zühlke
empowering ideas

Create your own DSL with xtext

SCG Seminar @ UniBE, 23. November 2010

Frank Buchli

Frank Buchli

- MS in Computer Science, Uni Bern 2003
- Master Thesis: „Detecting Software Patterns using Formal Concept Analysis”
- 3 years Software Engineer
  Glue Software Engineering AG
- Since January 2007
  Software Architect
  Zühlke Engineering AG in Bern

Definition DSL

zühlke
empowering ideas

„A computer programming language of limited expressiveness focused on a particular domain.“

[Martin Fowler]

Key elements

- Computer programming language
  - human → machine
  - Enhance communication human ←→human
- Language nature / Sense of fluency
- Limited expressiveness
- Domain focus

Frank Buchli

Motivation

- **Motivation for a DSL**
  - Improving Development Productivity
  - Communication with Domain Experts

Frank Buchli

Goal

zühlke
empowering ideas

```
datatype String
datatype Bool


entity Session {
  title: String
  isTutorial : Bool
}

entity Conference {
  name : String
  attendees : Person*
  speakers : Speaker*
}

entity Person {
  name : String
}

entity Speaker extends Person {
  sessions : Session*
}
```


Frank Buchli

What is needed?

zühlke
empowering ideas

- Parser
  - yacc
  - antlr
- Lexer
- Editor
  - Syntax Highlighting
  - Outline
  - Code Completation

Frank Buchli


Xtext
Language Development Framework

How to start?

zühlke
empowering ideas

Describe the concrete syntax of your language

Will contain information about how the parser shall create a model during parsing.

Frank Buchli

Grammar: EBNF like

zühlke
empowering ideas

- Consits of rules like:

```
DomainModel :
  Entity*;
```

- Cardinality

(no operator)	exactly one
?	zero or one
*	zero or more
+	one or more

Frank Buchli

Grammar
Give elements names!

zühlke
empowering ideas

```
DomainModel :
   (elements+=Entity)*;
```

- Assignement operators

=	single feature
+=	list
?=	condition (boolean)

Frank Buchli

Grammar Keywords

zühlke
empowering ideas

- Introduce keywords!

```
Entity :
   'entity' name=ID '{'
      (features+=Feature)*
   '}';
```

- ID is predefined rule

Frank Buchli

Grammar References


```
Entity :
  'entity' name=ID ('extends' superType=[Entity])? '{'
    (features+=Feature)*
  '}';
```

- (…)? means optional

Grammar Inheritance

zühlke
empowering ideas

```
DomainModel :
   (elements+=Type)*;

Type:
   DataType | Entity;

DataType:
   'datatype' name=ID;
```


Frank Buchli

Grammar

zühlke
empowering ideas

```
Feature:
  name=ID ':' type=TypeRef;
```

- Sorry, no new concept …

Frank Buchli

Grammar
Optional keyword

zühlke
empowering ideas

```
TypeRef :
  referenced=[Type] (multi?='*')?;
```


Frank Buchli

Prerequisites

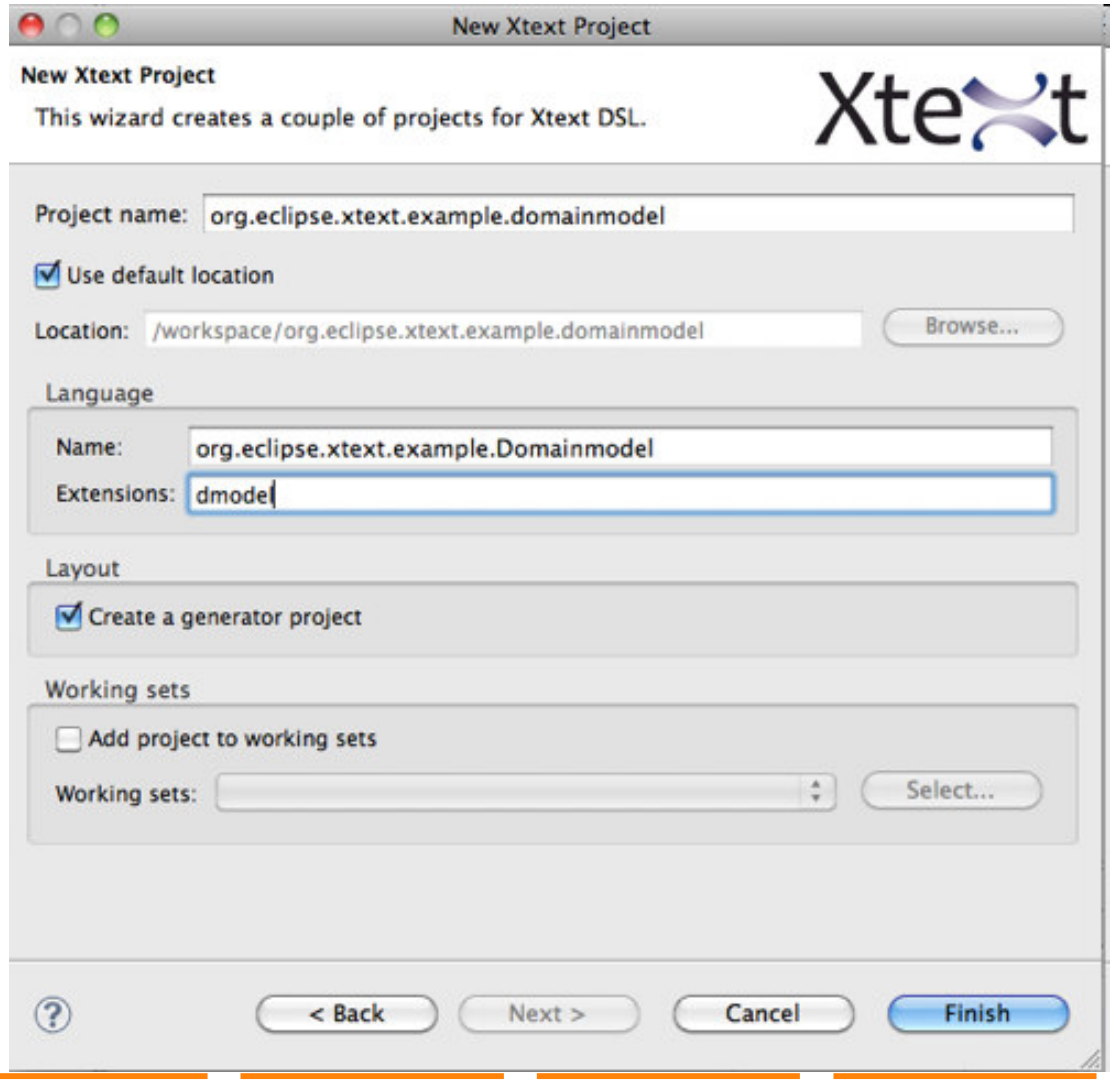
zühlke
empowering ideas

- Installed Xtext
  - Download Link:
    http://xtext.itemis.com/xtext/language=en/23947/downloads

Create a new project

zühlke
empowering ideas

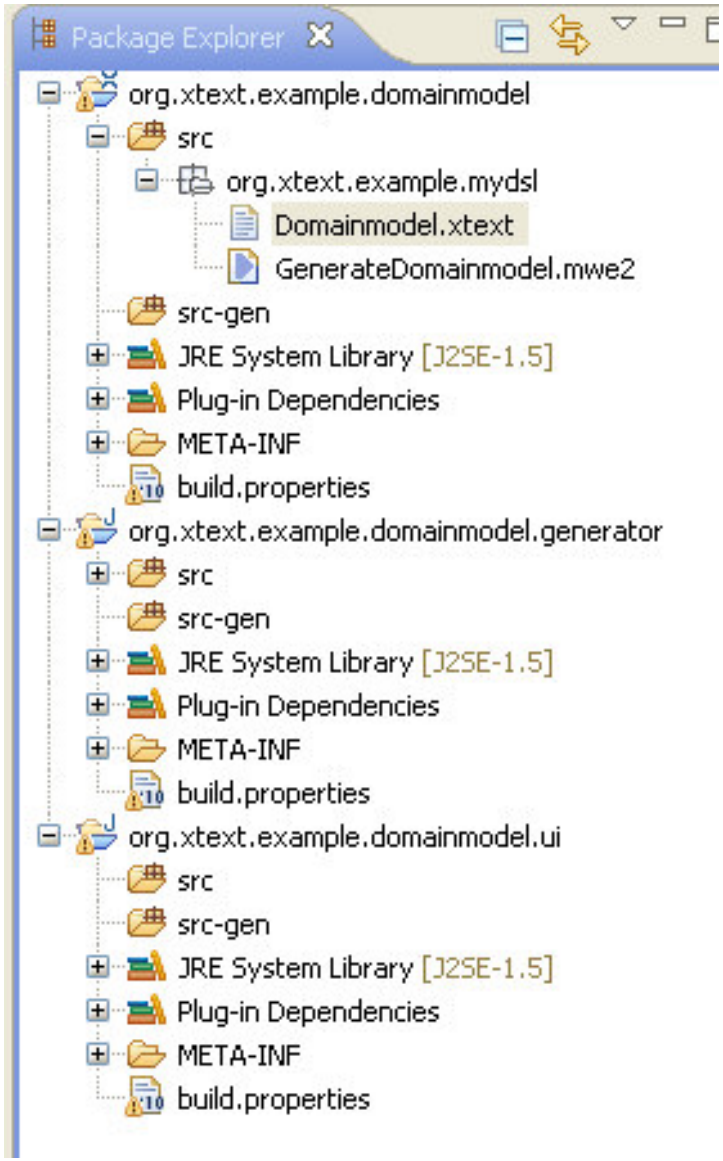
File → New → Project... → Xtext → Xtext project



Frank Buchli

Explain Project Layout

zühlke
empowering ideas



Frank Buchli

Build your own grammar

Meta information of the grammar:

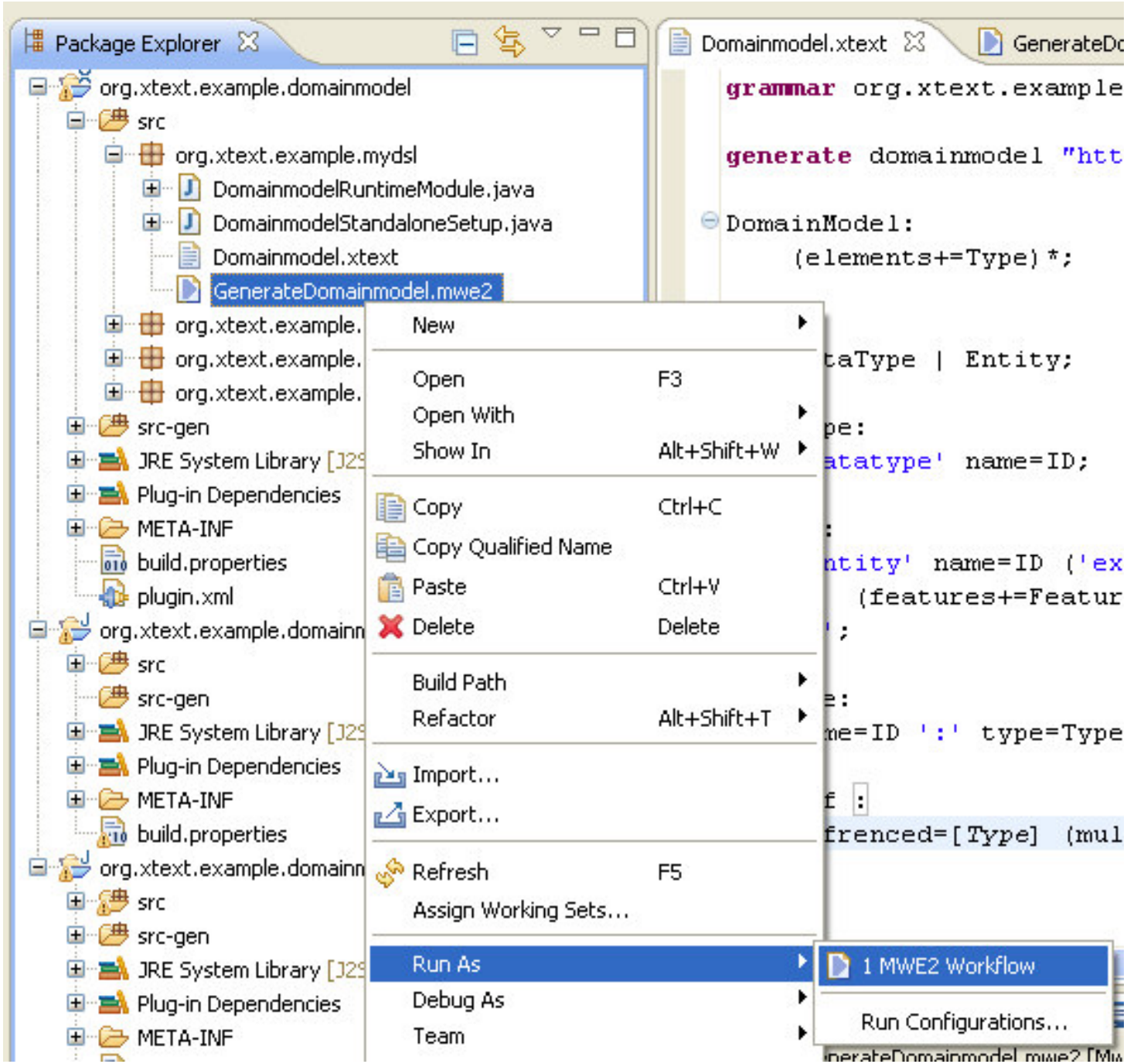
```
grammar org.xtext.example.mydsl.Domainmodel
  with org.eclipse.xtext.common.Terminals

generate domainmodel
"http://www.xtext.org/example/mydsl/Domainmodel"
```


Frank Buchli

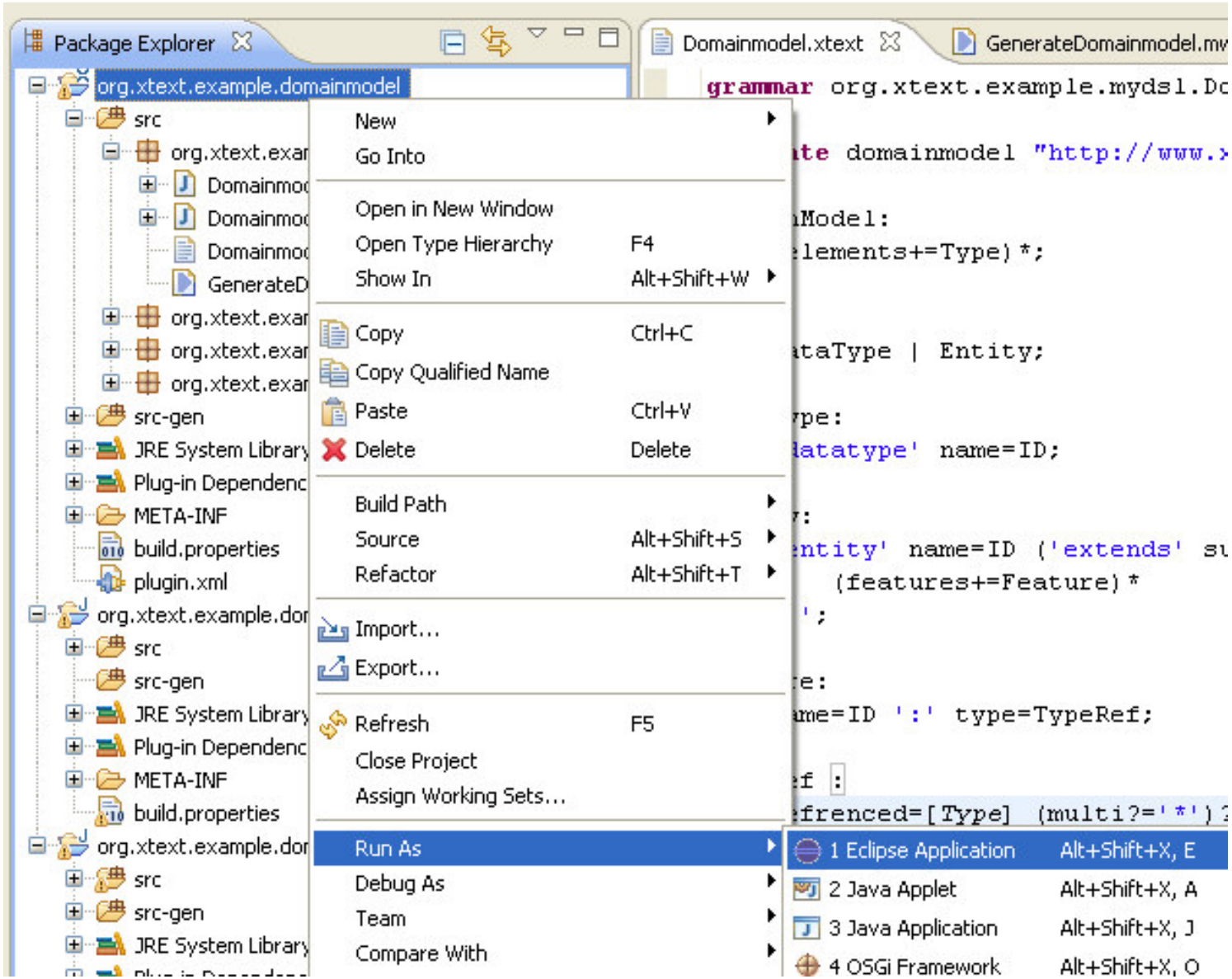
Generate Language artefacts

zühlke
empowering ideas



Frank Buchli

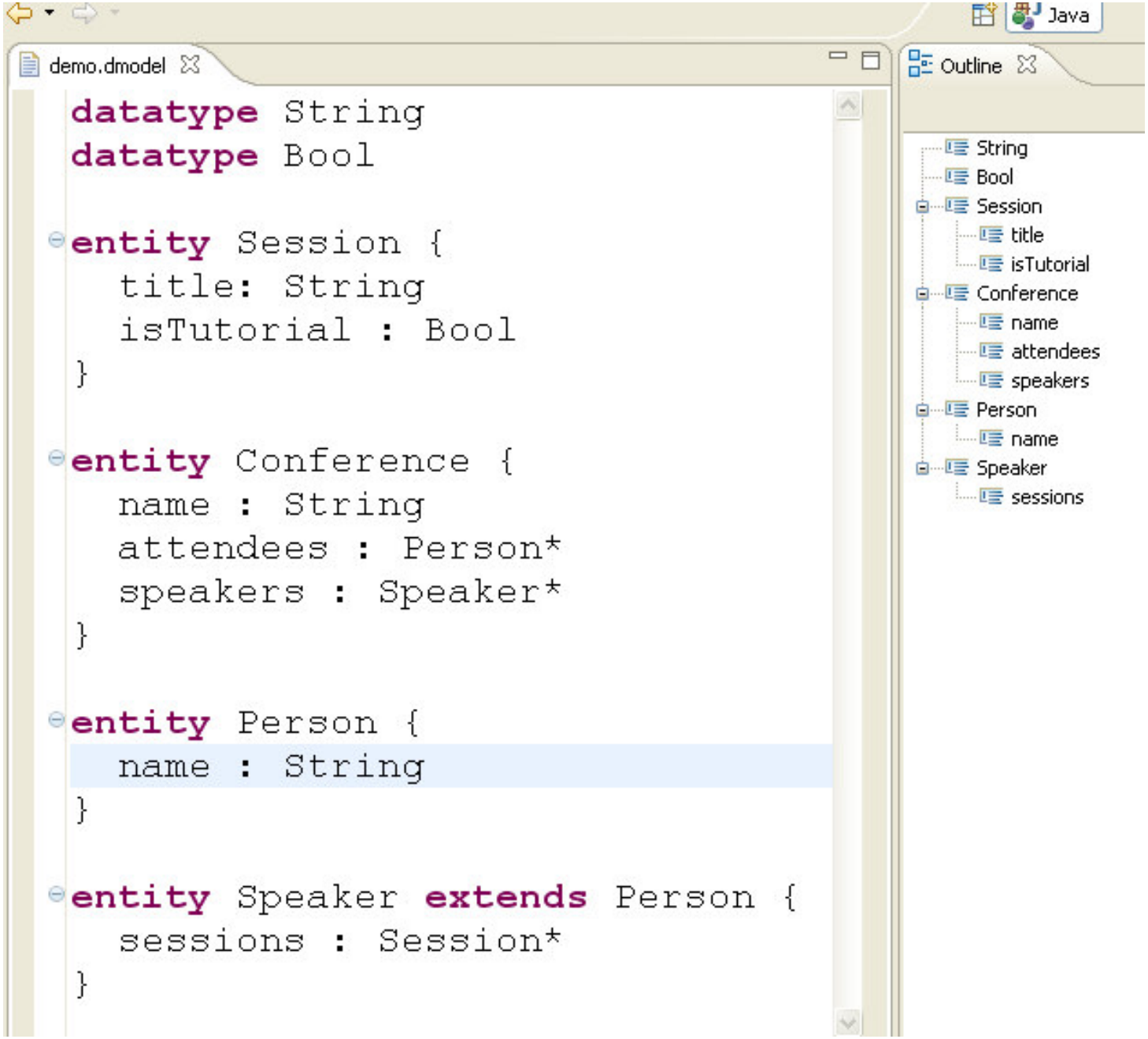
Run application



zühlke
empowering ideas

Frank Buchli

Write a file



zühlke
empowering ideas

Frank Buchli

What's next?

- Interpretation of the model
- Code generator

Interpretation

```
public static void main(String[] args) {
    init();

    DomainModel dm = readDomainModel("src/model/Example.dmodel");
    System.out.println("Number of elements: " + dm.getElements().size());

    for (Type type : dm.getElements()) {
        System.out.println(" Name: " + type.getName());
    }

    DomainModel aNewModel = DomainmodelFactory.eINSTANCE.createDomainModel();

}

private static DomainModel readDomainModel(String model) {
    ResourceSet rs = new ResourceSetImpl();
    Resource resource = rs.getResource(URI.createURI(model), true);
    EObject eobject = resource.getContents().get(0);
    System.out.println("Object Metaclass: " + eobject.eClass().getName());
    DomainModel dm = (DomainModel) eobject;
    return dm;
}

private static void init() {
    new DomainmodelStandaloneSetup().createInjectorAndDoEMFRegistration();
}
```

Generator
Use xpand template language

zühlke
empowering ideas

```
«IMPORT org::xtext::example::mydsl::domainmodel»

«EXTENSION templates::Extensions»

«DEFINE main FOR DomainModel»
    «EXPAND typeDef FOREACH elements»
«ENDDEFINE»

«DEFINE typeDef FOR Type»

«ENDDEFINE»

«DEFINE typeDef FOR Entity-»
«FILE name+".txt"-»
This is an example of a generated file.

The input element was "Hello «this.javaConformName()»!"

All greetings in the same file:
«ENDFILE-»
«ENDDEFINE»
```


Frank Buchli

Links

zühlke
empowering ideas

- http://www.xtext.org/
  - Tutorial:
    http://www.eclipse.org/Xtext/documentation/1_0_1/xtext.html
  - Interpretation:
    http://www.eclipse.org/Xtext/documentation/1_0_1/xtext.html#processing_Xtext_models
  - Generator:
    http://www.eclipse.org/Xtext/documentation/1_0_1/xtext.html#getting-started-xpand

- DSL Buch von Martin Fowler:
  http://martinfowler.com/dsl.html

- Zühlke:
  http://www.zuehlke.com

Frank Buchli