MultiThreadFileAccess - Developer Documentation

Overview

The MultiThreadFileAccess application demonstrates concurrent updates of a file in multithreaded environment. This application demonstrates that up to 10 threads can write 10 times without causing any side effects. The run can be executed multiple files but file generation will be consistent. The application can be executed in docker or Windows command line.

Solution structure

Solution consists of:

- **Program.cs** : This is entry point to the application. Based on if the application is executed from Docker environment or other leverages different root directory to save generated file
- **Helper/Constants.cs**: This defines various constant values used across the application in one single file.
- **Interfaces/IFilewriter.cs**: This interface defines various methods leveraged to write to file. We only have one class implementing this but can be enhanced to have more if different behavior needed
- **Core/DirectorManager.cs**: Directory Manager class is responsible for checking if directory exists and if not to create the same.
- **Core/FileWriter.cs** : FileWriter class is responsible to write to file and implements the IFileWriter interface. Few things to note
    - WriteFirstLine: This method will overwrite any file contents if they exist.
    - WriteLine: This method will append to existing contents of the file. This method also leverages file lock so that at any given time only one thread can write to file avoiding any race conditions or false date
- **Core/FileManager.cs**: This is the main orchestrator for the application and leverages all code to have multiple threads write to same file successfully.
- **DockerFile :** This represents to steps to create docker container which includes:
    - Build and copy code to runtime image container
    - Set environment variable to 'DOCKER' so that correct root directory is used
    - TImezone package installed and set to EST so that container uses Eastern timezone instead of UTC.

- **Tests**: The tests folder has test classes which include unit tests for all the above classes.

Features

Thread Handling

Threads are implicitly created by leveraging tasks. The program utilizes `Parallel.For` to execute file-writing tasks in parallel. Each thread runs executes to write to same file concurrently.

Environment-Specific Directory Handling

This application supports executing on windows or docker

- The application checks the environment type (Docker or Windows) using the `ENVIRONMENT` environment variable.
- Based on the environment, the program sets the correct root directory (`/log` for Docker and `c:/junk` for Windows), ensuring that file paths are correctly configured for each environment.

Error Handling

All the exceptions are logged to Console.

Output

The output file has format as follows

0, 0, 21:27:43.228

1, 7, 21:27:43.287

The data represented is:
- First digit represents line count

-Second digit represents the ThreadId

-Third field is timestamp, this is currently in EST timezone.

Execution

Windows Instructions

- Open Powershell window or command prompt
- Navigate to directory which has the executable
- Run the MultiThreadFileAccess.exe

Docker Instructions

- Pull docker image

```
docker pull deepaliap/multithreadfileaccessapp:latest
```

- Execute the docker command

```
docker run -i -v c:/junk:/log deepaliap/multithreadfileaccessapp
```

File is generated successfully in C:\junk