Hasher

Thomas A. McKernan

What is Hasher?

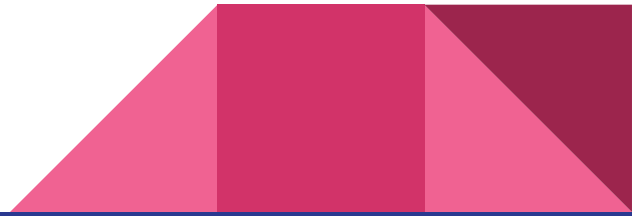
- Command line tool to save hashes of files
- Can be used to for security, verification, or troubleshooting

Implementation

- Kotlin CLI
  - Null Safety
  - Data Classes
  - JVM MessageDigest
- External Dependencies
  - Kotlinx Coroutines
  - Kotlinx Serialization
  - PicoCLI

How it works

- Hasher stores hashes in a JSON file that you specify the location of via an environment variable
- You are prompted to set this variable if it does not exist
- Default is $HOME/.hasher.json



```
~/Desktop/hasher | master#0.3  cat ~/.hasher.json | jq
{
  "hashes": {
    "test1": {
      "files": {
        "/Users/tmckernan/Desktop/hasher/README.md": "1593da608562a5f642739550fa042cc644c26af6"
      },
      "options": {
        "includeWhitespace": false,
        "includeTimestamp": false,
        "algorithm": "SHA-1"
      }
    },
    "test": {
      "files": {
        "/Users/tmckernan/Desktop/hasher/README.md": "4ce216f97b573ab8a06e8d3b20c68784"
      },
      "options": {
        "includeWhitespace": false,
        "includeTimestamp": false,
        "algorithm": "MD5"
      }
    }
  }
}
~/Desktop/hasher | master#0.3
```

```kotlin
package hasher

import kotlinx.serialization.Serializable

@Serializable
data class SettingsFile(
        val hashes: MutableMap<String, HashStore>
)

@Serializable
data class HashStore(
        val files: MutableMap<String, String>,
        val options: Options
)

@Serializable
data class Options(
        val includeWhitespace: Boolean,
        val includeTimestamp: Boolean,
        val algorithm: String
)
```

Kotlinx Coroutines

```
private fun handleFileOrDir(f: File): Map<String, String> {
    return runBlocking { this: CoroutineScope
        val fileQueue = LinkedList<File>()
        fileQueue.push(f)

        val jobList = mutableMapOf<String, Deferred<String>>()

        while (fileQueue.isNotEmpty()) {
            val currFile = fileQueue.pop()
            if (currFile.isFile) {
                jobList[currFile.absolutePath] = async { handleFile(currFile) }
            } else if (currFile.isDirectory) {
                fileQueue.addAll(currFile.listFiles()!!)
            }
        }

        println("Hashing...")
        return@runBlocking jobList.map { (key, value) ->
            key to value.await()
        }.toMap()
    }
}
```

Deployment

- Available on Homebrew through my tap

Additional features

- Having an option to ignore hidden files and directories
- Having a regex ignore option for text within files (e.g. comments)
- Add a max depth option for directory recursion
- Add global presets in JSON file