TP1: Méta-modèles et outils de création

Rappel

Tout les TPs sont à faire sur UN SEUL repo gitlab. Avant toutes choses, créez un dépôt et ajoutez Anne Etien et Vincent Aranega comme développeurs à votre dépôt. Vous pouvez travaillez en binôme, mais gardez en tête qu'un des TPs sera noté et à faire en temps limité pendant la séance de TP, donc soyez tous à l'aise avec la façon de travailler et surtout, créez bien tous un dépôt pour vous.

Pré-requis

Dans ce premier TP, vous allez créer des méta-modèles manuellement en utilisant les outils proposés dans Eclipse. Installer les outils de modélisation directement dans une instance d'Eclipse est quelque chose de pénible. Heureusement, il existe des "saveurs" d'Eclipse spéciales avec les outils déjà pré-installés. Dans notre cas, nous allons utiliser la version "Eclipse Modeling Tools".

1. Téléchargez "Eclipse Modeling Tools" à l'adresse suivante: https://www.eclipse.org/downloads/packages/ et décompressez le (il faut vraiment aller chercher la "flavour" modeling sur la page).

Modélisation du méta-modèle de Java simplifié

Dans un premier temps, vous allez modéliser le méta-modèle de Java simplfié que nous avons vu en cours.

1. Assurez-vous que vous avez bien tous les concepts et re-modélisez les sur papier si nécessaire.
2. Une fois Eclipse ouvert, créez un "Ecore Modeling Project" (`File→new→Others`) et nommez-le `simplejava`.

Vous pouvez remarquez que toute une hiérachie de répertoire et de "configuration" est automatiquement créée. Le projet créé est automatiquement paramétré avec les différentes dépendances nécessaires pour le bon fonctionnement de votre projet de méta-modélisation. Pour l'instant, vous allez uniquement travailler dans le répertoire `model` généré. Il contient trois fichiers, un `.aird`, un `.ecore` et un `.genmodel`.

- Le `.aird` est le fichier avec la représentation graphique de votre méta-modèle.
- Le `.ecore` est le fichier contenant la définition de votre méta-modèle.
- Le `.genmodel` est le fichier qui servira plus tard à générer le code de votre méta-modèle.

Pour ce TP, vous allez uniquement travailler avec le `.aird`, mais vous allez d'abord vérifier certaines

informations sur le `.ecore`.

1. Déroulez dans l'arbre le `.ecore` et cliquez sur le package `simplejava` et vérifiez dans l'onglet `Default` de la fenêtre `Properties` que toutes les informations sont bien remplies (préfix, URI et nom). Si elles ne le sont pas, remplissez les en utilisant `simplejava` comme nom et préfix et `http://simplejava/1.0` comme URI.
2. Ouvrez maintenant le fichier de représentation graphique de votre méta-modèle et modélisez le méta-modèle de Java simplifié (pour modifiez des informations d'élements, vous pouvez le faire dans l'onglet `Semantic` de la fenêtre de propriétés). **Pensez bien à prefixer tout vos concepts par un `J`** (par ex: `JClass` pour `Class`), pour éviter qu'il y ait des clash de noms par la suite lors de la génération de code Java et/ou Python (dans les prochains TPs).
3. Une fois terminé, ouvrez le `.ecore` (l'éditeur réfléxif générique devrait s'ouvrir) et vérifiez que tous les concepts sont bien présents.
4. De manière générale, il est important que vous validiez bien votre `.ecore`. L'opération de validation va bien vérifier que les informations que vous avez entrée dans votre méta-modèle sont bonnes (pas d'espaces en trop, de caractères non autorisés,...etc). Si vous ne le faites pas et que des erreurs se sont glissée dans votre méta-modèle, il y a de très fortes chances que les opérations suivantes s'appuyant sur votre méta-modèle (création d'instances, génération de code,...) vous retourne une erreur incompréhensible. Pour bien valider le méta-modèle, une fois que vous l'avez ouvert avec l'éditeur réflexif, faites un clic-droit sur la racine de votre méta-modèle (le `EPackage`) et choisissez `validate`. Si des problèmes sont relevés, ils sont indiqués et des labels sur les objets problématiques apparaissent dans l'éditeur réflexif. Corrigez les et revalidez votre modèle jusqu'à ce que le validateur vous assure que tout va bien. Attention, le fait que votre méta-modèle valide n'indique pas que votre méta-modèle représente correctement les concepts, cela veut juste dire que votre méta-modèle respecte bien le méta-méta-modèle et qu'aucune information entrée ne devrait poser problème par la suite.

Maintenant que votre méta-modèle est créé, pour vérifier que tout "est bon" dans votre méta-modèle, vous allez créer plusieurs modèles conformes. Pour créer des méta-modèles dynamiquement, sans générer de code, vous allez devoir choisir depuis la vue arborescente de votre ecore le concept qui représente la "racine" de votre modèle. Dans un modèle, il faut toujours considérer un conteneur principal qui est le conteneur de tous les éléments de votre modèle (ici, des packages java par exemple).

1. Ouvrez la vue arborescente et sélectionnez votre concept racine, puis faites un clic-droit sur l'élément, puis `Create dynamique instances···`. Cela va vous créer un fichier `.xmi`.
2. Ouvrez le fichier `.xmi` il devrait s'ouvrir avec l'éditeur générique réflexif (le même que lorsque vous ouvrez directement le `.ecore`).
3. Créez un premier modèle avec quelques packages et classes.
4. Créez maintenant un autre modèle (clic-droit...etc) et modélisez un autre modèle conforme au méta-modèle de java simplifié.

Modélisation du méta-modèle de système de fichier

Pour cet exercice, vous allez créer un nouveau "Ecore Modeling Project" avec le nom "metamodels". Ce projet servira à mettre tout vos autres méta-modèles (pas uniquement le java simplifié).

1. Créez le nouveau projet
2. Dans le répertoire `models`, créez un nouveau méta-modèle ecore: `File→new→Others→Ecore Model` nommé `filesystem.ecore`, puis ouvez le dans l'éditeur arborescent, sélectionnez le package à l'intérieur, faites clic-droit et `New representation→Class Diagram`. Cela crée une représentation graphique, mais ne crée pas pour autant de fichier `.aird` spécial.
3. En cliquant sur le package de votre méta-modèle, entrez les informations suivantes pour le nom, préfix et URI: `filesystem`, `filesystem` et `http://filesystem/1.0`. Vous pouvez rentrer n'importe quel nom et préfix, mais il est toujours mieux d'avoir les mêmes et d'avoir le nom apparaissant dans l'URI (qui, de préférence doit avoir la tête d'une URL).
4. Modélisez votre méta-modèle de système de fichier.
5. Modélisez ensuite quelques modèles instances de ce méta-modèle.

Modélisation du méta-modèle de base de données

Vous suivrez les mêmes étapes que pour le méta-modèle précédent, créez le dans votre projet `metamodels`.