Namespace EasySave.ViewModels

Classes

[BaseViewModel](#)

 Classe abstraite BaseViewModel

[JobViewModel](#)

 Classe JobViewModel

[LangueViewModel](#)

[MainViewModel](#)

Class BaseViewModel

Namespace: [EasySave](#).[ViewModels](#)

Assembly: EasySave.dll

Classe abstraite BaseViewModel

```
public abstract class BaseViewModel : INotifyPropertyChanged
```

Inheritance

[object](#) ← BaseViewModel

Implements

[INotifyPropertyChanged](#)

Derived

[JobViewModel](#), [LangueViewModel](#)

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Methods

NotifyPropertyChanged(string)

Méthode à appler pour avertir d'une modification

```
protected void NotifyPropertyChanged(string propertyName = "")
```

Parameters

`propertyName` [string](#)

Nom de la property modifiée (authomatiquement déterminé si appelé directement dans le setter une property)

Events

PropertyChanged

Evénement de modification d'une property

```
public event PropertyChangedEventHandler PropertyChanged
```

Event Type

[PropertyChangedEventHandler⤢](#)

Class JobViewModel

Namespace: [EasySave](#).[ViewModels](#)

Assembly: EasySave.dll

Classe JobViewModel

```
public class JobViewModel : BaseViewModel, INotifyPropertyChanged
```

Inheritance

[object](#) ← [BaseViewModel](#) ← JobViewModel

Implements

[INotifyPropertyChanged](#)

Inherited Members

[BaseViewModel.PropertyChanged](#) , [BaseViewModel.NotifyPropertyChanged(string)](#) ,
[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Constructors

JobViewModel()

Constructeur de JobViewModel initialise le JobManager

```
public JobViewModel()
```

Properties

JobManager

JobManager

```
public CJobManager JobManager { get; set; }
```

Property Value

CJobManager

Methods

CreateBackupJob(CJob)

Crée un nouveau job de sauvegarde

```
public bool CreateBackupJob(CJob lJob)
```

Parameters

`lJob` CJob

   Job à créer

Returns

bool⤢

   Succès de la création

DeleteJobs(List<CJob>)

Supprimer un ou plusieurs jobs

```
public bool DeleteJobs(List<CJob> pJobs)
```

Parameters

`pJobs` List⤢<CJob>

   List de jobs a delete

Returns

bool⤢

vrai si les jobs on été delete

LoadJobs(bool, string)

Charge la liste des jobs depuis un fichier

```
public void LoadJobs(bool IsDefaultFile = true, string pPath = null)
```

Parameters

IsDefaultFile [bool](#)⧉

   Indique si le fichier par défaut doit être chargé

pPath [string](#)⧉

   Chemin du fichier à charger, vide pour le fichier par défaut

RunJobs(List<CJob>)

Lance l'exécution des jobs sélectionnés

```
public List<CJob> RunJobs(List<CJob> pJobs)
```

Parameters

pJobs [List](#)⧉<[CJob](#)>

   Liste des jobs à lancer

Returns

[List](#)⧉<[CJob](#)>

   Liste mise à jour des jobs avec leur état après exécution

SaveJobs()

Sauvegarde la configuration des jobs

```
public void SaveJobs()
```

Class LangueViewModel

Namespace: [EasySave](#).[ViewModels](#)

Assembly: EasySave.dll

```
public class LangueViewModel : BaseViewModel, INotifyPropertyChanged
```

Inheritance

[object](#) ← [BaseViewModel](#) ← LangueViewModel

Implements

[INotifyPropertyChanged](#)

Inherited Members

[BaseViewModel.PropertyChanged](#) , [BaseViewModel.NotifyPropertyChanged(string)](#) ,
[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Constructors

LangueViewModel()

```
public LangueViewModel()
```

Properties

Langue

```
public CLangue Langue { get; set; }
```

Property Value

[CLangue](#)

Methods

SetLanguage(string)

Set the current language

```csharp
public bool SetLanguage(string pCultureInfo)
```

Parameters

`pCultureInfo` [string](#)

Returns

[bool](#)

    true if the language was changed

Class MainViewModel

Namespace: EasySave.ViewModels

Assembly: EasySave.dll

```
public class MainViewModel
```

Inheritance

object ← MainViewModel

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

MainViewModel()

```
public MainViewModel()
```

Properties

JobVm

```
public JobViewModel JobVm { get; set; }
```

Property Value

JobViewModel

LangueVm

```
public LangueViewModel LangueVm { get; set; }
```

Property Value

LangueViewModel

Namespace EasySave.Views

Classes

[BaseView](#)

Vue de l'application

[ConsoleExtention](#)

[LangueView](#)

[View](#)

Class BaseView

Namespace: [EasySave](#).[Views](#)

Assembly: EasySave.dll

Vue de l'application

```
public abstract class BaseView
```

Inheritance

[object](#) ← BaseView

Derived

[LangueView](#), [View](#)

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Properties

Title

```
public abstract string Title { get; }
```

Property Value

[string](#)

Methods

Run()

Lance le deroulement de la vue dans l'interface de maniere procedurale

```
public abstract void Run()
```

Class ConsoleExtention

Namespace: [EasySave](#).[Views](#)

Assembly: EasySave.dll

```
public static class ConsoleExtention
```

Inheritance

[object](#) ← ConsoleExtention

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Methods

Clear()

Clear the console and set the input to -1

```
public static void Clear()
```

ReadFile(string, Regex, string)

Read a file with GTK CrossPlatform interface if it fail open classic Console Interface

```
public static string ReadFile(string pDescription, Regex pRegexExtentions = null, string
pCurrentFolder = null)
```

Parameters

`pDescription` [string](#)

   Description for the interface

`pRegexExtentions` [Regex](#)

`pCurrentFolder` [string↗](#)

Returns

[string↗](#)

   return the selected file full path

ReadFolder(string)

Read a folder with GTK CrossPlatform interface if it fail open classic Console Interface

```
public static string ReadFolder(string pDescription)
```

Parameters

`pDescription` [string↗](#)

   Description for the interface

Returns

[string↗](#)

   return the selected folder full path

ReadResponse(string, Regex?, Func<string, bool>)

Read user input char by char

```
public static string ReadResponse(string pMessage, Regex? pRegex = null, Func<string, bool>
pIsValid = null)
```

Parameters

`pMessage` [string↗](#)

   Message to loop through if the user makes an input error

pRegex [Regex⧉](#)

Regex permettant de validée l'entrée utilisateur

pIsValid [Func⧉](#) < [string⧉](#), [bool⧉](#) >

Fonction qui prend un string en parametre et valide l'entrée utilisateur

Returns

[string⧉](#)

user input

Remarks

Mahmoud Charif - 05/02/2024 - Création

WriteLineError(string)

Write line a error in red

```
public static void WriteLineError(string pMessage)
```

Parameters

pMessage [string⧉](#)

message to write

WriteLineSelected(string)

Write a default message + input

```
public static void WriteLineSelected(string pInput)
```

Parameters

pInput [string⧉](#)

WriteLineSucces(string)

Write line a succes in green

```
public static void WriteLineSucces(string pMessage)
```

Parameters

pMessage string

　message to write

WriteLineWarning(string)

WriteLine the message Warning in DarkYellow

```
public static void WriteLineWarning(string pMessage)
```

Parameters

pMessage string

　message to write

WritePath(string)

Write Path with UNC Format in yellow

```
public static void WritePath(string pPath)
```

Parameters

pPath string

　path to write

WriteSubtitle(string, ConsoleColor)

WriteSubTitle

```
public static void WriteSubtitle(string pSubtitle, ConsoleColor pColor
= ConsoleColor.DarkGray)
```

Parameters

pSubtitle string↗

 subvtitle

pColor ConsoleColor↗

 couleur du subtitle

WriteTitle(string, ConsoleColor)

Write a personalized Title with separator

```
public static void WriteTitle(string pTitle, ConsoleColor pColor = ConsoleColor.White)
```

Parameters

pTitle string↗

 Title to write

pColor ConsoleColor↗

Class LangueView

Namespace: <u>EasySave</u>.<u>Views</u>

Assembly: EasySave.dll

```csharp
public class LangueView : BaseView
```

Inheritance

<u>object</u> ← <u>BaseView</u> ← LangueView

Inherited Members

<u>object.Equals(object)</u> , <u>object.Equals(object, object)</u> , <u>object.GetHashCode()</u> , <u>object.GetType()</u> , <u>object.MemberwiseClone()</u> , <u>object.ReferenceEquals(object, object)</u> , <u>object.ToString()</u>

Constructors

LangueView(LangueViewModel)

```csharp
public LangueView(LangueViewModel pJobVm)
```

Parameters

`pJobVm` <u>LangueViewModel</u>

Properties

Title

```csharp
public override string Title { get; }
```

Property Value

<u>string</u>

Methods

ListLanguage()

Liste les langue disponibles

```
public void ListLanguage()
```

Run()

Lance la selection du language

```
public override void Run()
```

Class View

Namespace: [EasySave](#).[Views](#)

Assembly: EasySave.dll

```
public class View : BaseView
```

Inheritance

[object](#) ← [BaseView](#) ← View

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Constructors

View()

```
public View()
```

Properties

Menu

Chaîne de caractères contenant le menu

```
public string Menu { get; }
```

Property Value

[string](#)

Title

Titre affiché pour l'application

```csharp
public override string Title { get; }
```

Property Value

[string](#)⤢

Methods

Run()

Start the main program

```csharp
public override void Run()
```

Namespace LogsModels

Classes

CLogBase

CLogDaily

CLogState

Interfaces

IPath

Class CLogBase

Namespace: LogsModels

Assembly: LogsModels.dll

```
[DataContract]
public abstract class CLogBase : IPath
```

Inheritance

object ← CLogBase

Implements

IPath

Derived

CLogDaily, CLogState

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Date

```
public virtual DateTime Date { get; set; }
```

Property Value

DateTime

IsSummary

```
public virtual bool IsSummary { get; set; }
```

Property Value

[bool](#)↗

Name

```
public virtual string Name { get; set; }
```

Property Value

[string](#)↗

SourceDirectory

```
public virtual string SourceDirectory { get; set; }
```

Property Value

[string](#)↗

TargetDirectory

```
public virtual string TargetDirectory { get; set; }
```

Property Value

[string](#)↗

TotalSize

```
public virtual double TotalSize { get; set; }
```

Property Value

[double](#)↗

Class CLogDaily

Namespace: [LogsModels](#)

Assembly: LogsModels.dll

```
public class CLogDaily : CLogBase, IPath
```

Inheritance

[object](#) ← [CLogBase](#) ← CLogDaily

Implements

[IPath](#)

Inherited Members

[CLogBase.Name](#) , [CLogBase.Date](#) , [CLogBase.TotalSize](#) , [CLogBase.SourceDirectory](#) ,
[CLogBase.TargetDirectory](#) , [CLogBase.IsSummary](#) , [object.Equals(object)](#) ,
[object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Properties

TransfertTimeSecond

```
public double TransfertTimeSecond { get; set; }
```

Property Value

[double](#)

Class CLogState

Namespace: [LogsModels](#)

Assembly: LogsModels.dll

```
[DataContract]
public class CLogState : CLogBase, IPath
```

Inheritance

[object](#) ← [CLogBase](#) ← CLogState

Implements

[IPath](#)

Inherited Members

[CLogBase.Date](#) , [CLogBase.TotalSize](#) , [CLogBase.SourceDirectory](#) , [CLogBase.TargetDirectory](#) ,
[CLogBase.IsSummary](#) , [object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) ,
[object.GetType()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) ,
[object.ToString()](#)

Constructors

CLogState()

```
public CLogState()
```

Properties

ElapsedMilisecond

```
public long ElapsedMilisecond { get; set; }
```

Property Value

[long](#)

EligibleFileCount

```
public int EligibleFileCount { get; set; }
```

Property Value

[int↗](#)

IsActive

```
public bool IsActive { get; set; }
```

Property Value

[bool↗](#)

Name

```
public override string Name { get; set; }
```

Property Value

[string↗](#)

RemainingFiles

```
public int RemainingFiles { get; set; }
```

Property Value

[int↗](#)

Interface IPath

Namespace: LogsModels

Assembly: LogsModels.dll

```
public interface IPath
```

Properties

SourceDirectory

```
string SourceDirectory { get; set; }
```

Property Value

string⤢

TargetDirectory

```
string TargetDirectory { get; set; }
```

Property Value

string⤢

Namespace Models

Classes

[CLangue](#)

Classe de la langue de l'application

[Settings](#)

Classe des settings de l'application permettant le chargement et la sauvegarde des parametres de l'utilisateur

Class CLangue

Namespace: Models

Assembly: Models.dll

Classe de la langue de l'application

```
[DataContract]
public class CLangue
```

Inheritance

object ← CLangue

Inherited Members

object.Equals(object) , [object.Equals(object, object)](object.Equals(object,object)) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , [object.ReferenceEquals(object, object)](object.ReferenceEquals(object,object)) , object.ToString()

Constructors

CLangue()

Constructeur de la classe Clangue Init the language with the installed culture of the operating system

```
public CLangue()
```

Properties

Languages

Dictionnaire de langues disponnible dans l'application

```
public Dictionary<int, string> Languages { get; set; }
```

Property Value

Dictionary<[int](int), string>

SelectedCulture

```
public string SelectedCulture { get; set; }
```

Property Value

[string](link)

Methods

SetLanguage(string)

Set the current UI culture

```
public bool SetLanguage(string pCultureInfo)
```

Parameters

pCultureInfo [string](link)

Returns

[bool](link)

true if the language was changed

Class Settings

Namespace: Models

Assembly: Models.dll

Classe des settings de l'application permettant le chargement et la sauvegarde des parametres de l'utilisateur

```
[DataContract]
public class Settings
```

Inheritance

object ← Settings

Inherited Members

object.Equals(object) , [object.Equals(object, object)](object.Equals(object,_object)) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , [object.ReferenceEquals(object, object)](object.ReferenceEquals(object,_object)) , object.ToString()

Properties

Instance

```
public static Settings Instance { get; }
```

Property Value

Settings

JobConfigFolderPath

Emplacement du répertoire dans lequel le fichier de configuration du travail est stocké

```
public string JobConfigFolderPath { get; set; }
```

Property Value

[string↗](#)

JobDefaultConfigPath

Emplacement par défaut du répertoire dans lequel le fichier de configuration du travail est stocké

```csharp
public string JobDefaultConfigPath { get; set; }
```

Property Value

[string↗](#)

Langue

Langue préférer de l'utilisateur

```csharp
public CLangue Langue { get; set; }
```

Property Value

[CLangue](#)

Methods

~Settings()

```csharp
protected ~Settings()
```

LoadJobsFile(string)

Charge la liste des jobs depuis un fichier

```csharp
public CJobManager LoadJobsFile(string pPath = null)
```

Parameters

pPath [string↗](#)

Chemin du fichier de configuration. Null pour le fichier par défaut.

Returns

[CJobManager](#)

Instance du gestionnaire de jobs chargé

LoadSettings()

Chargement des paramètres à partir d'un fichier json

```
public void LoadSettings()
```

SaveSettings()

Enregistrer les paramètres dans un fichier json

```
public void SaveSettings()
```

Namespace Models.Backup

Classes

CJob

Représente un travail/tâche à exécuter

CJobManager

Gestionnaire de jobs

Enums

ETypeBackup

Enumeration du type de backup

Class CJob

Namespace: [Models](#).[Backup](#)

Assembly: Models.dll

Représente un travail/tâche à exécuter

```
[DataContract]
public class CJob : IPath
```

Inheritance

[object](#) ← CJob

Implements

[IPath](#)

Inherited Members

[object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Constructors

CJob(string, string, string, ETypeBackup)

Constructeur de job

```
public CJob(string pName, string pSourceDirectory, string pTargetDirectory,
ETypeBackup pTypeBackup)
```

Parameters

`pName` [string](#)

Nom du job

`pSourceDirectory` [string](#)

Chemin source

`pTargetDirectory` [string↗](#)

    Chemin destination

`pTypeBackup` [ETypeBackup](#)

    Type de sauvegarde

Remarks

Mahmoud Charif - 30/01/2024 - Création

Properties

BackupType

Type de sauvegarde

```csharp
public ETypeBackup BackupType { get; set; }
```

Property Value

[ETypeBackup](#)

Name

Nom du job de sauvegarde

```csharp
public string Name { get; set; }
```

Property Value

[string↗](#)

SourceDirectory

Répertoire source à sauvegarder

```
public string SourceDirectory { get; set; }
```

Property Value

[string](#)

TargetDirectory

Répertoire cible de la sauvegarde

```
public string TargetDirectory { get; set; }
```

Property Value

[string](#)

Methods

Equals(object?)

Determines whether the specified object is equal to the current object.

```
public override bool Equals(object? obj)
```

Parameters

obj [object](#)

The object to compare with the current object.

Returns

[bool](#)

[true](#) if the specified object is equal to the current object; otherwise, [false](#).

Run(SauveJobs)

Lance l'exécution du job de sauvegarde

```
public void Run(SauveJobs pSauveJobs)
```

Parameters

pSauveJobs SauveJobs

Objet de sauvegarde des données de jobs

Class CJobManager

Namespace: Models.Backup

Assembly: Models.dll

Gestionnaire de jobs

```
[DataContract]
public class CJobManager
```

Inheritance

object ← CJobManager

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

CJobManager()

Contructeur de CJobManager initialise le chemin de sauvegarde

```
public CJobManager()
```

Properties

Jobs

Liste des jobs gérés

```
public List<CJob> Jobs { get; }
```

Property Value

List<CJob>

Name

Nom du gestionnaire

```csharp
public string Name { get; set; }
```

Property Value

[string↗]

SauveCollection

Interface de sauvegarde des données

```csharp
public ISauve SauveCollection { get; set; }
```

Property Value

[ISauve]

Methods

CreateBackupJob(CJob)

Crée un nouveau job de sauvegarde

```csharp
public bool CreateBackupJob(CJob lJob)
```

Parameters

`lJob` [CJob]

Returns

[bool↗]

True si le job a été créé avec succès, false sinon

Remarks

Created by Mehmeti Faik on 06/02/2024 Updated validation logic to handle null parameters

DeleteJobs(List<CJob>)

Supprimé un job

```
public bool DeleteJobs(List<CJob> pJobs)
```

Parameters

pJobs List☒ <CJob>

   List de jobs à supprimer

Returns

bool☒

   true si reussi

Remarks

Mehmeti faik

RunJobs(List<CJob>)

Lance l'exécution de la liste de jobs passée en paramètre

```
public List<CJob> RunJobs(List<CJob> pJobs)
```

Parameters

pJobs List☒ <CJob>

   Liste des jobs à exécuter

Returns

[List](#)↗ <[CJob](#)>

La liste des jobs, mise à jour avec leur état après exécution

SaveJobs()

Sauvegarde le JobManager

```
public void SaveJobs()
```

Enum ETypeBackup

Namespace: [Models](#).[Backup](#)

Assembly: Models.dll

Enumeration du type de backup

```
public enum ETypeBackup
```

Fields

COMPLET = 0

DIFFERENTIEL = 1

Namespace Stockage.Converters

Classes

[ConcreteCollectionTypeConverter<TCollection, TItem, TBaseItem>](#)
    Concrete Collection Converter

[ConcreteConverter<TInterface, TConcrete>](#)
    This convert can be used on any interface definition to instruct the JSON serializer to use a specific
    concrete class when deserializing the instance. The type specified by TConcrete must implement the
    interface specified by TInterface.

[ConcreteDictionnaryTypeConverter<TDictionary, TItem, TKey, TValue>](#)
    Concrete dictionnaryConverter

Class ConcreteCollectionTypeConverter<TCollection, TItem, TBaseItem>

Namespace: [Stockage](.).[Converters](.)

Assembly: Stockage.dll

Concrete Collection Converter

```csharp
public class ConcreteCollectionTypeConverter<TCollection, TItem, TBaseItem> : JsonConverter
where TCollection : ICollection<TBaseItem>, new() where TItem : TBaseItem
```

Type Parameters

`TCollection`

    Collection

`TItem`

    Item de la collection

`TBaseItem`

    Item de base

Inheritance

[object](.) ← JsonConverter ← ConcreteCollectionTypeConverter<TCollection, TItem, TBaseItem>

Inherited Members

JsonConverter.CanRead , JsonConverter.CanWrite , [object.Equals(object)](.) ,
[object.Equals(object, object)](.) , [object.GetHashCode()](.) , [object.GetType()](.) ,
[object.MemberwiseClone()](.) , [object.ReferenceEquals(object, object)](.) , [object.ToString()](.)

Remarks

Mahmoud Charif - 31/12/2022 - Creation

Methods

CanConvert(Type)

Can convert

```
public override bool CanConvert(Type objectType)
```

Parameters

objectType Type⊠

Returns

bool⊠

ReadJson(JsonReader, Type, object, JsonSerializer)

ReadJson

```
public override object ReadJson(JsonReader reader, Type objectType, object existingValue,
JsonSerializer serializer)
```

Parameters

reader JsonReader

objectType Type⊠

existingValue object⊠

serializer JsonSerializer

Returns

object⊠

WriteJson(JsonWriter, object, JsonSerializer)

Writes the JSON representation of the object.

```csharp
public override void WriteJson(JsonWriter writer, object value, JsonSerializer serializer)
```

Parameters

`writer` JsonWriter

    The Newtonsoft.Json.JsonWriter to write to.

`value` [object↗]

    The value.

`serializer` JsonSerializer

    The calling serializer.

Class ConcreteConverter<TInterface, TConcrete>

Namespace: [Stockage](#).[Converters](#)

Assembly: Stockage.dll

This convert can be used on any interface definition to instruct the JSON serializer to use a specific concrete class when deserializing the instance. The type specified by TConcrete must implement the interface specified by TInterface.

```
public class ConcreteConverter<TInterface, TConcrete> : JsonConverter where TConcrete :
TInterface, new()
```

Type Parameters

`TInterface`

The Type that was serialized into the JSON text.

`TConcrete`

The Type that specifies the class that will be created.

Inheritance

[object](#) ← JsonConverter ← ConcreteConverter<TInterface, TConcrete>

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Properties

CanRead

Gets a value indicating whether this Newtonsoft.Json.JsonConverter can read.

```
public override bool CanRead { get; }
```

Property Value

bool↗

CanWrite

Gets a value indicating whether this Newtonsoft.Json.JsonConverter can write JSON.

```
public override bool CanWrite { get; }
```

Property Value

bool↗

Methods

CanConvert(Type)

Determines whether this instance can convert the specified object type.

```
public override bool CanConvert(Type objectType)
```

Parameters

objectType Type↗

  Type of the object.

Returns

bool↗

  Returns true if this instance can convert the specified object type, false otherwise.

ReadJson(JsonReader, Type, object?, JsonSerializer)

Reads the JSON representation of the object.

```
public override object ReadJson(JsonReader reader, Type objectType, object? existingValue,
JsonSerializer serializer)
```

Parameters

`reader` JsonReader

The Newtonsoft.Json.JsonReader to read from.

`objectType` [Type⧉](#)

Type of the object.

`existingValue` [object⧉](#)

The existing value of object being read.

`serializer` JsonSerializer

The calling serializer.

Returns

[object⧉](#)

The object value.

WriteJson(JsonWriter, object?, JsonSerializer)

Writes the JSON representation of the object.

```
public override void WriteJson(JsonWriter writer, object? value, JsonSerializer serializer)
```

Parameters

`writer` JsonWriter

The Newtonsoft.Json.JsonWriter to write to.

`value` [object⧉](#)

The value.

`serializer` JsonSerializer

    The calling serializer.

Class ConcreteDictionnaryTypeConverter<TDictionary, TItem, TKey, TValue>

Namespace: Stockage.Converters

Assembly: Stockage.dll

Concrete dictionnaryConverter

```
public class ConcreteDictionnaryTypeConverter<TDictionary, TItem, TKey, TValue> :
JsonConverter where TDictionary : IDictionary<TKey, TValue>, new() where TItem : TValue
```

Type Parameters

TDictionary

TItem

TKey

TValue

Inheritance

object ← JsonConverter ← ConcreteDictionnaryTypeConverter<TDictionary, TItem, TKey, TValue>

Inherited Members

JsonConverter.CanRead , JsonConverter.CanWrite , object.Equals(object) ,
[object.Equals(object, object)](object.Equals(object,object)) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , [object.ReferenceEquals(object, object)](object.ReferenceEquals(object,object)) , object.ToString()

Methods

CanConvert(Type)

CanConvert

```
public override bool CanConvert(Type objectType)
```

Parameters

`objectType` [Type⧉](#)

Returns

[bool⧉](#)

ReadJson(JsonReader, Type, object?, JsonSerializer)

ReadJson

```
public override object ReadJson(JsonReader reader, Type objectType, object? existingValue,
JsonSerializer serializer)
```

Parameters

`reader` JsonReader

`objectType` [Type⧉](#)

`existingValue` [object⧉](#)

`serializer` JsonSerializer

Returns

[object⧉](#)

WriteJson(JsonWriter, object?, JsonSerializer)

WriteJson

```
public override void WriteJson(JsonWriter writer, object? value, JsonSerializer serializer)
```

Parameters

`writer` JsonWriter

`value` [object]↗

`serializer` JsonSerializer

Namespace Stockage.Load

Classes

[BaseCharge](#)

Classe abstraite de base pour le chargement d'un object

[ChargerCollection](#)

Interfaces

[ICharge](#)

Interface ICharge

Class BaseCharge

Namespace: [Stockage](#).[Load](#)

Assembly: Stockage.dll

Classe abstraite de base pour le chargement d'un object

```
public abstract class BaseCharge : ICharge
```

Inheritance

[object](#) ← BaseCharge

Implements

[ICharge](#)

Derived

[ChargerCollection](#)

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Constructors

BaseCharge(string)

Constructeur

```
public BaseCharge(string pPath)
```

Parameters

`pPath` [string](#)

Chemin du dossier

Remarks

Mahmoud Charif - 13/02/2024 - Création

Methods

Charger<T>(string, bool)

Charger un fichier

```
public virtual T Charger<T>(string pFileName, bool pIsFullPath = false)
```

Parameters

`pFileName` [string](#)

    Nom du fichier

`pIsFullPath` [bool](#)

    vrai si le premier parametre est un chemin complet et non le nom du fichier

Returns

`T`

    Data Cast in Generic Type

Type Parameters

`T`

    Type du fichier à charger

Remarks

Mahmoud Charif - 31/12/2022 - Creation

Class ChargerCollection

Namespace: [Stockage](#).[Load](#)

Assembly: Stockage.dll

```
public class ChargerCollection : BaseCharge, ICharge
```

Inheritance

[object](#) ← [BaseCharge](#) ← ChargerCollection

Implements

[ICharge](#)

Inherited Members

[BaseCharge.Charger<T>(string, bool)](#) , [object.Equals(object)](#) , [object.Equals(object, object)](#) ,
[object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) ,
[object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Constructors

ChargerCollection(string)

```
public ChargerCollection(string pPath)
```

Parameters

`pPath` [string](#)

Interface ICharge

Namespace: [Stockage](#).[Load](#)

Assembly: Stockage.dll

Interface ICharge

```
public interface ICharge
```

Remarks

Mahmoud Charif - 31/12/2022- Création

Methods

Charger<T>(string, bool)

Charger un fichier

```
T Charger<T>(string pPath, bool pIsFullPath = false)
```

Parameters

`pPath` [string](#)

`pIsFullPath` [bool](#)

vrai si le premier parametre est un chemin complet et non le nom du fichier

Returns

`T`

Data Cast in Generic Type

Type Parameters

`T`

Type du fichier à charger

Remarks

Mahmoud Charif - 31/12/2022 - Creation

Namespace Stockage.Logs

Classes

[BaseLogger<T>](#)

    Classe de base abstraite pour les loggers.

[CGenericLogger<T>](#)

    Classe de logger générique

[CLogger<T>](#)

    Classe Logger permettant de Logger des objet et des string dans un fichier

[CStringLogger](#)

    Logger spécialisé pour les chaines de caractères

Interfaces

[ILogger<T>](#)

Class BaseLogger<T>

Namespace: [Stockage](#).[Logs](#)

Assembly: Stockage.dll

Classe de base abstraite pour les loggers.

```
public abstract class BaseLogger<T> : ILogger<T>
```

Type Parameters

`T`

    Type des objets loggés

Inheritance

[object](#)⬈ ← BaseLogger<T>

Implements

[ILogger](#)<T>

Derived

[CGenericLogger<T>](#), [CStringLogger](#)

Inherited Members

[object.Equals(object)](#)⬈ , [object.Equals(object, object)](#)⬈ , [object.GetHashCode()](#)⬈ , [object.GetType()](#)⬈ ,
[object.MemberwiseClone()](#)⬈ , [object.ReferenceEquals(object, object)](#)⬈ , [object.ToString()](#)⬈

Constructors

BaseLogger()

```
protected BaseLogger()
```

Properties

Datas

Collection de données observables

```
public ObservableCollection<T> Datas { get; }
```

Property Value

[ObservableCollection](link)⧉ <T>

Methods

Clear()

Vide la collection de données

```
public virtual void Clear()
```

Log(T, bool, bool, string)

Méthode de logging des données

```
public virtual void Log(T pData, bool pSerialize = true, bool pAppend = true, string
pFileName = "Logs")
```

Parameters

pData T

   Données à logger

pSerialize [bool](link)⧉

   Indique si les données doivent être sérialisées avant d'être loggées

pAppend [bool](link)⧉

   Indique si on ajoute le logging au fichier existant ou si on recrée le fichier

pFileName [string](link)⧉

Nom du fichier où sont loggées les données

Class CGenericLogger<T>

Namespace:

Assembly: Stockage.dll

Classe de logger générique

```
public class CGenericLogger<T> : BaseLogger<T>, ILogger<T>
```

Type Parameters

T

   Type des objets loggés

Inheritance

[object](.) ← [BaseLogger](.)<T> ← CGenericLogger<T>

Implements

[ILogger](.)<T>

Inherited Members

[BaseLogger<T>.Datas](.) , [BaseLogger<T>.Log(T, bool, bool, string)](.) , [BaseLogger<T>.Clear()](.) ,
[object.Equals(object)](.) , [object.Equals(object, object)](.) , [object.GetHashCode()](.) , [object.GetType()](.) ,
[object.MemberwiseClone()](.) , [object.ReferenceEquals(object, object)](.) , [object.ToString()](.)

Class CLogger<T>

Namespace: [Stockage](#).[Logs](#)

Assembly: Stockage.dll

Classe Logger permettant de Logger des objet et des string dans un fichier

```
public static class CLogger<T>
```

Type Parameters

T

Inheritance

[object](#) ← CLogger<T>

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Properties

GenericLogger

Logger generic

```
public static CGenericLogger<T> GenericLogger { get; }
```

Property Value

[CGenericLogger](#)<T>

StringLogger

Logger de string

```csharp
public static CStringLogger StringLogger { get; }
```

Property Value

CStringLogger

Methods

Clear()

Vide les Liste de logs

```csharp
public static void Clear()
```

Class CStringLogger

Namespace: [Stockage](#).[Logs](#)

Assembly: Stockage.dll

Logger spécialisé pour les chaines de caractères

```
public class CStringLogger : BaseLogger<string>, ILogger<string>
```

Inheritance

[object](#) ← [BaseLogger](#)<[string](#)> ← CStringLogger

Implements

[ILogger](#)<[string](#)>

Inherited Members

[BaseLogger<string>.Datas](#) , [BaseLogger<string>.Log(string, bool, bool, string)](#) ,
[BaseLogger<string>.Clear()](#) , [object.Equals(object)](#) , [object.Equals(object, object)](#) ,
[object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) ,
[object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Interface ILogger\<T>

Namespace: [Stockage](#).[Logs](#)

Assembly: Stockage.dll

```
public interface ILogger<T>
```

Type Parameters

T

Properties

Datas

```
ObservableCollection<T> Datas { get; }
```

Property Value

[ObservableCollection](#)⬀ \<T>

Methods

Log(T, bool, bool, string)

```
void Log(T pData, bool pSerialize, bool pAppend = true, string pFileName = "Logs")
```

Parameters

pData T

pSerialize [bool](#)⬀

pAppend [bool](#)⬀

pFileName [string](#)⬀

Namespace Stockage.Save

Classes

[BaseSave](#)

Classe abstraite de base pour la sauvegrade d'un ficher ou le deplacement de Repertoire

[SauveCollection](#)

[SauveJobs](#)

Classe permettant de sauvegarder des jobs et de les logger

Interfaces

[ISauve](#)

Interface ISauve

Class BaseSave

Namespace: [Stockage](#).[Save](#)

Assembly: Stockage.dll

Classe abstraite de base pour la sauvegrade d'un ficher ou le deplacement de Repertoire

```
public abstract class BaseSave : ISauve
```

Inheritance

[object](#) ← BaseSave

Implements

[ISauve](#)

Derived

[SauveCollection](#), [SauveJobs](#)

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Constructors

BaseSave(string)

Sauvgarde

```
public BaseSave(string pPath)
```

Parameters

`pPath` [string](#)

Directory Path

Properties

Options

```
public JsonSerializerSettings Options { get; }
```

Property Value

JsonSerializerSettings

Methods

CopyDirectory(DirectoryInfo, DirectoryInfo, bool, ref CLogState, bool)

```
public virtual void CopyDirectory(DirectoryInfo pSourceDir, DirectoryInfo pTargetDir, bool
pRecursive, ref CLogState pLogState, bool pForce = false)
```

Parameters

pSourceDir [DirectoryInfo ↗](#)

pTargetDir [DirectoryInfo ↗](#)

pRecursive [bool ↗](#)

pLogState [CLogState](#)

pForce [bool ↗](#)

CopyDirectory(DirectoryInfo, DirectoryInfo, bool, bool)

Copy files and directory from the soruce path to the destinationPath

```
public virtual void CopyDirectory(DirectoryInfo pSourceDir, DirectoryInfo pTargetDir, bool
pRecursive, bool pForce = false)
```

Parameters

pSourceDir [DirectoryInfo ↗](#)

Path of the directory you want tot copy

pTargetDir [DirectoryInfo⧉](#)

Path of the target directory

pRecursive [bool⧉](#)

True if recursive

pForce [bool⧉](#)

true if overwrite

Exceptions

[DirectoryNotFoundException⧉](#)

Sauver<T>(T, string, bool, string, bool)

Crée un fichier Json par default avec les Settings

```
public virtual void Sauver<T>(T pData, string pFileName, bool pAppend = false, string
pExtention = "json", bool IsFullPath = false)
```

Parameters

pData T

Data a sauvgarde

pFileName [string⧉](#)

Name of the file

pAppend [bool⧉](#)

pExtention [string⧉](#)

Extention of the file can be null

IsFullPath [bool⧉](#)

Type Parameters

T

Interface ISauve

Namespace: [Stockage](#).[Save](#)

Assembly: Stockage.dll

Interface ISauve

```
public interface ISauve
```

Remarks

Mahmoud Charif - 31/12/2022 - Création

Methods

CopyDirectory(DirectoryInfo, DirectoryInfo, bool, bool)

Copy files and directory from the soruce path to the destinationPath

```
void CopyDirectory(DirectoryInfo pSourceDir, DirectoryInfo pTargetDir, bool pRecursive, bool
pForce = false)
```

Parameters

`pSourceDir` [DirectoryInfo](#)↗

　Path of the directory you want tot copy

`pTargetDir` [DirectoryInfo](#)↗

　Path of the target directory

`pRecursive` [bool](#)↗

　True if recursive

`pForce` [bool](#)↗

　true if overwrite

Exceptions

[DirectoryNotFoundException↗](#)

Sauver<T>(T, string, bool, string, bool)

Sauvagarde les données dans un fichier

```
void Sauver<T>(T pData, string pFileName, bool pAppend = false, string pExtention = "json",
bool IsFullPath = false)
```

Parameters

`pData` T

Data to serialize

`pFileName` [string↗](#)

File name

`pAppend` [bool↗](#)

True si on veux append sur le fichier

`pExtention` [string↗](#)

Extention

`IsFullPath` [bool↗](#)

vrai si pFileName est un chemin complet

Type Parameters

`T`

Remarks

Mahmoud Charif - 31/12/2022 - Création

Class SauveCollection

Namespace: [Stockage](#).[Save](#)

Assembly: Stockage.dll

```csharp
public class SauveCollection : BaseSave, ISauve
```

Inheritance

[object](#) ← [BaseSave](#) ← SauveCollection

Implements

[ISauve](#)

Inherited Members

[BaseSave.Options](#) , [BaseSave.Sauver<T>(T, string, bool, string, bool)](#) ,
[BaseSave.CopyDirectory(DirectoryInfo, DirectoryInfo, bool, bool)](#) ,
[BaseSave.CopyDirectory(DirectoryInfo, DirectoryInfo, bool, ref CLogState, bool)](#) , [object.Equals(object)](#) ,
[object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Constructors

SauveCollection(string)

```csharp
public SauveCollection(string pPath)
```

Parameters

`pPath` [string](#)

Class SauveJobs

Namespace: [Stockage](#).[Save](#)

Assembly: Stockage.dll

Classe permettant de sauvegarder des jobs et de les logger

```
public class SauveJobs : BaseSave, ISauve
```

Inheritance

[object](#) ← [BaseSave](#) ← SauveJobs

Implements

[ISauve](#)

Inherited Members

[BaseSave.Options](#) , [BaseSave.Sauver<T>(T, string, bool, string, bool)](#) ,
[BaseSave.CopyDirectory(DirectoryInfo, DirectoryInfo, bool, bool)](#) , [object.Equals(object)](#) ,
[object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

Constructors

SauveJobs(string)

```
public SauveJobs(string pPath = null)
```

Parameters

`pPath` [string](#)

Properties

TransferedFiles

```
public int TransferedFiles { get; set; }
```

Property Value

[int↗]

Methods

CopyDirectory(DirectoryInfo, DirectoryInfo, bool, ref CLogState, bool)

Copy files and directory from the soruce path to the destinationPath

```
public override void CopyDirectory(DirectoryInfo pSourceDir, DirectoryInfo pTargetDir, bool
pRecursive, ref CLogState pLogState, bool pForce = false)
```

Parameters

pSourceDir [DirectoryInfo↗]

    Path of the directory you want tot copy

pTargetDir [DirectoryInfo↗]

    Path of the target directory

pRecursive [bool↗]

    True if recursive

pLogState [CLogState]

pForce [bool↗]

    true if overwrite

Exceptions

[DirectoryNotFoundException↗]

GetDirSize(string)

Calcule la taille d'un repertoire

```
public long GetDirSize(string pPath)
```

Parameters

pPath string⍈

    Chemin du repertoire

Returns

long⍈

    la taille du repertoire en bytes

UpdateLog(CLogState)

```
public void UpdateLog(CLogState logState)
```

Parameters

logState CLogState

Namespace UnitTestJobs

Classes

JobsTestUnit

Class JobsTestUnit

Namespace: UnitTestJobs

Assembly: UnitTestJobs.dll

```
public class JobsTestUnit
```

Inheritance

object ← JobsTestUnit

Inherited Members

object.Equals(object) , [object.Equals(object, object)](object.Equals(object,_object)) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , [object.ReferenceEquals(object, object)](object.ReferenceEquals(object,_object)) , object.ToString()

Methods

CreateJob()

```
[Fact]
public void CreateJob()
```

DeleteJob()

```
[Fact]
public void DeleteJob()
```

SaveLoadJobManager()

```
[Fact]
public void SaveLoadJobManager()
```

SaveLoadJobs()

```csharp
[Fact]
public void SaveLoadJobs()
```

Namespace UnitTestStorage

Classes

StockageTestUnit

Class StockageTestUnit

Namespace: <u>UnitTestStorage</u>

Assembly: UnitTestStorage.dll

```
public class StockageTestUnit
```

Inheritance

<u>object</u> ← StockageTestUnit

Inherited Members

<u>object.Equals(object)</u> , <u>object.Equals(object, object)</u> , <u>object.GetHashCode()</u> , <u>object.GetType()</u> , <u>object.MemberwiseClone()</u> , <u>object.ReferenceEquals(object, object)</u> , <u>object.ToString()</u>

Methods

TestSerialisation()

```
[Fact]
public void TestSerialisation()
```