DataMail - Gestion des mails

mail/mailing en ligne de commande

Jouer la commande :

```
send-mail path/to/mail.md
```

Le fichier `path/to/mail.md` qui définit toutes les données doit être correctement formaté.

Options de la ligne de commande

```
-s/--simulation     Pour faire simplement une simulation d'envoi

-e/--mail_errors    Pour procéder au dernier envoi avec les mails qui ont
                    échoué.
```

Fichier mail

On appelle “fichier mail” le fichier qui définit entièrement l’envoi à faire.

Il doit impérativement :

- être au format markdown
- définir ses métadonnées
- définir le texte du mail

Définition des variables

Il existe deux types de variable : les variables qui dépendent des destinataires (quand c’est un mailing-list) et les variables qui permettent de simplifier le code (typiquement : pour les images).

Définition des variables template

Les “variables-template” sont définies une fois pour toutes dans le message à envoyer. Elles fonctionnent de façon très simples, avec un identifiant (en général majuscule pour le repérer plus facilement) dans le texte et sa définition dans le corps du message. Par exemple :

```
---
...
SRPS = "Savoir rédiger et présenter son scénario"
---
Cher ami,

Avez-vous lu « SRPS » ? Si ce n'est pas le cas, je vous conseille
de l'acheter car « SRPS » est un livre intéressant pour la rédaction
du scénario.
```

Insérer une image

Les *variables-templates* permettent d'insérer de façon simple une image (en dur) dans le code consiste à utiliser une variable qui :

- commence par `IMG`,
- définit le chemin d'accès au fichier image.

Par exemple :

```
---
from = ...
to = ...
IMG1 = /path/to/mon/image.jpg
IMG1-alt = Son nom par défaut de l'image…
---
Bonjour,

Que penses-tu de cette image ?

IMG1

Cool, non ?
```

Ci-dessus, la variable `IMG1` sera remplacée par le code en dur de l'image de path `/path/to/mon/image.jpg`.

La variable `IMG1-alt` permet de définir la légende par défaut mais n'est pas obligatoire.

Définition des variables destinataire

Dans le message, elles sont repérées par la code template `%{nom}`. Par exemple :

```
---
To = ...
From = ...
Subject = ...
---
Cher %{patronyme},

Allez-vous mieux ?
```

Le nom de la variable est obligatoirement en minuscule, même si elle est définie en majuscule dans le fichier de données.

Dans le code, on peut utiliser les variables classiques (`mail`, `patronyme`, `fonction`) mais on peut aussi utiliser n'importe quelle propriété qui serait définie dans le fichier de données. **Mais attention** : il faut que cette donnée soit définie pour tous les destinataires. Par exemple, si tous les destinataires définissent la propriété `album`, on peut avoir :

```
---
To = ...
From = ...
Subject = ...
---
Cher %{patronyme},

Avez-vous terminé de lire %{album} ?
```

Messages sexués

On peut utiliser une sexualisation du message (différents suivant femme ou homme, quand la propriété `sexe` est définie) grâce aux *propriétés féminines* (ou "féminines"). Par exemple :

```
---
...
---
Ch%{ere} ami%{e},

Vous êtes trop bon%{ne} avec moi.

etc.
```

Note implémentation : ces propriétés sont définies dans la constantes `FEMININES` dans le fichier `constants.rb` dans le cas où il faille en ajouter.

Définition du ou des destinataires

On peut définir un ou plusieurs destinataires, par fichier ou par valeur explicite. Ces destinataires se définissent grâce à la métadonnée `To` de la manière suivante.

Par valeur explicite :

```
---
To = philippe.perret@yahoo.fr
---
```

Avec un patronyme :

```
---
To = Phil <philippe.perret@yahoo.fr>
---
```

Par valeur explicite avec plusieurs destinataires :

```
---
To = ["mail1@chez.lui", "mail2@chez.eux", "Phil <mailphil@chez.lui"]
---
```

Avec des valeurs explicites, un sexe et un patronyme précisés :

> Note : l'ordre importe peu, l'application est capable de reconnaitre le type de la donnée.

```
---
To = ["H,Patrick,patrick@gmail.com"]
---
```

Par liste d'adresses :

```
---
To = /path/to/liste/adresses.csv
---
```

Liste d'adresses dans fichier

Pour fonctionner avec **MailManager**, un fichier contenant une liste d'adresses doit respecter certaines règles :

- Si c'est un fichier `YAML`, ça doit être une liste (`Array`) d'éléments qui définissent tous, au minimum, la propriété `:mail` (ou `'mail'` ou `'Mail'` et la propriété `:sexe` définissant le sexe du destinataire, par `F` ou `H`.
- Si c'est un fichier `CSV`, il doit impérativement :

- utiliser la **virgule** comme délimiteur de données,
- posséder une entête avec le **nom des colonnes**,
- définir la colonne `Mail` et la colonne `Sexe` (valeur `H` ou `F`),
- il peut définir la colonne `Patronyme` avec la patronyme de la personne,
- il peut définir la colonne `Fonction` définissant la fonction du destinataire.

Utilisation dans une application ruby

Prise en main rapide

```ruby
require 'mail_manager'


MailManager.send('/Users/phil/lemail.md')
```

Avec le fichier au chemin `path_mail_file` qui contient :

```
---
from = philippe.perret@yahoo.fr
to =   phil@atelier-icare.net
---
Bonjour à toi, Phil,

Comment ça va ?

Phil
```

Définition de la police et de la taille

Utiliser `font_family = ...` et `font_size = 14pt` dans les métadonnées.

Par défaut, la police est 'Times' et la taille est '14pt'.