH

```
git clone git@github.com:binarymash/evelyn.git
```

We use GitFlow branching; development occurs on the default develop branch, and stable releases are on the master branch.

CHAPTER 3

Building the code

Evelyn can be built on the command line on any environment that supports the .NET Core SDK and, if you're running on Windows, in any version of Visual Studio 2017.

3.1 Building on the command line

This project has automated build scripts that will compile the code and run the test suite. The build scripts are written using Cake. These scripts can be used by developers, and are also used in the project's build and release pipeline in AppVeyor.

- Ensure that you have the latest .NET Core SDK installed
- Run the appropriate build script for your development environment
  - `./build.ps1` (Powershell)
  - `./build.sh` (Linux/macOS shell)

The build will take a few moments. The outputs of the build (nuget packages, test results etc) will be published to the `./artifacts` folder. You might need to have administrator rights to run some of the tests.

3.2 Building in Visual Studio

- Open Visual Studio 2017 with administrator rights (you'll need this to run some of the tests, and when running the server in debug)
- Open the `./src/Evelyn.sln` solution
- Build the solution

CHAPTER 4

Running the REST Server

A sample server host is provide for the REST API - this can be found at `./src/Evelyn.Host`. By default this will run with an in-memory event store. You won't want to run this host in a production environment, but it's useful for now because there are no dependencies to set up and nothing to configure.

You can run the sample host in Visual Studio or on the command line.

4.1 Running in Visual Studio

Run the `Samples\Evelyn.Host` project

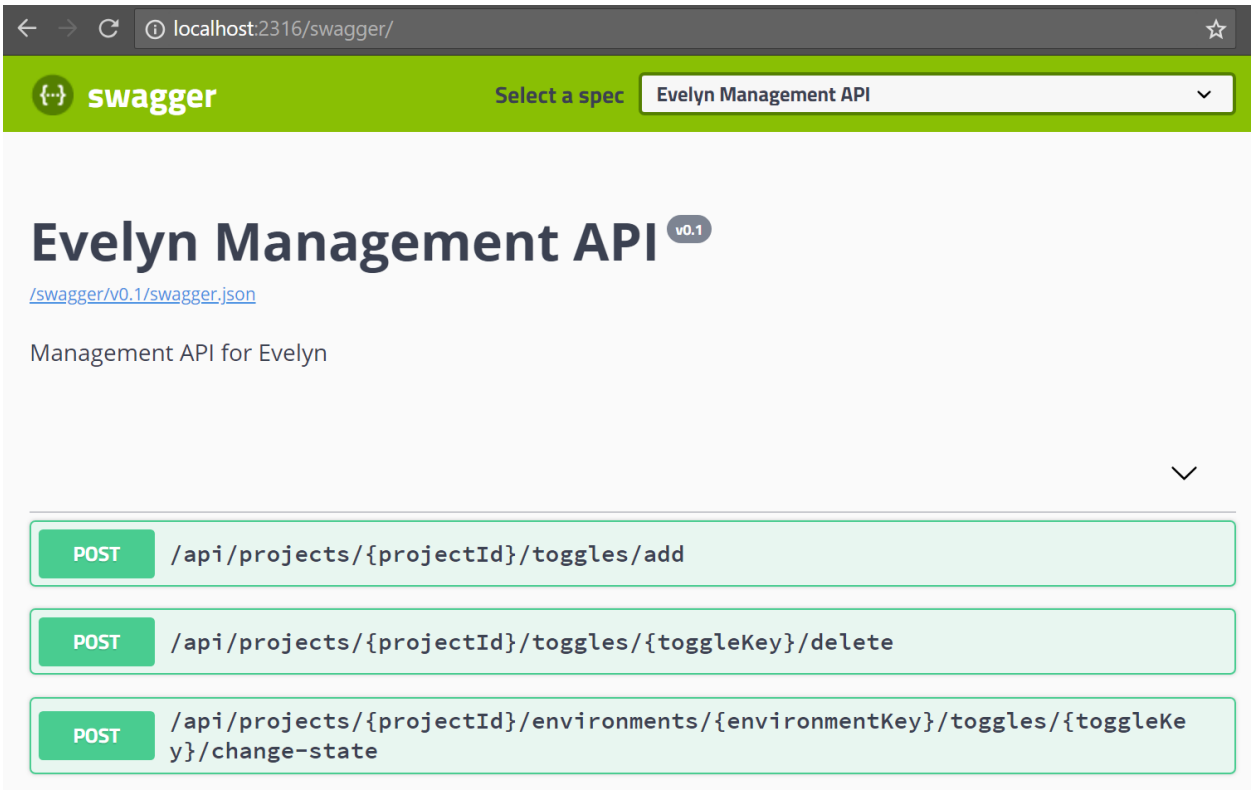
4.2 Running on the command line

```
dotnet .\src\Evelyn.Host\bin\Release\netcoreapp2.0\evelyn.host.dll
```

4.3 Accessing the REST Server

By default, the sample server host will listen on localhost:2316 - this is configured in `./src/Evelyn.Host/Properties/launchSettings.json`.

The Evelyn REST API endpoints are specified using OpenAPI. The sample server host is configured to show us this specification using Swagger UI. We can access the Swagger UI in a browser by navigating to `/swagger/` endpoint of the server. So, if we're using the default config, this will be at `http://localhost:2316/swagger/`.



When the server runs for the first time, it will set up a default account for us and add create some sample data to get us started. Lets check it is all set up correctly:

- In Swagger UI, expand the *GET /api/projects* section
- Click the *Try it out* button
- Click the *Execute* button

In Evelyn, a project is a logical collection of feature toggles and environments. The `/api/projects` endpoint returns us a list of all the projects on our account. When we click the *Execute* button, Swagger will make a call to this endpoint. The response should look something like this:

```
{
   "accountId": "e70fd009-22c4-44e0-ab13-2b6edaf0bbdb",
   "projects": [
      {
         "id": "8f73d020-96c4-407e-8602-74fd4e2ed08b",
         "name": "My First Project"
      }
   ],
   "created": "2018-05-27T15:58:13.6253741+00:00",
   "createdBy": "SystemUser",
   "lastModified": "2018-05-27T15:58:30.7611496+00:00",
   "lastModifiedBy": "SystemUser",
   "version": 1
}
```

We can see here that our account ID is `e70fd009-22c4-44e0-ab13-2b6edaf0bbdb`, and we have a project called `My First Project` which has the ID `8f73d020-96c4-407e-8602-74fd4e2ed08b`.

Now we know the ID of the project, lets now get more details about it:

- Expand the *GET /api/projects/{id}* section
- Click the *Try it out* button
- In the *id* input box, enter the id of the project, `8f73d020-96c4-407e-8602-74fd4e2ed08b`
- Click the *Execute* button

The response should look something like this:


```
{
   "id": "8f73d020-96c4-407e-8602-74fd4e2ed08b",
   "name": "My First Project",
   "environments": [
      {
         "key": "my-first-environment"
      }
   ],
   "toggles": [
      {
         "key": "my-first-toggle",
         "name": "My First Toggle"
      }
   ],
   "created": "2018-05-27T15:58:30.7715006+00:00",
   "createdBy": "SystemUser",
   "lastModified": "2018-05-27T15:58:30.8970043+00:00",
   "lastModifiedBy": "SystemUser",
   "version": 2
}
```


We can see that the project has a single environment, `my-first-environment` and has one toggle, `my-first-toggle`.

So far so good. Now lets turn our attention to the client.

CHAPTER 5

Running the client

A sample client application is included - this can be found at `./src/Evelyn.Client.Host`. This client is already configured to point at the default location of the sample server host, and

You can run the sample host in Visual Studio or on the command line.

5.1 Running in Visual Studio

Run the `Samples\Evelyn.Client.Host` project

5.2 Running on the command line

```
dotnet .\src\Evelyn.Client.Host\bin\Release\netcoreapp2.0\evelyn.client.host.
dll
```

5.3 Using a toggle

An application that uses the Evelyn client must be configured to connect to the server to retrieve the current toggle states for a particular enviroment and project.

The sample client host is already configured to get the toggle state for the sample project and environment that was created when we started the server; you can find this configuration in `.\src\Evelyn.Client.Host\Startup.cs`. Note that in this class we also start a background service, which is used to poll the server for the current state.

Now, take a look in the `OutputWriter` class. You'll see we're injecting an `IEvelynClient` in the constructor. This interface lets us access the toggle states for our chosen environment. We use the `GetToggleState` method on this interface to get the current state of our `my-first-toggle` toggle, and then use this to decide which block of code to execute.

Lets run the client application. We'll get this output:

```
Hit enter to continue, or any other key to exit
```

Hit enter a few times...

```
This code is called when the toggle is OFF
This code is called when the toggle is OFF
This code is called when the toggle is OFF
This code is called when the toggle is OFF
This code is called when the toggle is OFF
```

It's clear from this that our execution path is that specified for when the toggle is turned off.

5.4 Changing toggle state

Now, lets change the state of our toggle. We can do this either through the Swagger UI or via the Evelyn Management UI (if you've set it up):

5.4.1 Changing toggle state in Swagger UI

- Expand the *POST /api/projects/{projectId}/environments/{environmentKey}/toggles/{toggleKey}/change-state* section
- Click the *Try it out* button
- In the *projectId* input box, enter the id of the project, `8f73d020-96c4-407e-8602-74fd4e2ed08b`
- In the *environmentKey* input box, enter the key of our environment, `my-first-environment`
- In the `toggleKey` input box, enter the key of our toggle, `my-first-toggle`
- In the *message Body* input box, enter this:

```
{
  "expectedToggleStateVersion": 0,
  "state": "True"
}
```

- Click the *Execute* button

5.4.2 Changing toggle state in Evelyn Management UI

- From the dashboard, select *My First Project*
- Select *my-first-environment* from the list of environments
- Find *my-first-toggle* in the list of toggles, and click its icon to change the state from `OFF` to `ON`

Return to the client application and hit enter a few more times...

```
This code is called when the toggle is ON
This code is called when the toggle is ON
This code is called when the toggle is ON
This code is called when the toggle is ON
This code is called when the toggle is on
```

Now, we're going through the other code block! So, in changing the toggle state, we've changed the behaviour of our application.

CHAPTER 6

Running the management UI

The Swagger UI is convenient for us as developers, but not particularly great for end users. Happily, Evelyn also has a management application that we can use to manage our toggles.

The code for the management UI is in a separate repository. If you've not already had a look, now might be a good time - Read the docs.

CHAPTER 7

Introduction

Evelyn is a feature toggling framework. It allows users to decouple software releases from the functional changes within, reducing the risk of deployment and providing rollback functionality.

The Evelyn Stack consists of the following parts:

- A core framework providing the underlying feature toggling functionality, written in C# and targetting .NET Standard 2.0
- A ReST API server and client that expose this functionality over HTTP, written in C# and targetting .NET Standard 2.0. Sample hosts are provided for .NET Core 2.0.
- A management user interface, built on React/Redux/Node.

Evelyn has a modular architecture which allows for flexible deployment configurations and user extensibility. The core framework is built around CQRS and Event Sourcing: implementations are provided for an in-memory event store and for Greg Young's Event Store; you can plug in your own event store integration.

This project is pre-release: things might break at any moment; APIs might change; it is insecure.

CHAPTER 8

This documentation

This documentation is for the core framework, REST API and client.

For more information on the management UI head over to https://evelyn-management-ui.readthedocs.io/en/latest/.