Documentation développeur de FullSWOF_UI

Version 1.01.01 (2015-03-17)

2015-03-17

Création d'un arbre de configuration

L'architecture de FullSWOF_UI est destinée à permettre aisément de modifier le nombre et la nature des paramètres d'entrée utilisés par FullSWOF, tout en permettant à l'utilisateur de choisir entre différentes configurations. Les configurations disponibles actuellement correspondent à FullSWOF_1D et FullSWOF_2D. Pour assurer la rétro-compatibilité, il est recommandé de créer une configuration pour chaque version majeure de FullSWOF plutôt que de modifier les configurations 1D et 2D fournies.

Créer une configuration

Toutes les classes nécessaires pour la création d'une configuration sont situés dans le package *model*. Pour la plus d'informations, consultez la documentation de ces classes.

Une configuration est un arbre, constitué d'objets de type *Node*. Bien que la création d'un noeud soit simple, décrire une configuration complète couvre souvent de nombreuses lignes de code. Pour la clarté du code, il est recommandé de faire créer cette configuration par une méthode statique située dans une classe dédiée uniquement à la création de cet objet. Cette méthode renverra la racine de l'arbre, à laquelle on aura attaché les autres noeuds. Placez cette classe dans le package *model.definition* où se trouvent déjà les classes dédiées à la construction des configurations 1D et 2D.

Chaque élément visible du panneau de paramétrage correspond à un noeud. Ceci inclut aussi bien les paramètres proprement dits (*ExternalNode*) que les groupes de paramètres (*InternalNode*) comme par exemple les différents onglets du panneau de paramétrage ou des groupement de paramètres sur une seule page (par exemple le groupe « limite gauche » situé dans l'onglet « Limites »).

Création de la racine et des noeuds internes

Commencez par créer la racine de l'arbre, de type *RootNode*. C'est sur cet objet que l'on devra pointer pour faire référence à l'ensemble de l'arbre.

```
Node root = new RootNode(String nom, String description);
```

Attention le nom de ce noeud est celui de la configuration, il est parfois utilisé par le code, par exemple pour ouvrir un fichier anciennement créé. Veuillez à ne pas le changer et à ne pas utiliser un nom déjà employé par une autre configuration.

Puis créez les noeuds internes et ajoutez les au noeud parent :

```
Node node1 = new InternalNode(String nom);
```

```
root.addNode(node1);
```

Le premier niveau de noeuds internes constituera les onglets dans le panneau de paramétrage, tandis que les niveaux inférieurs seront affichés comme des cadres sur la page. Il n'y a pas de limite au nombre de niveaux imbriqués.

Création des paramètres

Une fois les groupes de paramètres créés, il faut leur ajouter les paramètres proprement dit. Tous les paramètres dérivent de la classe abstraite *ExternalNode*. Pour une description détaillée de chaque type de paramètre, consulter la documentation Doxygen de la classe *ExternalNode* et des classes qui en dérivent. Voici toutefois un résumé des classes les plus utiles :

- *FieldParameter* : champ texte acceptant n'importe quelle valeur
- *FileParameter* : permet de spécifier le chemin vers un fichier
- *IntegerParameter* : Nombre entier (avec possibilité de définir l'intervalle des valeurs acceptées)
- FloatParameter : Nombre décimal (avec possibilité de définir l'intervalle des valeurs acceptées)
- MultipleChoiceParameter : Paramètre avec un ensemble fini de valeur possibles (affichés sous forme de menu déroulant). Pour ajouter des valeurs, utilisez la commande addPossibleValue(String nom, String valeur)

Les autres classes étendant ExternalNode sont des paramètres au comportement particulier (tels que l'extension du dossier de sortie) ou des noeuds externes qui ne sont pas des paramètres (comme les noeuds utilisés pour créer des fichiers d'entrée).

Dépendances

Pour que la valeur d'un noeud enclenche une réaction sur d'autres noeuds, il est nécessaire de créer une dépendance (classe abstraite *Dependency*). Une dépendance a toujours un noeud « maître », un noeud « esclave » et une valeur cible. Lorsque la valeur du noeud maître est égale à la valeur cible, un changement est effectué sur

le noeud esclave. Cet outil est particulièrement utile avec les paramètres à choix multiple. Un choix particulier peut ainsi désactiver d'autres paramètres ou changer leur valeur. Toutefois les dépendances peuvent être utilisées avec tout type de paramètre.

L'exemple suivant permet par exemple de désactiver le paramètre node2 lorsque le paramètre node1 a la valeur « 4 ». La simple création de l'objet suffit à rendre la dépendance active.

```
new DisablingDependency(node1, node2, "4");
```

Le code actuel définit trois sortes de dépendances :

- *DisablingDependency :* permet de désactiver le noeud esclave pour une certaine valeur du noeud maître. Attention le noeud esclave n'est pas réactivé automatiquement si le noeud maître change de valeur par la suite. Pour simuler ceci vous devez utilisez *EnablingDependency* sur les autres valeurs possibles du noeud maître.
- *EnablingDependency :* offre la fonctionnalité inverse.
- *SettingDependency :* modifie la valeur du paramètre cible lorsque le paramètre principal a une certaine valeur.

Activer la configuration

Une fois la configuration créée, il faut l'ajouter à la liste des configurations utilisées. Celle-ci se trouve dans la classe *io.Procedures,* sous la forme d'une variable (tableau) nommée *AVAILABLE_CONFIGURATIONS*. Ajoutez votre configuration à ce tableau sous la forme d'un appel à la fonction de création de la configuration. C'est la racine de l'arbre de configuration qui doit être retournée par cette fonction.

Il est nécessaire de compiler (commande javac) la classe que vous avez créé ainsi que la classe *io.Procedures* et de placer les fichiers .class produits dans l'archive jar. À chaque package correspond un répertoire où placer le fichier .class. Comme le reste du projet les nouvelles classes devront être compilées en utilisant Java 6 (ou supérieur).

Si vous souhaitez plutôt recompiler l'ensemble du projet, il est nécessaire d'utiliser l'outil d'export d'Eclipse (Export > Runnable jar file). C'est en effet un des seuls outils qui permet de créer un jar contenant d'autres jar, ce que n'autorise pas la commande habituelle java -jar.

Localisation des paramètres

Afin de conserver l'aspect internationalisé de FullSWOF_UI, mieux vaut ne pas écrire les noms et descriptions des paramètres directement dans la classe. Il est recommandé de créer un fichier de localisation pour votre configuration et de le placer dans un répertoire qui lui est propre (cf. ci-dessous).

Pour accéder à cette localisation, créez un *ResourceBundle* de la manière suivante (cet exemple suppose que le fichier de localisation par défaut se situe dans le dossier */l10n/maconfig/* et se nomme *Config.properties*):

```
ResourceBundle messages = ResourceBundle.getBundle("l10n.maconfig.Config", Start.currentLocale);
```

Cette commande se chargera de récupérer les messages dans la langue approprié si celle-ci est disponible. Ne précisez donc jamais de code pays dans la méthode de récupération du *ResourceBundle*.

Ensuite, plutôt que d'écrire directement le nom du paramètre dans la classe, écrivez :

```
messages.getString("cle");
```

votre fichier de localisation doit alors contenir la ligne :

```
cle = Nom du paramètre
```

La seule exception à cette règle est le nom de racine de la configuration (*RootNode*) qui doit être écrit directement dans la classe et différer des autres noms de configuration.

Pour plus de précisions, consultez la documentation Oracle de la classe *ResourceBundle*.

Localisation

FullSWOF_UI est entièrement internationalisé, c'est à dire que les messages utilisateurs ne sont pas écrits directement dans le code. L'application charge à l'exécution les messages depuis un fichier, ce qui permet de pouvoir choisir aisément la langue. La localisation est le fait de fournir un fichier contenant des traductions des messages dans une langue spécifique. L'interface est actuellement localisée en français et en anglais. L'ajout d'une nouvelle localisation peut se faire sans recompilation de l'archive jar. Il suffit d'ajouter les fichiers de localisation pour voir apparaître un nouveau choix de langue dans les préférences de l'interface.

Les fichiers de localisation sont de simples fichiers texte avec une liste de paires clé/valeur. Pour créer un fichier de localisation, copiez le contenu du fichier par défaut (cf. nommage des fichiers) et remplacez les valeurs par la traduction appropriée. La clé doit rester identique. Si une clé est absente de votre fichier, l'interface utilisera la valeur qui y est associé dans le fichier par défaut.

Certains mots (par exemple les réponses « Oui » et « Non », « Annuler » des boîtes de dialogues) ne sont pas définis par des fichiers de localisation mais par la machine virtuelle Java. Il est possible que la machine virtuelle installée par l'utilisateur ne comporte pas la traduction de ces mots dans la localisation choisi. Dans ce cas on

verra apparaître ces mots dans une autre langue que celle choisie par l'interface (la langue par défaut de la JVM).

Nommage des fichiers

Les fichiers de localisation sont répartis dans plusieurs dossiers situés dans le répertoire */l10n* de l'archive jar. Dans chaque dossier se trouvent plusieurs fichiers dont le nom suit toujours le motif *nom_codePays.extension* (par exemple *UIMessages_fr.properties*). Pour qu'un fichier de localisation soit pris en compte, il doit suivre ce motif. Le code pays en question est un code à deux lettres défini par la norme ISO 639.

Par ailleurs chaque dossier contient un fichier sans code pays. Il s'agit de la localisation par défaut. Si une clé de traduction n'est pas disponible dans un fichier de localisation créé, c'est la clé contenue dans le fichier par défaut qui sera utilisé à la place.

Les différents fichiers de localisation

Les fichiers de localisation sont répartis dans plusieurs dossiers, chaque dossier correspondant à un type de fichier et contenant les différentes localisations disponible pour ce type de fichier.

- */l10n/ui* contient les fichiers *UIMessages*. Il s'agit des messages principaux de l'interface. Si vous souhaitez ajouter une localisation, vous devez absolument créer un fichier *UIMessages* avec le code pays approprié. C'est la présence de ce fichier qui permettra l'apparition du choix de la langue dans les préférences. Notez que ce fichier, comme les autres, peut être vide ou incomplet, l'interface utilisera alors les traductions du fichier par défaut.

- */l10n/config* contient les fichiers de localisation propre à chaque configuration de paramètres (un dossier par configuration : FullSWOF_1D, FullSWOF_2D...). Il est recommandé de créer un nouveau dossier pour chaque configuration définie plutôt que de réemployer le fichier d'une autre configuration, même si de nombreuses clés sont communes.

- */l10n/manual* contient le manuel utilisateur affiché par l'interface. Contrairement aux autres fichiers, il s'agit d'un fichier HTML dont le contenu est affiché dans une fenêtre de l'application.