KATEDRA INFORMATIKY A VÝPOČETNÍ TECHNIKY

Michal Dékány

Diplomová práce

Inženýrská informatika
Softwarové inženýrství
2015/2016

Vedoucí práce:
Ing. Richard Lipka, Ph.D.

FAKULTA APLIKOVANÝCH VĚD ZÁPADOČESKÉ UNIVERZITY V PLZNI

Generování testovacích dat z anotací

Motivace

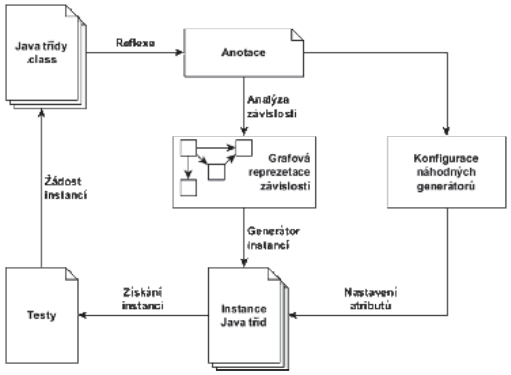
Některé druhy testů, především funkcionální a zátěžové testy, vyžadují velké množství realistických testovacích dat. Jejich příprava může být náročná a může se stát velmi snadno únavnou, obzvlášť když doménový model obsahuje příliš mnoho souvisejících tříd. Navíc s každou úpravou těchto tříd je obvykle spojena nutná úprava testovacích dat a kódu pro jejich přípravu. Tím je narušena možná automatizace tohoto procesu.

Cíl práce

Hlavním cílem je vytvoření knihovny v *Javě*, která bude využívat náhodné generátory a umožní snadné generování realistických testovacích dat. Náhodné hodnoty budou generovány podle pravidel, které je možné přidat do definic tříd v podobě anotací.

Použití knihovny

V rámci testů je možné využít knihovnu k získání instancí tříd. Ta nejprve zanalyzuje zadané třídy a podle použitých anotací generuje náhodná data pro jednotlivé atributy a vytvoří či nalezne instance pro atributy obsahující reference na závislé třídy.



Možnosti generování

Knihovna umožňuje generování instancí pro libovolné třídy, které nemusí splňovat pravidla pro *JavaBeans* nebo *POJO*, což znamená, že jednotlivé atributy nemusí mít veřejné *gettery* či *settery*. Každá třída může také mít libovolný počet konstruktorů či může využívat statickou tovární metodou pro vytvoření její instance.

Pro řízení generování objektů a hodnot jejich atributů lze využít různé anotace, které slouží:

- k určení způsobu přístupu k atributům objektů (přístup přes *gettery* a *settery* nebo přímý přístup s využitím reflexe);
- k určení konstruktoru či statické tovární metody pro vytvoření nové instance objektu;
- k určení strategie generování atributu:
  - ignorování atributu,
  - vynulování hodnoty atributu (nastavení výchozí hodnoty),
  - vytvoření nové instance pro atribut referující na závislou třídu (nesmí vzniknout kruhová závislost mezi objekty),
  - vyhledání již vytvořené instance pro atribut referující na závislou třídu (může vzniknout kruhová závislost mezi objekty);
- pro výběr třídy, která bude použita pro vytvoření nové instance;
- k určení způsobu vytvoření nové instance;
- k určení kritérií pro hledání vytvořených instancí;
- pro vygenerování náhodné hodnoty primitivního či komplexního atribut;
- pro určení způsobu uložení vygenerované hodnoty do atributu.

Postup pro vygenerování objektů

1. Jednotlivé atributy anotujeme anotacemi pro řízení generování objektů a hodnot jejich atributů.
2. Získáme instanci *populátoru objektů*, nad kterým zavoláme metodu pro generování objektů.
3. Získanou instanci (popř. seznam instancí) můžeme libovolně využít v rámci testů.

```
@Access(AccessType.FIELD)                                          1
public class Box {
  @CustomValueGenerator(RandomSizeGenerator.class)
  @SizeParameters(maxWidth = 100, maxHeight = 50)
  private Size size;
  @RandomColor
  private Color color;
  @NewInstance
  private Item item;
}
ObjectPopulator populator =                                        2
    ObjectPopulatorProvider.getObjectPopulator();
Box box = populator.populate(Box.class);
box= Box (id=20)                                                   3
  color= Color (id=24)
  item= Item (id=29)
  size= Size (id=31)
```

Závěr

V rámci této práce byla navržena a úspěšně implementována knihovna, která je schopna na základě anotací generovat náhodná data pro primitivní i komplexní atributy libovolného *Java* objektu a případně i vytvořit graf objektů, které jsou navzájem propojeny referencemi. Je snadno rozšiřitelná a již nyní je knihovna schopna konkurovat existujícím řešením, která jsou vyvíjena delší dobu a na jejich vývoji se podílí více vývojářů. Může být použita pro generování náhodných dat za účelem testování softwaru nebo naplnění databází. Další možností jejího využití může být generování ukázkových dat pro vyvíjené aplikace.