Manuel français du runner

La commande `run` permet d'installer sur le bureau, dans les applications une configuration de travail quelconque.

Création de la nouvelle configuration de travail

Jouer dans un Terminal (n'importe où) :

```
> run
```

... choisir "Nouvelle configuration de travail" et suivre la procédure.

Configuration d'un travail

Nom humain de la configuration de travail

```yaml
---
name: "Nom du travail"
# ...
```

> C'est ce nom qui apparait dans la liste pour choisir un travail.

Dossier par défaut de la configuration

Si ce dossier est défini, il sera utilisé dès qu'un chemin d'accès ne sera pas trouvé tel quel.

```yaml
---
# ...
folder: "path/to/existing/folder"
# ...
```

Définition des étapes de la configuration

Tout ce qu'il y a à installer se trouve dans la propriété `setup` du fichier `YAML` de la configuration de travail.

> Pour ouvrir ce manuel, jouer `run -h/--help`

Ce paramètre contient les étapes (`steps`) à jouer pour obtenir une installation complète.

Configuration d'une étape

Une étape est avant tout définie par son `:type` (cf. ci-dessous tous les types d'étape). C'est la première chose à définir.

```yaml
:setup:
  - type: <type de l'étape>
```

Chemin d'accès dans une étape (dossier par défaut)

De nombreuses étapes nécessitent de définir un `:path`, chemin d'accès à un fichier ou un dossier. L'écriture de ce path peut être absolu, mais il peut également être simplifié en définissant un dossier par défaut dans la propriété générale `:folder` de la configuration du travail.

Par exemple :

```yaml
---
:name: "Configuration du travail à faire"
:folder: "path/absolu/to/folder/default
:setup:
  - type: open
    name: "Ouverture du dossier par défaut
    path: "."
    description: "Cette étape ouvrira le dossier :folder"
```

Étapes optionnelles

Par défaut, toutes les étapes sont jouées. Mais une étape peut être optionnelle, dans ce cas, le programme demande s'il faut la jouer avant de le faire ou non.

Pour indiquer qu'une étape est optionnelle, on ajoute `opt: true` à sa configuration. Pour pouvoir être désignée, il lui faut également un `:name`. Si elle contient aussi une `:description`, celle-ci sera discrètement affichée pour préciser l'étape.

Donc une étape optionnelle peut ressembler à :

```
:setpup:
  - type: open
    path: path/mon/fichier
    opt: true
    name: "Ouverture du fichier d'aide"
    description: |
      Ce fichier contient des renseignements qui peuvent
      aider à comprendre l'étape et savoir ce qu'elle va
      faire précisément.
```

Tous le types d'étapes

Ouverture d'un fichier

Pour ouvrir un fichier spécifique :

```
:setup:
  - type: open
    path: path/to/file # peut être relatif à :folder
    app: application optionnelle # sinon l'application par défaut
    bounds: [top, left, width, height]
```

Ouverture d'un dossier

```
:setup:
  - type: open
    path: path/to/folder # peut être relatif à :folder
    bounds: [top, left, width, height]
```

Ouverture d'un dossier dans l'IDE

On peut ouvrir un dossier dans l'IDE courant qui doit être défini dans `./lib/constants.rb` dans la constante `IDE` et définir la commande à utiliser dans `IDE_CMD`.

```
IDE = "Nom de l'IDE"
IDE_CMD = "<commande> '%s'" # %s sera remplacé par le :path défini
```

Par exemple :

```
IDE = "Sublime Text"
IDE_CMD = "subl -n '%s'"
```

Définition dans le fichier setup du travail :

```
setup:
  - type: open
    app:  IDE
    path: path/to/folder
```

Jouer un script utilitaires

Ce script doit se trouver dans le dossier `scripts` du dossier `Run` ou être spécifié par chemin d'accès complet. Il reçoit toujours les arguments définis par `:args` comme un json-string, donc comme une table avec des clés string. Si `args` n'est pas défini, il appelle le script sans argument.

```
:setup:
  - type: script
    path: mon_script
    args: '{"key":"value", "key2":"value2",...}' # format JSON
```

Jouer du code à la volée

```
:setup:
  - type: code
    lang: ruby # le langage ('ruby', 'bash', 'python', 'applescript')
    cmd: "puts('le code ruby à evaluer')"
```