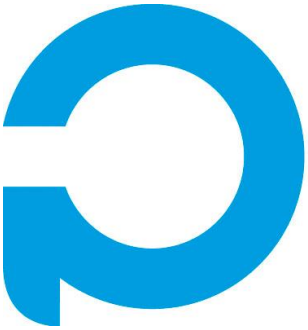
GENERATEUR SCXML -> JAVA



POLYTECH©
NICE-SOPHIA

15/04/2017

Manuel utilisateur (Ajroud Rami G1)

FONCTIONNALITES PROPOSEE

Le code fourni est capable de générer une machine à états Java à partir d'un fichier SCXML la décrivant. Il supporte dans sa version 1.0 les fonctionnalités suivantes :

- Support de machine à états simples (non hiérarchique, non parelles) :
  Je justifie ce choix par le fait qu'il est possible de transformer une machine à états parallèles et/ou hiérarchique en prétraitant le fichier SCXML. On pourra ensuite utiliser ce fichier dans le générateur.
- Support de priorité des évènements : Les évènements reçu et émis par la machine sont stockés dans une pile. Les évènements « send » ont priorité sur les « raise » et sont traité avant.
- Support des évènements temporisé: suivant la définition SCML, l'attribut « delay » doit être défini dans le tag « send » ou « raise ». La valeur est interprété sera en millisecondes. Si la transition (ou l'état) déclenche un évènement temporisé, l'évènement sera émis après le temps défini.
- Support d'appel de code métier : L'utilisateur a la possibilité de lié un évènement à l'appel d'une méthode Java. Pour cela il devra fournir à son la machine : l'objet sur lesquelles la méthode doit être appliqué, la méthode, ses arguments (si il y en a).

Exemples et Tests

Pour chacune des fonctionnalités citées ci-dessus il existe un ou plusieurs tests d'intégrations. Ces tests utilisent des fichiers scxml différents (pour différencier la fonctionnalité testée) situé dans le répertoire src\main\resources\integrationTests. Les classes Java correspondant à cette machine sont générées et compilé dynamiquement au lancement des tests. Pour lancer les tests tapez :

```
$ mvn clean test
```

Il y a aussi un package exemple avec un exemple d'implementation. Seulement il connait quelques problemes du a l'utilisation de Thread.

MODE D'EMPLOI

Compilation du générateur

La compilation se fait à l'aide de maven. La commande à taper dans le répertoire du projet est la suivante :

```
$ mvn clean compile assembly:single
```

Cela génèrera un fichier jar dans le répertoire target nommé :

« scxml-to-fsm-1.0-SNAPSHOT-jar-with-dependencies.jar »

Exécution du générateur

Pour générer la classe java associé à la state machine scxml, tapez :

```
$ java –jar scxml-to-fsm-1.0-SNAPSHOT-jar-with-dependencies.jar rep fichierSCXML
```

Ou rep et fichiers SXML sont respectivement le chemin vers le répertoire ou va être généré le fichier java et le chemin vers le fichier scxml (chemins absolus).

Mise en place du projet

La classe générée par le jar aura besoin d'être copié dans votre projet. Cependant pour fonctionner elle aura besoin des certaines classes. Vous devrez donc suivre deux étapes simples :

- Dans un premier temps copier la classe générée (GStateMachine.java) dans le projet (note : le package est par défaut la racine, vous devrez le changer manuellement si vous bougez la classe)
- Puis copier le package statemachine dans votre projet.

Utilisation de la state machine généré

Pour utiliser toutes les possibilités de la state machine vous avez la possibilité de créer une classe fille pour pouvoir surcharger le constructeur et ajouter de nouvelle méthodes ou l'utiliser tel quelle. Les différentes utilisation de la classe sont définies dans l'interface statemachine.StateMachine et sont :

```
void start();
void stop();
void notifyEvent(String event);
void notifyEvent(Event e);
void connectToEvent(String eventName, Callable callable);
void connectToEvent(String eventName, Object object, Method method);
void connectToEvent(String eventName, Object object, Method method, Object[]
args);
```

- start: Permet de démarrer la state machine. Elle doit être appelle avant son utilisation, autrement la state machine ne répondra pas aux évènements mais enregistrera ceci dans sa pile. Lors de l'appel de start(), si la pile n'est pas vide chaque évènement enregistré sera traité dans l'ordre chronologique d'arrivée.
- stop: arrête exécution des évènements. Cet état est identique à l'état de la machine avant l'appel de start.
- notifyEvent: Cette fonction permet d'envoyer un évènement à la machine. Note: pour la version avec String en paramètre, l'évènement sera un send.
- connectToEvent: cette méthode permet de connecter une méthode à exécuter lors de la réception d'un évènement. Une seule méthode peut être liée. Note

importante: le nom de l'évènement à lier a la méthode doit être un évènement émis par la machine et non reçu.