Coin Trader by Peter Pellegrini README

Overview

Coin Trader displays an updating value of a cryptocurrency coin as well as the percentage change since the last value was retrieved.

Coin Trader can be viewed on Azure at https://gentle-mushroom-067b88b00.1.azurestaticapps.net

It is hosted on GitHub under https://github.com/pfxp/cointest

Architecture

Coin Trader is comprised of a ASP.NET Core Web API backend and an Angular SPA frontend.

The backend core layer is CoinTrader.Shared which contains data classes, enumerations and services. There is a corresponding xUnit test *CoinTrader.Shared.Tests*.

CoinTrader.Shared is used by *CoinTrader.API*. *CoinTrader.API* is the backend of the coin trading platform. Two endpoints are present and Swagger/SwaggerUI is enabled in development builds.

Once this was running I was also able to test it with Postman.

CoinTrader.Api.Client contains NSwagStudio v13.15.9 definitions to create C# and TypeScript clients. The C# client is used by the xUnit test *CoinTrader.Api.Client.Tests*, and the TypeScript client is used by the Angular frontend.

The generated typescript client makes good use of rxjs eg. observables.

The frontend is an Angular SPA that calls the backend. An Angular module *price* contains the relevant components of this coding challenge.

Prerequisites

- .NET Core 6
- Visual Studio 2022
- Node.js v16.13.1 or later
- Angular CLI v13.1.1 or later

Features

- The backend only does data processing so does not need to generate views like a MFC app.
- Efficient socket utilisation with IHttpClientFactory.
- Injects services where necessary.
- An Angular Material Button Toggle is used for coin selection so only a valid coin can be chosen.

- If the backend can't communicate with https://trade.cointree.com/api/prices/aud/{coinsymbol} the UI will update with empty values.
- Price precision goes to 8 decimal places and uses a currency price for correct symbol, thousands separator.
- Frontend is decoupled from the backend API using a swagger endpoint and NSwagStudio to generate backend clients.
- Easy to test the backend with cURL, Postman, jMeter
- Should be easy to scale out/up the backend.

Todo/Things to improve if I had more time

- Implement a GetPreferredCoin backend endpoint so that after a F5 refresh the user receives the preferred coin from the backend so the button toggle can be set correctly. Will also fix the problem where the frontend assumes the backend is set to BTC.
- Angular component tests have not been implemented.
- Tighten the CORS policy. Currently you can call the backend from any origin.
- Get rid of vertical scrollbar.
- Unit tests are not configurable and assume the backend is running under VS2022.
- The frontend hits the backend on a 1 second timer. In future the backend should detect when the price updates and use SignalR or similar to update the connected frontends.
- Better typography and layout

How to run

The backend was developed in VS 2022 and the frontend in VS Code.

CoinTrader has been deployed to Azure but can also run from a developer's machine.

Azure

Go to https://gentle-mushroom-067b88b00.1.azurestaticapps.net/price

Developer machine

Note: '~' denotes the home folder of the cloned repository.

To start the backend server

From Visual Studio...

Open `~\src\backend\Coin.sln` solution, and run CoinTrader.API as Debug|AnyCPU.

OR from the command line...

```
cd ~/src/backend/API/CoinTrader.Api
dotnet run --launch-profile CoinTrader.Api
Then in a browser, navigate to https://localhost:7134/swagger
```

To start the Angular frontent

To prepare the frontend (once-off):

```
cd ~\src\frontend\coin-trader-web
npm install
```

To start the frontend:

```
cd ~\src\frontend\coin-trader-web
ng serve --open
```

This can then point a browser at http://localhost:4200/

Backend

VS2022 workspace at ~/src/backend/Coin.sln

Frontend

VS Code workspace at ~/src/frontend/coin-trader-web.code-workspace

xUnit tests

To run the tests from the command line:

```
cd ~\src\backend\Tests\oinTrader.Shared
dotnet watch test
```

and

```
cd ~\src\backend\Tests\CoinTrader.Api.Client\CoinTrader.Api.Client.Tests
dotnet watch test
```

Timings

- Creating the backend solution with basic skeletons for backend API, xUnit test project, Swagger Client project, Angular project skeleton, .gitignore, making a start on the README.me: 4 hours
- Configurable source API for real pricedata, using IHttpClientFactory and Newtonsoft to parse JSON: 1 hour.
- Fleshing out some unit tests, refactoring services outside the CoinTrader.API service: 0.75 hour.
- Regenerating the API clients and adding the API Client xUnit test: 0.5 hour.
- Cleaning up the skeleton Angular project and documenting the architecture: 0.5 hour.
- Creating Az Powershell scripts to make backend site in Azure, setting up CORS in backend, tweaking NSwag configurations, configuring launch.json so I can reliably run Chrome with the frontend from VS Code, creating Angular components to display prices, setting up routing module, configuring the CoinTraderApiClient (generated by NSwagStudio) to be injected, getting first calls to backend to work: 4 hours.
- Creating the backend comms service, implementing an observable to send the ask value and change to pricing components: 2 hours
- Fiddling with .browserslistrc to work around an Angular bug: 0.5 hours
- Configuring backend to return enums in swagger as strings so CoinType has values [BTC,ETH,XRP], updating components to use CoinType correctly, set the coin type on the backend: 1 hour.
- Fixing routing: 0.5 hours
- Tweaking user interface to make it clearer: 0.5 hours
- Configuring fallback routes, extra documentation.