Module Projet

Présentation de l'application ppFolder

Cahier des charges et spécifications

Année 2020-2021

Yacine BOUFALA

Romain HUISDACK

Rayan KHALFOUN

Professeur référant : Madame Nadia Abchiche-Mimouni

Présentation

PPfoler est une application utilitaire ayant pour objectif d’aider ses utilisateurs dans la gestion de leurs fichiers. Elle est chargée de mettre de l’ordre dans les dossiers en appliquant un tri sur les fichiers en fonction de leurs extensions.

L'utilisateur a la possibilité de créer des règles regroupant des paquets d'extensions selon ses besoins et préférences. En effet, il y a autant d’utilisations possibles de ppfolder que d’utilisateurs, cela correspondrait alors à un maximum de personnes sans gêner leur méthode d’organisation.

Utilisation

L’utilisateur, en lançant l’application, arrive sur un page lui demandant d’indiquer le chemin correspondant au Dossier qu’il souhaite ranger.

L'application teste pour savoir si le chemin est bel et bien existant sur l’ordinateur de l’utilisateur.

Si oui, il est basculé sur la deuxième partie de l’interface graphique qui regroupe cette fois trois parties:

-Une zone ou sont gérées et affichées les règles

Dans cette partie l’utilisateur aura le choix d’ajouter une règle de tri qu’il pourra par la suite utiliser sans avoir à la retaper pour les autres Dossiers qu’il triera à l’avenir. En effet on a remarqué que certains paquets d'extensions comme « png, jpg, svg.. » sont souvent utilisés de cohortes. C’est dans l'optique de simplifier l'utilisation des ces groupes récurrents que les règles ont été ajoutées.

Si une règle ne convient plus à l’utilisateur il pourra la supprimer grâce à un bouton ajouté sur l’interface.

- Une zone où les extensions déjà mises en place sur le dossier sont résumées

La zone a pour objectif de visualiser les règles déjà mises en place. Elle affiche les sous-dossiers déjà créés par l'application pour que l’utilisateur puisse aisément se retrouver.

Cette fonctionnalité permet aussi, lorsque que l'utilisateur revient effectuer des modifications sur un tri, de grader une trace dans le temps pour lui permettre de se remémorer le travail déjà mis en place.

- Une zone qui affiche le contenu du dossier à trier

La zone centrale a pour objectif d'afficher les différentes extensions des fichiers présents dans le Dossier à trier. Elle a pour objectif d'aider l'utilisateur a visualiser le travail qu’il doit effectuer.

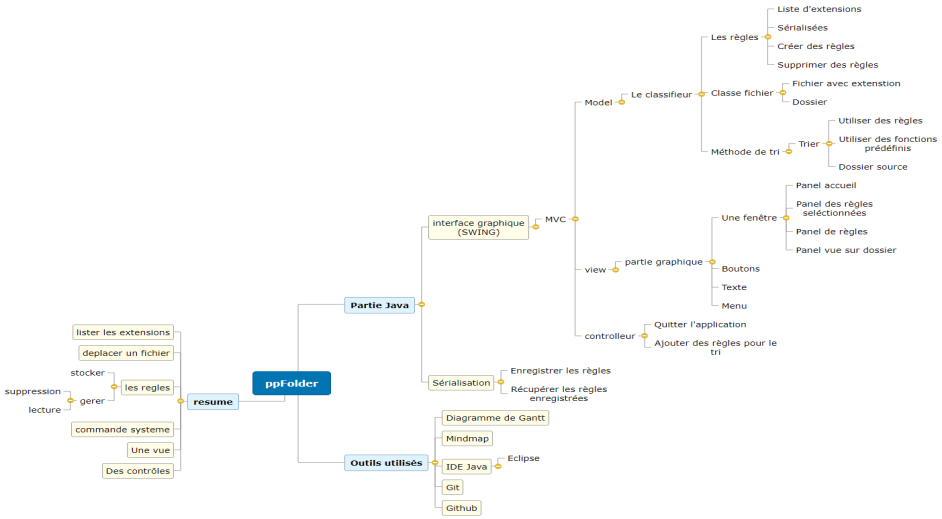
En effet il est toujours plus agréable de visualiser les choses, surtout quand il s’agit d’un travail organisationnel.

Une fois toutes les règles renseignées il faut appuyer sur le bouton « trier » pour lancer la procédure.

Voilà comment les différents espaces disponibles sur la fenêtre sont répartis. Ils répondent selon nous aux besoins de l’utilisateur tout en restant accessible et simple d'utilisation .C’est d’ailleurs dans cette optique que l'interface graphique a été créée : permettre au plus grand nombre d’utiliser ppdolder de manière intuitive et efficace.

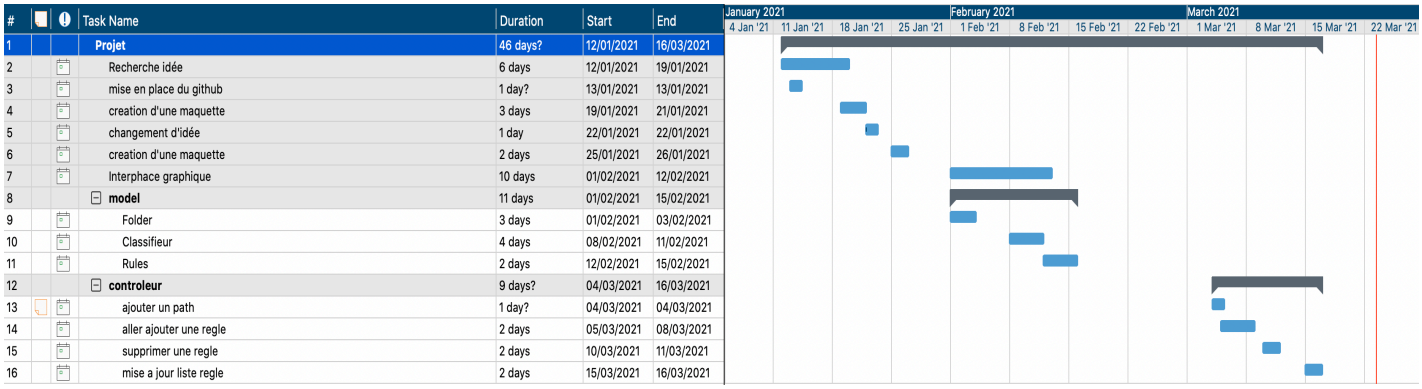
mindmap

La mindmap est un outil organisationnel qui nous a permis, dès le debut du projet,de bien fixer nos objectifs et donc de plus facilement visualiser la direction que notre équipe devait prendre



Gant

Le diagramme de Gantt permet grâce a un système de tableau et de schéma de se représenter les différentes étape d'un projet et permettre ainsi a l'équipe un meilleur suivie du travail effectué



#	Task Name	Duration	Start	End
1	**Projet**	46 days?	12/01/2021	16/03/2021
2	Recherche idée	6 days	12/01/2021	19/01/2021
3	mise en place du github	1 day?	13/01/2021	13/01/2021
4	creation d'une maquette	3 days	19/01/2021	21/01/2021
5	changement d'idée	1 day	22/01/2021	22/01/2021
6	creation d'une maquette	2 days	25/01/2021	26/01/2021
7	Interphace graphique	10 days	01/02/2021	12/02/2021
8	**model**	11 days	01/02/2021	15/02/2021
9	Folder	3 days	01/02/2021	03/02/2021
10	Classifieur	4 days	08/02/2021	11/02/2021
11	Rules	2 days	12/02/2021	15/02/2021
12	**controleur**	9 days?	04/03/2021	16/03/2021
13	ajouter un path	1 day?	04/03/2021	04/03/2021
14	aller ajouter une regle	2 days	05/03/2021	08/03/2021
15	supprimer une regle	2 days	10/03/2021	11/03/2021
16	mise a jour liste regle	2 days	15/03/2021	16/03/2021

Implémentation des instructions dans le code java :

Modèle

Partie fichiers :

Folder : -Permet de représenter un répertoire.

-Un objet Folder possède comme attribut son chemin d'accès.

-Pour construire une instance de répertoire (Folder), on utilise son constructeur qui prend en paramètre son chemin d'accès sur l'ordinateur en tant que String. Le constructeur permettra ainsi d'instancier un nouveau répertoire.

create_Dir : - Permet de créer un répertoire et de le nommer.

- Vérifie si le nom n'est pas déjà utilisé dans le même dossier parent. Si c'est le cas, la fonction renvoie « faux » et annule la procédure. Sinon, la méthode créer le répertoire à l'intérieur du Folder.

IsDirEmpty : - Prend le nom du dossier à analyser en paramètre.

- Test si le Dossier est vide. La méthode renvoie vrai si le répertoire contient un fichier. Sinon, elle renvoie faux.

removeDirectory : - Prend en paramètre un nom de répertoire à supprimer. Si le dossier est vide la méthode le supprime

Ainsi, avec cette classe, on peut faire des analyses à l'intérieur du répertoire ciblé de manière efficace ce qui permet ainsi de structurer la partie des fichiers de l'application qui est essentielle.

Partie règles :

La classe Rules qui implémente « Serializable » et « Iterable » a pour attribut une liste de règles et un identifiant de sérialisation.

La méthode itérateur est implémenté ce qui permet de parcourir la liste des règles et aussi d'utiliser les méthodes de l'itérateur pour optimiser la gestion des règles.

getRules: Permet de retourner l'ensemble des noms de règles dans la console.

Deux méthodes de suppression son aussi développées pour garantir la flexibilité de l'application.

La méthode removeRule prend un index en paramètre et supprime la règle se trouvant à cette position dans la liste des règles. Si la règle existe, elle est supprimée et la méthode renvoie vrai. Sinon la méthode renvoie faux.

L'autre méthode removeRule prend un nom de règle en paramètre et la liste des règles est parcourue. Si la règle est identifiée dans la liste des règles. Alors elle est supprimée et la méthode renvoie vrai. Sinon la méthode renvoie faux.

AddRule: Prend un nom de règle en paramètre. Si la liste des règles est vide, alors on ajoute la règle et on renvoie vrai. Sinon, si la liste de règles ne contient pas cette règle. On l'ajoute et on renvoie vrai. Dans le cas échant la méthode renvoie faux.

Pour garantir la conservation des règles deux méthodes de sérialisation sont ajoutées à la classe Rules.

serializeRules: Crée un nouveau fichier s'appelant « Rules.ser » dans le dossier où l'application se trouve. L'application va ensuite écrire l'instance de l'objet Rules dans le fichier de sérialisation. Une fois l'écriture terminée, on prend soin de vider puis fermer le flux de sortie.

readSerializedRules :Cette méthode est conçue pour lire le fichier « Rules.ser » se trouvant au niveau de l'application. Pour se faire on utilise un flux d'objets entrants qui va lire le fichier de sérialisation. Ainsi on pourra lire la valeur des règles enregistrées dans le fichier de sérialisation. On pourra affecter cette valeur à une liste de règles que l'on retournera. Si une erreur survient ou que le fichier de sérialisation n'existe pas on retourne une valeur nulle.

Rules : Classe utilisant la sérialisation ce qui rend l'application plus autonome. Cela conserve les valeurs des règles stockées par l'utilisateur. L'utilisateur pourra donc ajouter et supprimer des règles lors de l'utilisation de l'application.

Partie traitement des fichiers :

La classe Classifieur regroupe comme attributs : une liste de règles qui pourront être appliquées à un tri, un dictionnaire servant à identifier une liste de fichiers avec leur nom en retournant leur chemin d'accès ainsi qu'un chemin d'accès au répertoire qui devra être trié.

Le Constructeur du classifieur prend son chemin en paramètre. L'attribut de l'instance prendra donc ce nouveau chemin comme valeur.

get_extension est une méthode qui, à partir d'un nom de fichier, va récupérer son extension et la renvoyer. Si le fichier n'existe pas, la méthode renvoie une extension vide.

listFiles prend en paramètre le chemin d'un dossier à traiter. Cette méthode va ensuite ajouter au dictionnaire du Classifieur tous les chemins de ces fichiers avec leur extension.

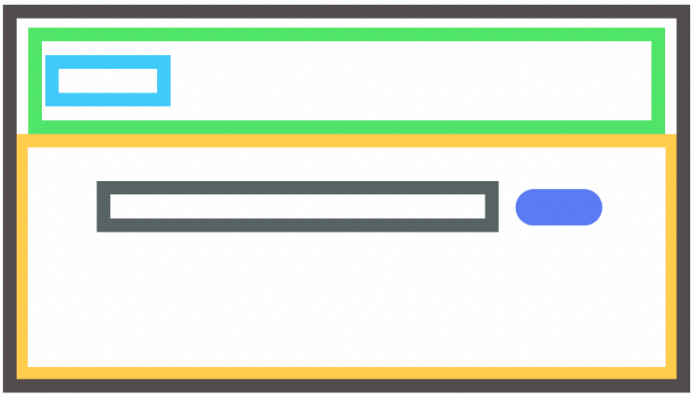
trier comme son nom l'indique permet de trier tous les fichiers en fonction des règles. Cela permettra de déplacer chaque fichier dans un dossier comportant leur extension en nom.

rulesToFolders va prendre l'initiative de créer un nouveau s'il n'existe pas en fonction des règles qui ont été analysées.

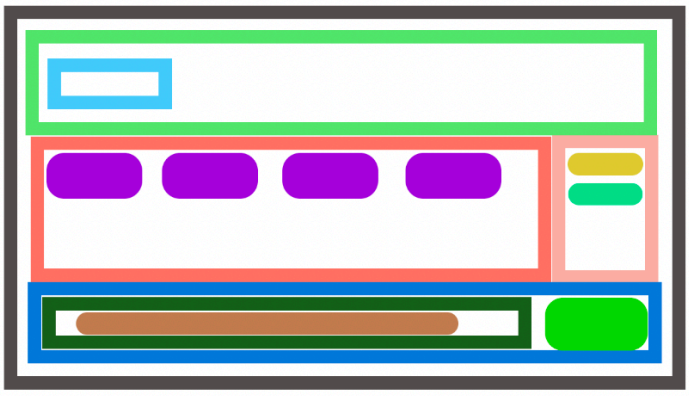
move_file est une méthode prenant en entrée le chemin du fichier à traiter et le chemin du répertoire qui accueillera le nouveau fichier. Si le chemin de la destination mène bien vers un dossier, alors le fichier sera déplacé et prendra la place d'un autre fichier possédant un nom identique pour éviter les erreurs. Si le déplacement est effectué, alors la méthode renvoie vrai. Sinon, elle renvoie faux.

La classe Classifieur est donc le corps de l'application. Cette classe possède en attribut qui est l'ensemble des règles à traiter ainsi que tous les fichiers du répertoire ciblé avec leur extension. Cette classe est donc l'atout principal de l'application ppFolder.

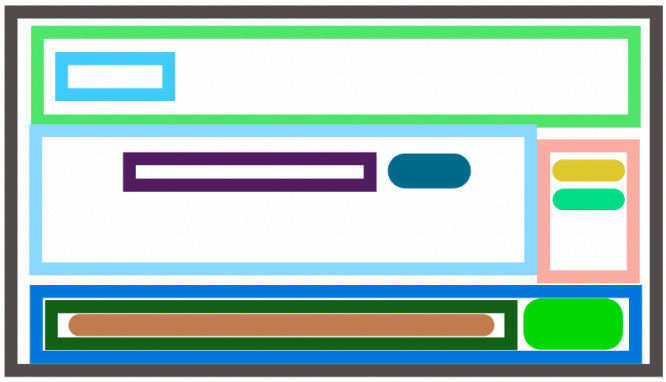
Partie interface graphique :



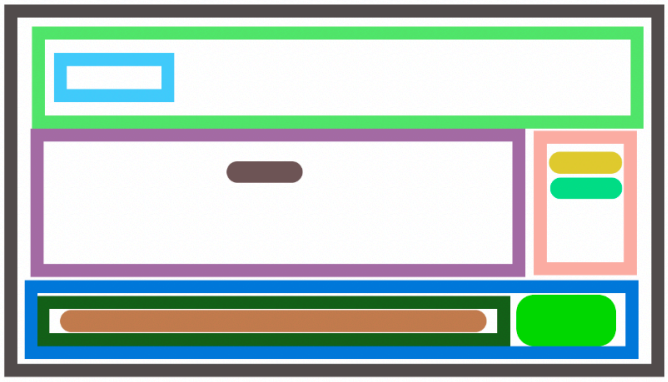
Page choix dossier



Page de tri



Page ajout des regles

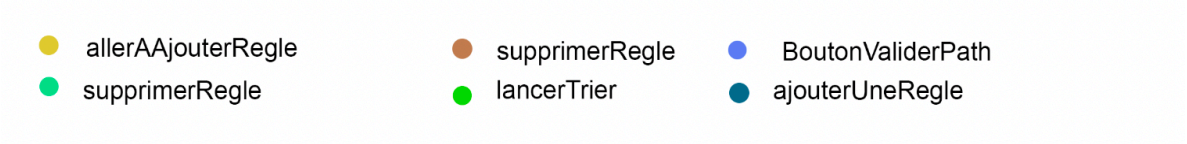


Page suppression des regles

Panel:

- panelFichiers (scrollable)
- panelRegles (scrollable)
- ppFolderInterface(North)
- fenetreDepart
- panelbas
- ppFolderInterface
- panelAjoutRegle
- panelSuppressionRegle
- panelRegleUtiliser (scrollable)

Bouton:



PlaceholderTextField:

Partie Controleur:

La partie controlleur sert a exécuter les actions faites sur les boutons de l'interface graphique, c'est ce qui apporte une interactivité avec l'utilisateur.

this.vue.ajouterUnPathListener(new ActionListener( ) : Cela correspond au bouton "valider" de la page du choix des chemins. En appuyant sur le bouton, le controlleur analyse s'il s'agit bien d'un path valide. Puis il récupère le chemin et ajoute les éléments de la vue suivante (page de tri).

this.vue.allerAajouteruneregleListener(new ActionListener( ): Nous permet de vider le panel fichier et de le remplacer par le panel ajoutRegle en appuyant sur le bouton « allerAAjouterRegle »

this.vue.allerASupprimerUneRegleListener(new ActionListener(): Nous permet de vider le panel fichier et de le remplacer par le panel suppressionRegle en appuyant sur le bouton « supprimerRegle »

this.vue.ajouterUneReglelistener(new ActionListener( ) :Cela correspond au bouton dans le panel ajoutRegle,il permet d'ajouter une règle si:
-Le champ rempli n'est pas vide
-La liste des règles n'est pas vide.
Ensuite on retourne sur le panel avec la nouvelle règles ajoutée

this.vue.supprimerUneRegleListener(new ActionListener( ): Supprime la règle sur le panel suppressionRegle en la choisissant dans la liste déroulante.
puis on revient sur la Page de tri

this.vue.lancerLeTri(new ActionListener() : Créer un classifieur en prenant le chemin récupéré au début et lance la méthode de tri

public void refresh_regle_selectionnees( ) : Permet d'actualiser le panel du bas avec les règles sélectionnées en redessinant tout les boutons
Cela ajoute aussi un listener sur chacun des boutons. Le listener ajouté permet de supprimer la règle seléctionnée

public void ajouter_boutons_regle( ) : Rafraîchit les règles dans le PanelRegles en rafraîchissant les boutons en leur mettant un listener qui permet d'ajouter une règle dans la liste de celle sélectionnée

dessin_regles(), public void dess_panelbas() , public void dessin_panel_fichier() : permettent respectivement de dessiner le panelRegles, le PanelFichier et le panelBas