DriveSizeVisualizer

Project by Maximilian Wahl as part of the course "Parallel and Asynchronous Programming with .NET"

Introduction

DriveSizeVisualizer is a simple .NET Maui App that can be used to view the tree-like structure of a Hard-Drive or any directory in the Filesystem of a PC.

Unit of Work

Heart of the Application is a recursive algorithm which goes through the file system and gathers file and directory information. This algorithm can be switched between Two Modes:

- Sequential : The algorithm traverses the tree breadth first in a single Thread, or
- Parellel : The algorithm uses the .NET Parallel Programming Library to traverse the tree, splitting subdirectories into separate tasks. This does not however, bring a giant performance improvement. See the last Section for more detail.

Features

Here are some extra features which this program provides.

Sort

The tree can be sorted either by name or size of the files and directories. This is implemented with LINQ Queries using a custom Built Expression Tree, because the Sortable properties of the tree are defined by a custom expression, which then gets read at runtime.

Filter

Files can be filtered by both name and file type/extension. This filter is dynamically generated after loading the tree. Directories which contain no filtered files will not be displayed.

RenderDepth

As .Net Maui does not have an Official TreeView Component, I used a Third-Party Component which seems to not be very optimized. As a result, one can set a RenderDepth to stop the tree from getting rendered until a certain depth. Of course one can traverse the tree up and down and will see the correct section of the tree.

MultiThreaded IO

Under normal circumstances, the divide and conquer approach utilised by this application benefits massively from parallel execution of the tasks. However, in this specific application this is not the case because IO and multithreading do not work well together, because in the end, IO operations get serialized by the OS and will get executed one after another, killing some of the performance Parallel execution brings.