Benchmark Figures Documentation:

The Benchmark Figures application is an application that reads price list figures and benchmark history figures from 2 separate csv files , then calculates portfolio performances and then writes a report into a csv file.

Assumptions:

- The csv files use a comma delimiter.
- Benchmark performance start date equals the first day of the month
- Benchmark performance end date equals last day of the month
- The user can only switch using start of the month dates.
- Benchmark codes/ Instrument codes are not case sensitive
- All dates use the “yyyy/MM/dd” format
- All path files must be valid

How to run the application

Run the console project “BenchmarkFigures.ConsoleApp” which will prompt the user for input, please follow the instructions.

The application will ask for the path of the pricelist file and benchmark history file. It is assumed that the files given are valid and the app has access to read the files. These paths will be stored in memory and can be reused to generate all the reports.

The application will show the user multiple options on the type of report to run and will further prompt the user for input based on the chosen report.

After it is done, it will print that the print that “YOUR REPORT IS READY...” and asks notify the user that they can run another report.

The report will be located at the path given by the user, and will be named this this format:

PortfolioReport_yyyyMMddHHmmss.csv Example *PortfolioReport_20230204124115.csv*

Application details

All projects target .Net 6.0. and written in the C# programming language. Tests use Microsoft.VisualStudio.TestTools.UnitTesting

The application applies a variation of DDD Architecture (for scalability and maintainability). The **Domain** layer has the 2 entities Price and BenchmarkPerformance and a Dto, PriceDifferenceDto for storing the calculated price differences.

The **Infrastructure** layer contains a CsvHelper class that reads from the csv files, and creates the entities dynamically. The CsvHelper is also used for writing a Csv formatted string into a Csv file.

The **Domain Services** layer houses all the business logic processing i.e. price differences, benchmarks and portfolio performances. The method *`CreateBenchmarkForOneOrMoreInstruments`* is the only use case available as the functioning of the application is just a variation of calculation benchmarks with one or more instrument codes and because of this, No Use Case layer/ Application layer was created because all use cases would just be calling this function and then creating a report of the benchmark performances returned.

A **console application** has been used for interacting with the user. All conversions/validation of the input is done in console application.

Tests:

The only tests created are for methods that perform calculations.