Rapports de discussions de recherche et développements

Comparaison `tiramisu` et `Créole`

Rédacteurs

Gwenaël Rémond (gremond@cadoles.com)
Emmanuel Garette (egarette@cadoles.com)

Référence

```
tiramisu/doc/eole-reports
git clone ssh://gitosis@git.cadol.es:2222/tiramisu.git
```

ETATS ET STATUTS DES OPTIONS DE CONFIGURATION

état des variables et lisibilité de l'API

Creole

`EoleVar()`

- `get_value()`
- `get_final_value()`
- `get_final_value_at_index()`
- `check_value()`
- `get_prec_value()`
- `get_calculated_value()` -> automatique

tiramisu

`Option()`

- **aucune API** d'accès à la valeur d'une option au niveau de l'option de configuration
- `option.getdefault()`
- `option.setoption(config, value, owner)`

variables "automatiques"

si `owner` == 'auto', la variable est automatique et la configuration le sait, elle lance alors les fonctions de calcul à chaque évaluation

dans Créole, c'est validé aux niveau de la variable par un appel à `eval_func()`

Accès suivant les états de la configuration

- disabled
- hidden
- mode (normal/expert)
- obligatoire (mandatory)
- ...
- `EoleVar.hidden`
- `EoleVar.disabled`

pas d'objet `Family` dans Créole donc l'organisation des hiérarchie de hidden est opaque

- `EoleDict.families['hidden']` pour avoir accès à l'état d'une famille

dans Tiramisu

- `hidden` au niveau `Option`, `OptionDescription` et **aussi** au niveau de la configuration ce qui permet d'avoir des états (inexistant dans `Créole`)

`hidden_if_in`, `hidden_if_not_in`

La notion est généralisée dans tiramisu avec les `requires`.

Dans Créole : très difficile de conserver une cohérence des `hidden_if_in` quand il y en a plusieurs.

Dans Tiramisu : validation et levée d'exception si les **requirements** sont incohérents, action inverse si aucun requires n'est matché.

exemple de requires

```
<family name="clamav">
<variable name="activer_clam">

<variable name="activer_clam_exim">
<valeur>non

<variable name="activer_clam_samba">
<valeur>oui

<condition name='hidden_if_in' source='activer_clam_exim'>
<param>non
<target type='variable'>activer_clam
<!-- ça hide (momentanément)-->

<condition name='hidden_if_in' source='activer_clam_samba'>
<param>non
<target type='variable'>activer_clam
<!-- ça show (et c'est le dernier qui a raison) -->
```

résultat

`activer_clam` est visible, c'est la dernière condition qui a raison
avec tiramisu, `activer_clam` **dans les même conditions**, est cachée.

```
>>> activer_clam = StrOption('activer_clam', 'activer clamav',
            requires=[('activer_clam_exim', 'non', 'hide'),
                      ('activer_clam_samba', 'non', 'hide'),])
>>> config.clamav.activer_clam_exim = 'non'
>>> config.clamav.activer_clam_samba = 'oui'
>>> config.clamav.activer_clam
>>> Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
    HiddenOptionError("trying to access to a hidden
option:activer_clam")
>>>
```