|  |
| --- |
|  |
| Petits pains |
| Application de rapport pour les CRA |
| *Spécification technique* |

|  |
| --- |
| Benoit Masson-Bedeau  24/10/2017 |

Table des matières

[1 Introduction 4](#_Toc496786872)

[2 Dossier Assets 4](#_Toc496786873)

[3 Dossier Command 4](#_Toc496786874)

[3.1 Fichier CommandHandler.cs 4](#_Toc496786875)

[3.2 Fichier CommandHandlerT.cs 4](#_Toc496786876)

[4 Dossier Converter 4](#_Toc496786877)

[4.1 Fichier BoolToVisibilityConverter.cs 5](#_Toc496786878)

[4.2 Fichier NullToBoolConverter.cs 5](#_Toc496786879)

[4.3 Fichier NullToVisibilityConverter.cs 5](#_Toc496786880)

[4.4 Fichier StringNullOrEmptyToBoolConverter.cs 5](#_Toc496786881)

[4.5 Fichier StringNullOrEmptyToVisibilityConverter.cs 5](#_Toc496786882)

[5 Dossier Data 6](#_Toc496786883)

[5.1 Fichier PetitsPainsStore.cs 6](#_Toc496786884)

[5.1.1 Propriétés 6](#_Toc496786885)

[5.1.2 Méthodes ReadConfig/WriteConfig 6](#_Toc496786886)

[5.1.3 Méthode WriteCroissantsLines/ReadCroissantsLines 6](#_Toc496786887)

[5.1.4 Méthode ReadPersons 7](#_Toc496786888)

[5.1.5 Méthode GetDefaultPeople 7](#_Toc496786889)

[6 Dossier Resources 7](#_Toc496786890)

[6.1 Fichier EmailTemplate.tt 7](#_Toc496786891)

[6.2 Fichier EmailTemplate.cs 7](#_Toc496786892)

[6.3 Fichier EmailTemplateExtended.cs 7](#_Toc496786893)

[7 Dossier Utils 7](#_Toc496786894)

[8 Dossier Model 7](#_Toc496786895)

[9 Dossier View 7](#_Toc496786896)

[10 Dossier ViewModel 7](#_Toc496786897)

Historique

|  |  |  |
| --- | --- | --- |
| Date | Version | Commentaire |
| 24/10/2017 | 1.0 | Création du document |

# Introduction

L’application *Petits Pains* consiste en une application permettant de saisir le rapport pour les CRA des personnes d’une équipe puis d’envoyer un email à ces derniers pour les informer de la synthèse.

Le tout, dans la joie et la bonne humeur : on n’est pas la Stasi 😊 !

# Dossier Assets

Dossier des ressources à utiliser dans le projet. Il contient :

* le fichier *credit.txt* est le fichier contenant un lien pointant vers l’auteur de l’image *croissant\_empty* ;
* les fichiers images sont utilisés pour représenter les statuts des emplacements.

# Dossier Command

Dossier dans lequel se trouve les classes de commande. Les classes de commande permettent de réaliser des *bind* sur des éléments de l’interface graphique, comme des boutons.

## Fichier CommandHandler.cs

*Command* qui implémente l’interface *ICommand*. Elle permet d’exécuter une fonction qui ne prend pas de paramètre en fonction d’une autre fonction qui détermine si la première fonction peut s’exécuter.

Le seul constructeur prend donc 2 paramètres :

* *methodToExecute* de type *Action* : méthode à exécuter ;
* *methodToDetermineCanExecute* de type *Func<bool>*: méthode permettant de déterminer si *methodToExecute* peut être exécutée.

## Fichier CommandHandlerT.cs

Même principe que **Fichier CommandHandler**. La seule différence est que *methodToExecute* prend un paramètre en entrée.

Le constructeur prend donc ces 2 paramètres en entrée :

* *methodToExecute* de type *Action<T>* : méthode à exécuter ;
* *methodToDetermineCanExecute* de type *Func<bool>*: méthode permettant de déterminer si *methodToExecute* peut être exécutée.

# Dossier Converter

Dossier contenant les *Converter*, permettant de convertir des objets d’un type particulier vers un type manipulable par l’interface graphique (et vice-versa).

## Fichier BoolToVisibilityConverter.cs

Converti un booléen vers un objet de type *Visibility* :

* booléen vaut *true* : *Visibility* en sortie vaut *Visible* ;
* sinon : *Visibility* en sortie vaut *Collapsed*.

Ceci permet de piloter la visibilité d’éléments de l’interface graphique grâce à un booléen.

## Fichier NullToBoolConverter.cs

Converti un objet vers un booléen.

Dans cette classe, un enum *NullToBoolDirection* est présent, et permet de déterminer le sens de conversion :

* *NullIsTrue* : un objet *null* doit être converti vers une valeur *true* ;
* *NullIsFalse* : un objet *null* doit être converti vers une valeur *false*.

La valeur de l’enum doit être passée dans le paramètre *Parameter*. La sortie est donc conforme à ce paramètre.

Si le paramètre est *null*, on considère que *NullIsFalse*.

## Fichier NullToVisibilityConverter.cs

Converti un objet vers un objet de type *Visibility*.

Dans cette classe, un enum *NullToVisibilityDirection* est présent, et permet de déterminer le sens de conversion :

* *NullIsVisible* : un objet *null* doit être converti vers une valeur *Visible* ;
* *NullIsCollapsed* : un objet *null* doit être converti vers une valeur *Collapsed*.

La valeur de l’enum doit être passée dans le paramètre *Parameter*. La sortie est donc conforme à ce paramètre.

Si le paramètre est *null*, on considère que *NullIsCollapsed*.

## Fichier StringNullOrEmptyToBoolConverter.cs

Converti une chaine de caractère *null* ou vide vers un booléen :

* la chaine est *null* ou vide : la valeur en sortie est *false* ;
* sinon : la valeur en sortie est *true*.

## Fichier StringNullOrEmptyToVisibilityConverter.cs

Converti une chaine de caractère *null* ou vide vers un objet de type *Visibility*:

* la chaine est *null* ou vide : la valeur en sortie vaut *Collapsed* ;
* sinon : la valeur en sortie vaut *Visible*.

# Dossier Data

Dossier contenant les classes permettant la manipulation des données.

## Fichier PetitsPainsStore.cs

Fichier permettant la manipulation des données de l’application.

Il s’agit d’une classe *static* dans laquelle se trouvent les méthodes permettant de :

* stocker les données ;
* lire les données.

### Propriétés

* *RootPath* :
  + disponible en lecture/écriture ;
  + si vide, retourne le chemin de l’application.
* *ConfigFilePath* :
  + disponible uniquement en lecture ;
  + retourne le chemin complet vers le fichier de config :
    - *[chemin de l’application]/petitspains.config*.
* *DefaultPeopleFileName* :
  + disponible uniquement en lecture ;
  + retourne le nom du fichier contenant les personnes par défaut :
    - *ListePersonnesDefaut.json*.
* *CroissantLinesFileName* :
  + disponible uniquement en lecture ;
  + retourne le nom du fichier contenant la sauvegarde du rapport :
    - *ListeLignes.json*.

### Méthodes ReadConfig/WriteConfig

Méthodes *public*.

Ces méthodes permettent de lire et d’écrire le fichier de configuration.

Le fichier de configuration est un fichier XML : il faut donc que ces méthodes écrivent/lisent les paramètres concernés au format XML.

### Méthode WriteCroissantsLines/ReadCroissantsLines

Méthodes *public*.

Ces méthodes permettent de lire et d’écrire la sauvegarde pour le rapport des personnes suivies.

Le fichier de sauvegarde du rapport est un fichier JSON : il faut donc que les méthodes sérialisent/désérialisent celui-ci.

Dans le cas où le fichier n’existe pas lors de l’appel de *ReadCroissantsLines*, les personnes par défaut sont chargées via la méthode **Méthode ReadPersons**.

### Méthode ReadPersons

Méthode *private*.

Cette méthode permet de lire le fichier des personnes par défaut au format JSON ; il faut donc désérialiser celui-ci.

Si le fichier n’existe pas, une liste vide de personne est retournée.

### Méthode GetDefaultPeople

Méthode permettant de récupérer la liste des lignes contenant les personnes par défaut à charger, lorsqu’un fichier de sauvegarde n’est pas présent par exemple.

Concrètement, cette méthode construit une liste de lignes contenant les personnes récupérées par **Méthode ReadPersons**.

# Dossier Resources

Dossier contenant les fichiers annexes nécessaires au fonctionnement de l’application.

## Fichier EmailTemplate.tt

Fichier permettant la génération d’un texte au format HTML.

Le fichier utilise les champs passés en paramètre du constructeur de **Fichier EmailTemplateExtended.cs** pour générer un email conforme à ce qui est décrit dans la SF.

### Configuration

* le HTML généré doit être du HTML5 ;
* le HTML doit être au format UTF-8 ;
* l’accès aux images dans un email se fait par l’attribut *cid:*. Exemple :

<img src="cid:croissantEmpty" />

## Fichier EmailTemplate.cs

Fichier autogénéré lors de l’enregistrement de **Fichier EmailTemplate.tt**.

Ce fichier ne doit donc pas être modifié manuellement.

## Fichier EmailTemplateExtended.cs

Fichier d’extension pour la classe autogénérée **Fichier EmailTemplate.cs**. Permet d’étendre cette classe *partial*.

### Champs

* *date*, de type *DateTime* :
  + *private* ;
  + date à laquelle le rapport est traité.
* lines, de type *ItemsChangeObservableCollection<Line>* :
  + *private* ;
  + liste des lignes à afficher dans le rapport.

### Constructeurs

Un seul constructeur à ajouter : le constructeur prend 2 paramètres :

* *date*, de type *DateTime* :
  + *private* ;
  + date à laquelle le rapport est traité.
* lines, de type *ItemsChangeObservableCollection<Line>* :
  + *private* ;
  + liste des lignes à afficher dans le rapport.

Le constructeur s’occupe juste de valoriser ses champs avec ce qui lui est transmis en paramètre. Ainsi, les paramètres deviennent accessibles dans **Fichier EmailTemplate.tt**.

# Dossier Utils

# Dossier Model

# Dossier View

# Dossier ViewModel