Controller

AbstractController

View: view

AbstractController (View, view)

CommandController

- workModel : WorkModel
- backupModel : BackuModel
- commandProcessor : Processor
- + CommandController (View view, WorkModel workModel)
- + Process (string command): void
- + showHelp (string translationKey, string parameters): void
- + exitToMenu () : void
- + launchGrouped (List<string> listLabels, bool threaded) : void
- + launchIndividual (string label): void
- + updateProgress(Work work) : void
- + clearConsole(): void
- + ExceptionHandler(string message): string

MenuController

- backupModel : BackupModel
- + MenuController (View view, BackupModel backupModel)
- + getAvailableBackup (): List<Backup>
- + castIntoBackupType (string inputEnum) : BackupType
- + createBackup (string inputLabel, string inputSource, string inputDestination, string inputEnum): void
- + editBackup (string inputSource, string inputDestination, string inputEnum, Backup backup): void
- + getBackup (string label) : Backup
- + deleteBackup (string label) : void
- + checkElementExists (string input, string mode) : string

View

View

- input : string

- viewOptions : List<ViewOption>

- oldMenu : List<ViewOption> - translationManager : TranslationManager

- menuController : MenuController

- commandController : CommandController

- fileCountTranslation : string

fileSizeTranslation : string

- statusTranslation : string

- labelTranslation : string

+ View (TranslationManager translationManager)

+ initView () : void

+ SetMenuController (MenuController menuController) : void

+ SetCommandController (CommandController commandController): void

- WriteMenu (List<ViewOption> options, ViewOption selectedOption) : void

- ask (string msg) : string

+ WriteTemporaryMessage (string message, [string message, [int sleepTime = 1000]) : void

+ WriteTemporalyMessageAndConsole (string message, int sleepTime = 1000) : void

- SwitchMenu (string newMenu) : void

- generateListBackupSubMenu () : void

- generateLanguageSubMenu () : void

- generateBackupSubMenu () : void

- changeLanguage (string locale): void

- updateMenus (): void

- backOldMenu () : void

+ consoleHandler (): void

- createBackup () : void

- editbackup () : void

deleteBackup () : void

- invokeActionAndHandleExceptions (Action action, string success, string exception) : void

- writeAndCheck (string showQuestion, Func<string>, verificationAction, [string baseValue) = null]) : string

+ showCommandHelp (string translationKey,string parameters) : void

- addProgress (string value, string translation, int positionView) : void

+ initiateProgressTranslation(): void

+ updateProgress(Work work) : void

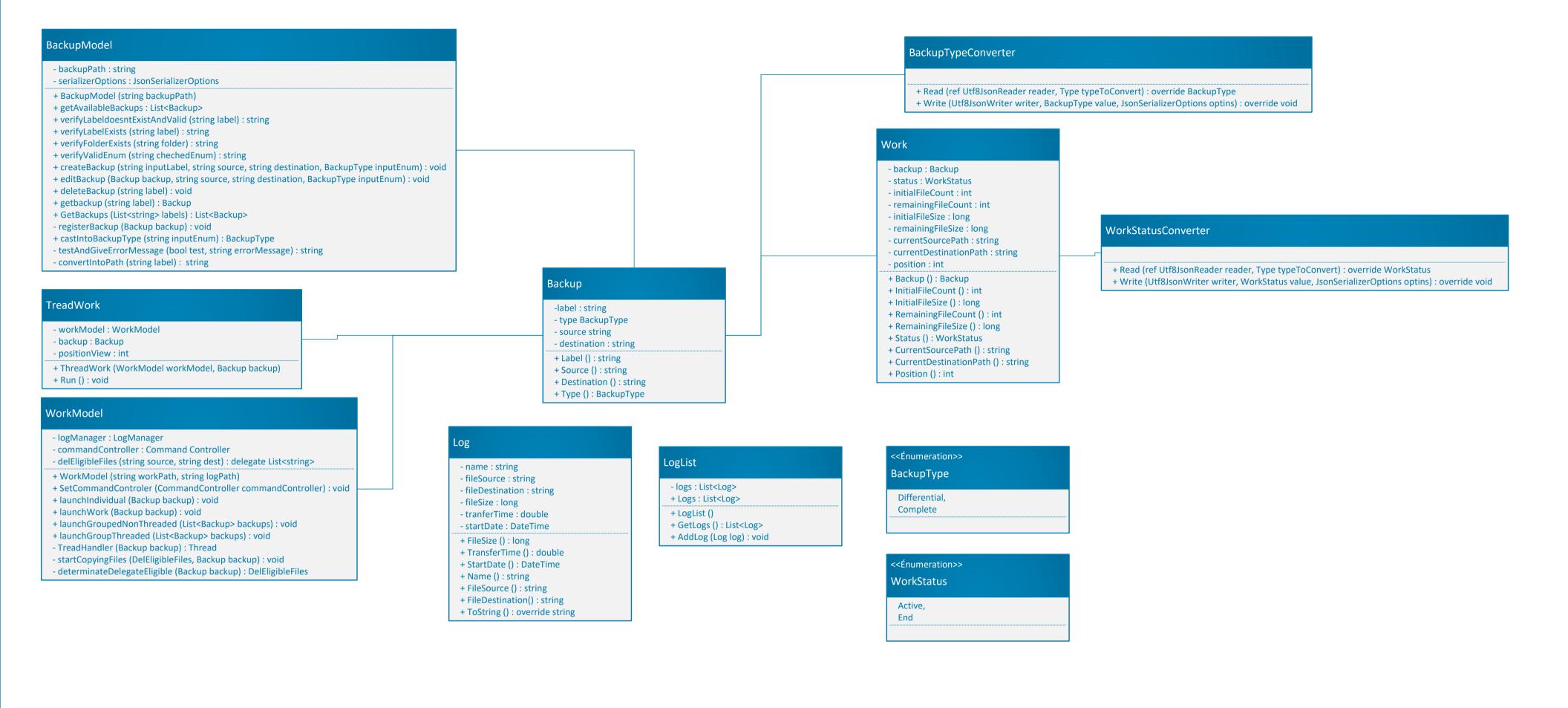
+ initiateProgressBar() : void

+ ClearConsole(): void

ViewOption

- + Name {get; set; } : string
- + Key {get;} : string
- + Selected { get: } : Action
- + ViewOption (string name, Action selected)
- + ViewOption (string name, string key, Action selected)

Models



ssingEnvVariableException: Exception

missingtranslationLocation : string

 missingbackupLocation: string missingworksLocation: string

- missingbaseLocale: string missingenvPath: string

nvFileManager

- translationLocation : string

+ TranslationLocation : string

+ EnvFileManager(string envPath)

determinateWhichEnvFile(string path) : stirng

- openEnvFileAndGiveVariables (string proportiesPath) : void

getVariable(JsonElement data, string varName) : string

- getPathEnvVariable(JsonElement data, string label) : string

+ WorksLocation : string

+ LogLocation : string

+ BaseLocale : string

- backupLocation : string

worksLocation : string

logLocation : string

baseLocale : string

envPath : string

 logexception: string + Logexception(string missingEnvVariableException) : string

+ Checkmissingvariable(string translationLocation, string backupLocation, string worksLocation, string logLocation, string baseLocale, string envPath): void

AvailableEnvFilesException: Exception

NoAvailabletranslationLocation: string

 NoAvailablebackupLocation : string NoAvailableworksLocation : string NoAvailablebaseLocale : string

 NoAvailableenvPath: string logexception : string

- Logexception(string NoAvailableEnvVariableException) : string

logManager : LogManager

timestamp : string

+ Run() : void

eLoggingThread

+ Run() : void

- logManager : LogManager

_logFileQueue : BlockingCollection<Log>

- _workFileQueue : BlockingCollection<Log>

+ FileLoggingThread(BlockingCollection<Log> logFileQueue, LogManager logManager)

-nomMembre

+ getAvailableFileNamesRegex(string path, string regex) : List<string> + getBackup(string backupFilePath) : Backup

+ getLogList(string fullPath) : LogList

+ getWorkList(string fullPath) : Dictionary<string, List<Work>> + getStream(string filePath) : string

+ doFileExist(string path) : bool

+ WriteAndSaveFile(string path, string content): void + deleteFile(string path): void

+ checkIfValidFileName(string filename) : bool + getCompleteFiles(string source, string destination) : List<string>

+ getDifferentialFiles(string source, string destination) : List<string> + copyFile(string source,string destination): void

+ WorkLoggingThread(BlockingCollection<Work> workFileQueue, LogManager logManager, string timestamp)

 WorkFileName : string logPath : string workPath : string

.ogManager

logFileName : string logFileQueue : BlockingCollection<Log>

workFileQueue : BlockingCollection<Work> loggingThreadHandle : Thread

workingThreadHandle : Thread - serializerOptions : JsonSerializerOptions

+ LogManager(string workPath,string logPath)

+ initiateLogging() : void + stopLogging(): void

setAndCreateLogFile() : void setAndCreateWorkFile(): void

initiateWorkFile() : string

initiateLogFile() : string SerializeLogList(LogList logList): string

SerializeWorkList(Dictionary<string, List<Work>> workList): string

+ appendLogToLogList(Log log): void + appendWorkToWorkList(Work work) : void

+ updateLogFile(LogList logList) : void

+ updatedWorkLogFile(Dictionary<string, List<Work>> workList) : void

+ getLogList() : LogList

+ getWorkList() : Dictionary<string, List<Work>>

- missingtranslation : string + missingtranslationexception(): string

lissingTranslationExeption:

- jsonContent : JsonElement local : string + Translation(JsonElement jsonContent, string locale) + Local(): string + Translate(): string

slationManager

translation : Translation

- translationLocation : string + TranslationManager(string translationLocation, string locale)

+ Translate(string key, params string[] args) : string

+ getAvailableTranslations() : List<string>

ranslation

- getTranslation(string newLocale) : Translation? + ChangeTranslation(string newLocale): void

- VoidHandleTranslationChangeException(Exception exception, string newLocale): string - FindAndReturnTranslation(string locale): JsonElement

ommandArguments

commandName : string - aimedElement : List<string>

options : List<string>

_commandFactory : CommandFactory

+ Process(string cmd): void

unchIndividualCommand

+ Arguments : CommandArguments

+CommandName: string + Execute() : void

+ Processor(CommandController commandController)

- ParseArguments(string cmd): CommandArguments

+ CommandName(): string

+ Options: List<string>

+ AimedElement : List<string>

+ LaunchIndividualCommand(CommandController commandController)

- commandController : CommandController

+ Arguments : CommandArguments + ExitCommand(CommandController commandController)

ommandFactory

- _availableCommands : IEnumerable<ICommand>

+ CommandFactory(CommandController commandController)

+ CreateCommand(CommandArguments commandArguments): ICommand

- commandController : CommandController

+CommandName : string - commandController : CommandController + Execute() : void

NotFoundCommand

commandController : CommandController

+ Arguments : CommandArguments

+ NotFoundCommand(CommandController commandController)

+CommandName : string

+ Execute() : void

CommandName : string Arguments : CommandArguments

Execute() : void

HelpCommand

- commandController : CommandController + Arguments : CommandArguments

+ HelpCommand(CommandController commandController)

+CommandName : string + Execute() : void

+ Arguments : CommandArguments + LaunchGroupedCommand(CommandController commandController) +CommandName : string

commandController : CommandController

nchGroupedCommand

+ Execute() : void