GenModelAddon / Modeling Symposium 2019

genModelAddon

Modeling Symposium ECE 2019
**GenModelAddon**
A tooling for an extended EMF code generation.

**OPCoach**

![OPCoach Image](http://www.opcoach.com/en)

Olivier PROUVOST
Eclipse Expert

15 bis rue des Troubadours - 31270 Cugnaux (France)
+33 106 28 07 65 64
olivier.prouvost@opcoach.com
@OPCoach_Eclipse
www.opcoach.com

**Image 1**

- Training/Consulting: RCP, E4, Modeling, Git in French, English and Spanish
- [http://www.opcoach.com/](http://www.opcoach.com/)
- @OPCoach_Eclipse
- olivier@opcoach.com

**MDD best practices**

- Separate the generated code and the overridden code
- Do not commit the generated code
- Integrate the generation process into the build phase
- Override the generated classes using inheritance
- Generate a clean code: even if it is generated, it must be human readable
- Test generation with a default model


**GMA**

GMA will apply this best practices:

- It splits the generated code into src-gen and src
- It manages inheritance, subpackages and referenced models
- It works with generics

---

➢ It generates an emf_factory_override extension
➢ In src-gen: generates using the current EMF generators
➢ In src: produces the classes extending the EMF generation

GMA is hosted here: http://opcoach.github.io/genModelAddon/3

---

3 - http://opcoach.github.io/genModelAddon/
How to install it?

Look for `genModelAddon` on the market place:
How to use it?

In your genmodel editor, right click on the root node:

And then fill the dialog with relevant values:

This value will be used to generate the dev source structure. (0) is the name of the EClass. A good idea here is to keep the default names when the ‘M’ prefix has been added for the generated classes. Example: (0)Impl for the EClass ‘Car’ will generate the ‘ICarImpl’ class extending the MCarImpl class and implementing the MCar interface.
In your code you can directly use your developer code:

```java
// MProjectFactory is the generated factory. It was used like this:
MProjectFactory mFactory = MProjectFactory.eINSTANCE;
MTask mt = mFactory.createTask();

// Now the ProjectFactory extends the MProjectFactory and
// creates instances of the developer objects. You can get it directly like this:
ProjectFactory factory = ProjectFactory.eINSTANCE;
Task t = factory.createTask();
```

---

**EMF Simple generation**

Simple EClass

```
ECore
```

```
A
```

```
EMF
```

```
src-gen
```

```
A
```

```
Almpl
```

---

**With inheritance**

```
ECore
```

```
A
```

```
B
```

```
EMF
```

```
src-gen
```

```
A
```

```
Almpl
```

```
src-gen
```

```
B
```

```
BImpl
```

---

**GMA Generation**

Simple EClass
**GMA technical implementation**

- Fully written in Eclipse 4 (injection and model fragments)
- Use the current EMF code generators
  - Override the GenModel factory to compute expected names
- Generators are written in XTend
- Fully tested on several advanced use cases (create a workspace and generate)
- Totally rewritten for version 2.X

**GMA metrics**
- Project started in December 2014
- 78 issues (72 closed)
- 103 unit tests
- Used by several companies (Sogeti, Airbus, Continental, Thales Alenia Space, Altran...)

**Q & A**
- Any Questions?
- After the conference:
  - olivier@opcoach.com