Analyse dynamique avec Spoon

Simon Allier(simon.allier@inria.fr)

October 20, 2014

1 Spoon

Spoon[1] Spoon vous permet de transformer et d'analyser le code source Java. Spoon construit l'ast du programme cible. Cet ast peut être modifier (modification, ajout ou suppression de noeud) puis réécrit en code java.

Dans ce tp, nous allons utiliser Spoon afin d'analyser dynamiquement un programme (analyse de l'exécution d'un programme) en créant des `Processor` qui vont instrumentaliser le code du programme. Pour cela, vous devez sous-classer la classe abstraite `AbstractProcessor<E extends CtElement>` et implémenter la méthode `process(E élément)` qui sera appelé sur chaque élément de type `E` de l'ast.

API de Spoon: http://spoon.gforge.inria.fr/mvnsites/spoon-core/apidocs/.

1.1 Exemple

Le Processor `LogProcessor` remplace tout les `System.out.println(String)` par un loggeur qui redirige toutes les sorties dans la console vers un fichier log. utilisation du processor `LogProcessor` sur le projet example: `java -cp .:target/tpSpoon-1.0-SNAPSHOT-jar-with-dependencies.jar vv.spoon.MainExample src/main/resources/example src/main/java out`

Le code de la classe `A` après l'instrumentalisation:

```
package example;

public class A  {

    public static void main(String[] args) {

/**         System.out.println("A.main(String[] args)");
**/
    vv.spoon.logger.LogWriter.out("A.main(String[] args)",false);

        A a = new A();
        a.mth1(Integer.parseInt(args[0]));
```

[1] http://spoon.gforge.inria.fr/

```
    }

    public void mth1(int count) {
/**        System.out.println("A.mth1(int count)");
**/
    vv.spoon.logger.LogWriter.out("A.mth1(int count)",false);

        B b = new B();
        for(int i = 0; i < count; i++) {
            try {
                b.mth1(i);
                b.mth2();
            } catch (Exception e) {
/**                 System.err.println("error in A.mth1(int count)");
**/
    vv.spoon.logger.LogWriter.out("error in A.mth1(int count)",true);
            }
        }

    }
}
}
```

Le fichier log obtenue apres avoir execute le projet example (`java -cp .:target/example-1.0-SNAPSHOT.jar example.A 2`):

```
INFO: A.main(String[] args)
INFO: A.mth1(int count)
INFO: B.mth1(int i)
INFO: C.C(int i)
INFO: C.mth1()
ERROR: error in A.mth1(int count)
INFO: B.mth1(int i)
INFO: C.C(int i)
INFO: C.mth1()
INFO: result = 100
INFO: B.mth2()
```

1.2 Question 1

Utiliser Spoon afin de compter tous les appels de méthode d'un programme donné.
Pour le projet example, le résultat doit être semblable à ceci:

```
A.main(String[] args): 1
A.mth1(int count): 1
B.mth1(int i): 5
C.C(int i): 5
C.mth1(): 5
B.mth2(): 1
```

1.3 Question 2

Utiliser Spoon afin de construire l'arbre d'appel des méthodes d'un programme donné. Pour le projet example, le résultat doit être semblable à ceci:

```
A.main(String[] args)
 |  A.mth1(int count)
 |  | B.mth1(int i)
 |  |  | C.C(int i)
 |  |  | C.mth1()
 |  | B.mth1(int i)
 |  |  | C.C(int i)
 |  |  | C.mth1()
 |  | B.mth2()
```