

MDSD in der Praxis

Ralph Winzinger, Senacor

Agenda 14.01.2015

- ⌚ Vorstellung R. Winzinger, Senacor & MDSD Projekte
- ⌚ DSLs textuell & grafisch
- ⌚ Übungsblock 1: getting started with Xtext grammars
- ⌚ Hello World
- ⌚ single grammar, multiple grammars
- ⌚ Q&A, Diskussion

Agenda 21.01.2015

- Übungsblock 2: some more Xtext and Xtend
- Editor: Validierungen, Hilfestellungen, Formatierung, ...
- Xtend: Sprachkonstrukte, (Code-)Generierung
- Best Practices
- Q&A, Diskussion

Ralph Winzinger & Senacor

RALPH WINZINGER
PRINCIPAL ARCHITECT

 SENACOR

Beratungsschwerpunkte

- Enterprise Architektur
- Legacy-Integration/-Migration
- Software-Engineering
- JEE, SOA, Webservices, SAP Integration, OSGi, Frontendtechnologien, Mobile



Branchenfokus

- Banking

Ausbildung und berufliche Erfahrung

- Über 10 Jahre Erfahrung in der Anwendungsentwicklung, technischen Projektleitung und Architekturberatung im Bankenumfeld
- Architekt, Senacor Technologies AG (seit 2000)
- Freiberufliche Tätigkeit im Bereich Mobile Computing (1996 – 2002)
- Diplom in Informatik, Univ. Erlangen-Nürnberg

Senacor Technologies AG

DA Siegelabendkoforsen

Ralph Winzinger & sonst



Senacor & MDSD

Warum DSLs?

Wer hat schon mal mit  DSLs gearbeitet?

Wer wusste vorher, dass  mit einer DSL arbeitet?

DSLs sind vielleicht nicht überall,
aber durchaus weit verbreitet.

Warum DSLs?

```
internalOnly (true|false) "false">

<!ELEMENT characteristic EMPTY>
<!ATTLIST characteristic
  name CDATA #REQUIRED
  value CDATA #REQUIRED>
]>

<processmodel>
  <domain name="Baufi" id="1">

    <module name="Vorgang" id="1">
      <state name="n.a." id="1">
        <transition name="externBearbeiten" state="extern in Bearbe...
        <transition name="bearbeiten" state="inBearbeitung"/>
        <transition name="internerVKBearbeiten" state="interner VK ...
      </state>

      <state name="extern in Bearbeitung" maps="vertrieb-inBearbeitu...
        <transition name="uebergeben" state="uebergeben"/>
        <transition name="interner VK in Bearbeitung" state="interner ...
      </state>

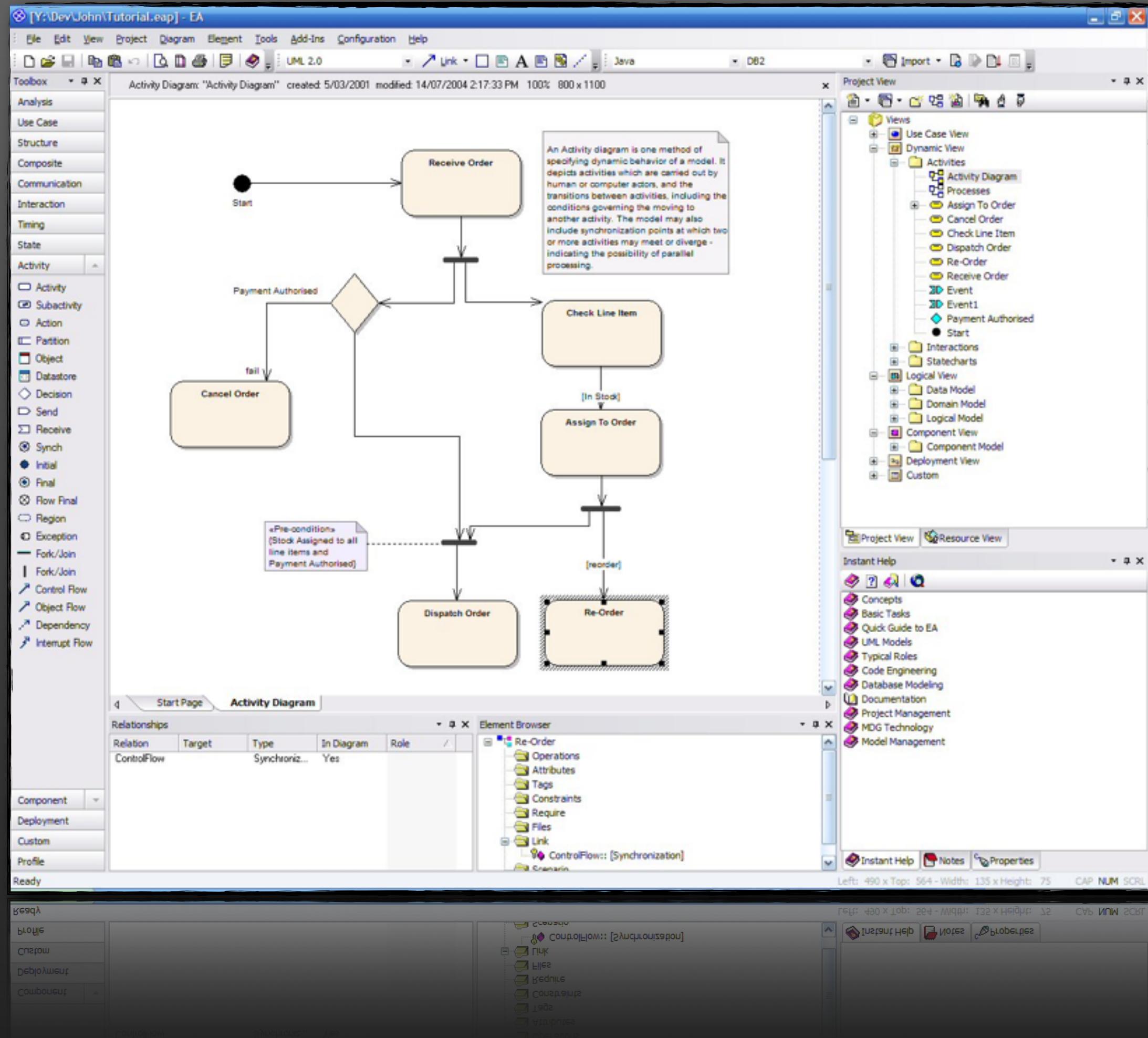
      <state name="uebergeben" id="3">
        <dependency module="PreScoring" state="vollstaendig"/>
        <transition name="bearbeiten" state="inBearbeitung"/>
      </state>

      <state name="frei" id="4">
        <transition name="bearbeiten" state="inBearbeitung"/>
        <transition name="zurueckstellen" state="zurueckgestellt"/>
      </state>

    </state>
    <!-->
    <!-->
    <!-->
  </processmodel>
```

- ⌚ Abstraktion von eingesetzter Technologie
- ⌚ Annäherung an fachliches Vokabular, gemeinsame Diskussionsbasis

Warum DSLs?



ocular Dokumentation

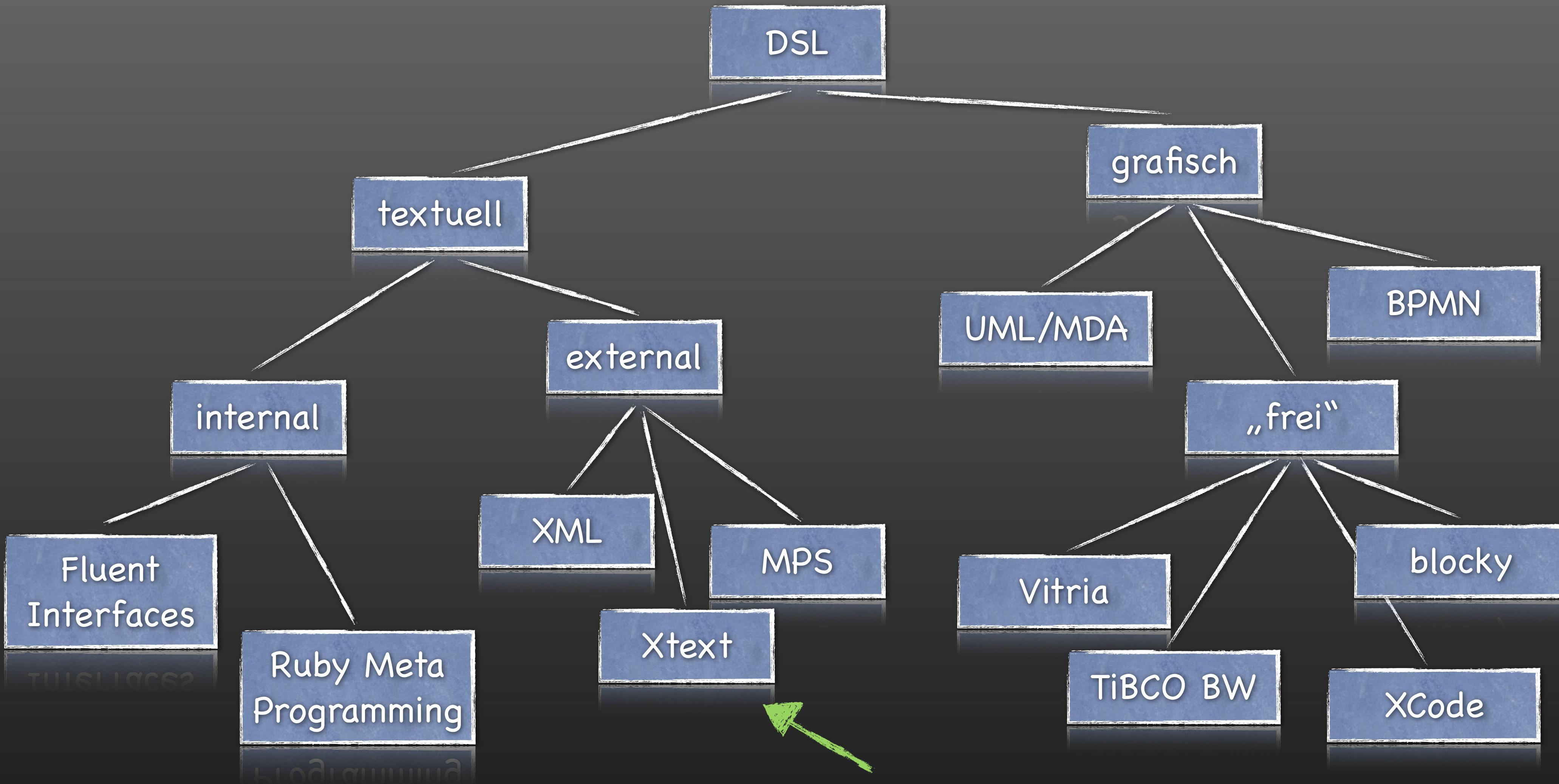
ocular Definition von
Sachverhalten mit
komplexen
Strukturen

Warum DSLs?



Dont
Repeat
Yourself

DSLs haben viele Gesichter



Welche DSL
und weshalb nun eigentlich?

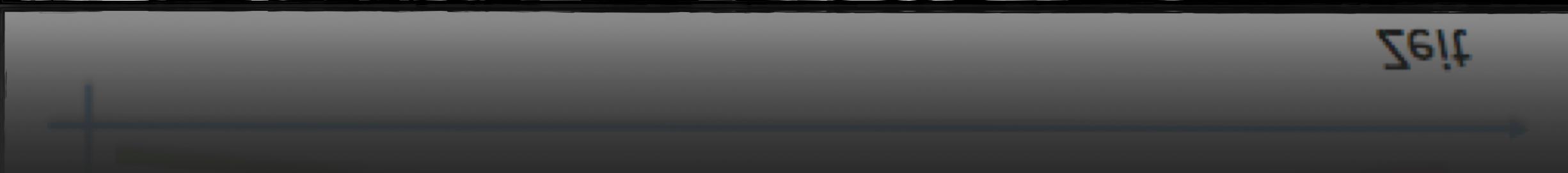
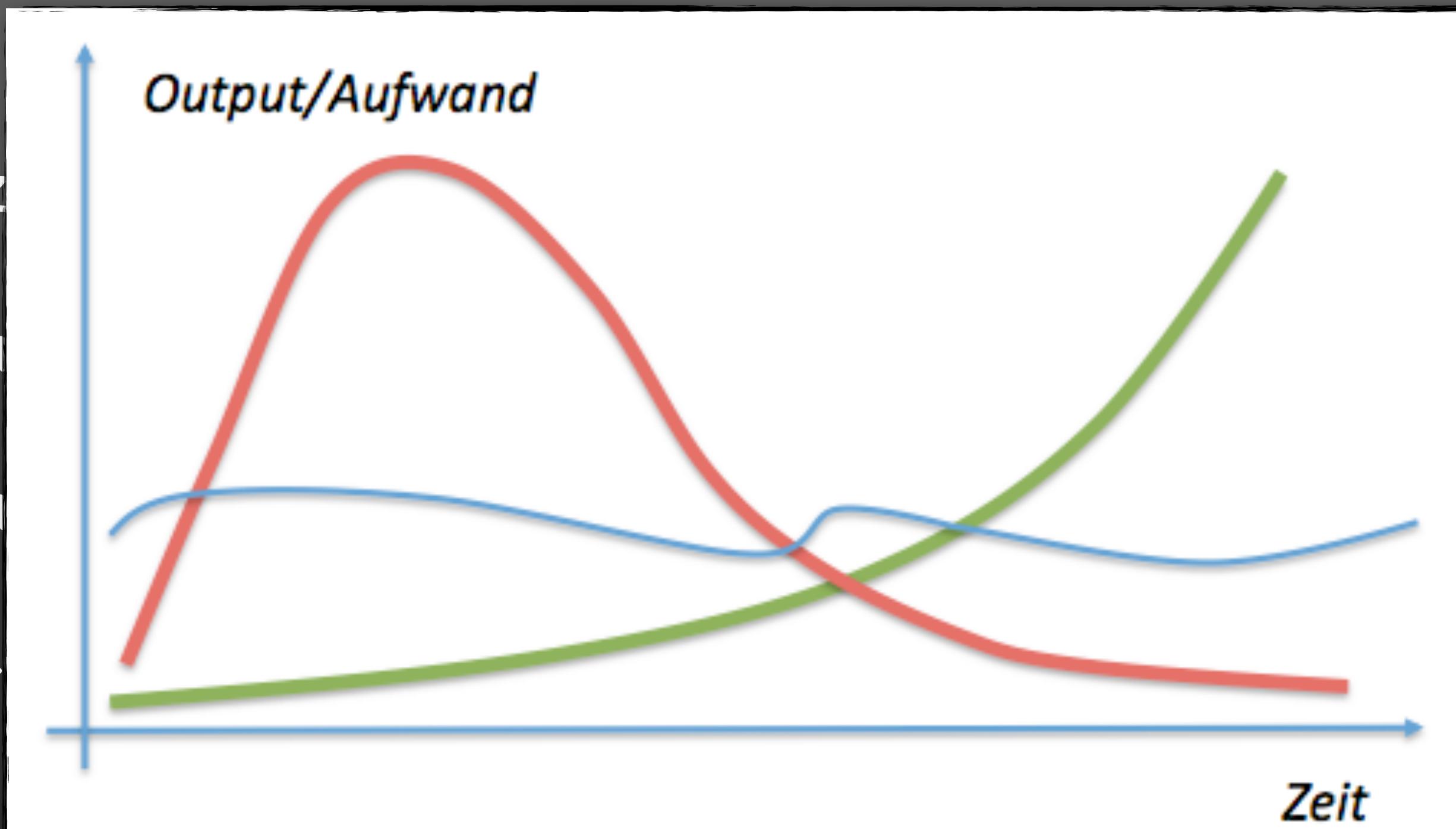
~~do obey „the law of the instrument“!~~

„do we have a problem here?“

- @@ Lern- bzw. Effizienzkurve
- @@ Problem von hinreichender Größe
- @@ Problem von hinreichender Dauer
- @@ Spricht Domain-/Projekt-/Sprachgröße für GPL oder DSL?

„do we have a problem here?“

- Lern- bzw. Effizienzproblem von hinten
- Problem von hinten
- Spricht Domain-...



Grafisch oder Textuell?

- ⌚ Grafische DSL abstrahiert und vereinfacht - kann sie dem Problem gerecht werden?
- ⌚ Können komplexe Sachverhalte überhaupt sinnvoll dargestellt werden?
- ⌚ Können Lösungen effizient beschrieben werden?

Grafisch oder Textuell?

The image displays three distinct graphical interfaces for programming:

- Logic Editor:** Shows waveforms for HCLK, HSEL, HADDR, HTRANS, HREADY, and HREADYOUT. A central workspace contains a green gear icon labeled 'CB' and an orange character icon labeled '4'.
- Scratch-like Script:** A 'Gated' script block is shown, containing a 'repeat while [true]' loop. Inside the loop, it performs a 'move [forward v] steps' action, followed by a conditional branch: 'if [not wall to the left v] then [turn left v]' (with a note 'Returns true if there is a wall in front'), 'else if [wall ahead v] and [not v] then [turn right v]', and 'else if [wall ahead v] then [turn right v] [turn right v]'. A 'Move' block at the bottom provides port and steering settings.
- BPMN Process Diagram:** A Business Process Model and Notation (BPMN) diagram. It features a central 'BusinessPartnerProcess' node connected to four parallel gateway nodes ('FromSapBpQ1', 'FromSapBpQ2', 'FromSapBpQ3', 'FromSapBpQ4'). These gateways converge into another 'BusinessPartnerProcess' node, which then leads to a 'BusinessPartnerStatusProcess' node. A 'CompletionCallback' connector links the final process back to the initial one.

Grafisch oder Textuell?

The screenshot displays the DSL-Editor interface with two main panes illustrating different ways to represent the same function.

Left Pane (Graphical Representation): Shows a section of the German Social Security Act (BGB) under § 102 Ergänzende Leistungen. It details the entitlement to winter supplement for part-time workers and seasonal workers. Below the text is a navigation bar with links like "zum Seitenanfang" and "Datenschutz".

Right Pane (Textual Representation): Shows the same function definition in a textual DSL language. The code defines a mask for editing calculation periods, specifying fields for start date, end date, and reason, along with conditional logic for employment type.

```
DSL-Editor v31.3.0 - BA-DSL-Perspektive - PRV_13.01.00.00 VERBIS/Funktionen/Unterfunktion/Personenverwaltung/Funktion_ANZb_AnrechnungszeitBearbeiten/Maske_ANZb_AnrechnungszeitBearbeiten.gui - DSL-Editor
```

```
maske Maske_ANZb_AnrechnungszeitBearbeiten
(
  funktion Funktion_ANZb_AnrechnungszeitBearbeiten
  titel "Anrechnungszeit bearbeiten - " & #Person.anzeigeName
  maskenmuster eingabe
  initialisieren %ANZb_AnrechnungszeitBearbeiten_Initialisieren
  beschreibung "Diese Maske zeigt Informationen einer Anrechnungszeit zur Bearbeitung an"
  praxishilfe "anrechnungszeitenauflisten"
)

abschnitt Abschnitt_Anrechnungszeit
(
  titel wenn #IstAnrechnungszeitAV = ja
  dann "Anrechnungszeit Arbeitsvermittlung"
  sonst "Anrechnungszeit Berufsberatung"
)

daten Startdatum #Anrechnungszeit.startdatum (
  beschriftung "Beginn MAZ-Zeitraum"
)

daten Endedatum #Anrechnungszeit.enddatum (
  beschriftung "Ende MAZ-Zeitraum"
)

daten Meldungsgrund #Anrechnungszeit.meldungsgrund (
  beschriftung "Meldungsgrund"
  vorbelegung #Anrechnungszeit.meldungsgrund
  inhalt wenn #IstAnrechnungszeitAV=ja dann $BLA_AV sonst $BLA_BB
  status wenn #IstAnrechnungszeitAV=ja dann aktiv sonst anzeigen
)
```

Tooling

- ⦿ Bei grafischen DSLs in der Regel vorgegeben, keine Anpassungsmöglichkeiten
- ⦿ Maturity
- ⦿ Benutzbarkeit (Zielgruppe!)
- ⦿ Integration in den Entwicklungsprozess
- ⦿ Refactoring

Collaboration & Versioning

- ⌚ Sind Modellierer Teamplayer oder Einzelkämpfer?
- ⌚ Diff & Merge, insbesondere bei grafischen DSLs
- ⌚ Wird Versionierung angemessen unterstützt?
- ⌚ Wird gewähltes Vorgehen unterstützt?

Modell- & Templatedefinition

- Top down, nicht auf der grünen Wiese
- mit dem Fachbereich DSL anhand tatsächlicher Problemstellungen erarbeiten
- Templates entwickeln, nachdem hinreichend viele Use-Cases prototypisch umgesetzt wurden und daraus generische Codeanteile extrahiert werden können

nicht zu kurz denken ...

- ☛ Nicht nur primäre Artefakte modellieren/generieren!
 - ☛ Tests
 - ☛ Spezifikation, Dokumentation
 - ☛ Querschnittliche Aspekte
 - ☛ ...
- ☛ Aber nur soweit sinnvoll!

<CODE>

Umgebung & Technologie



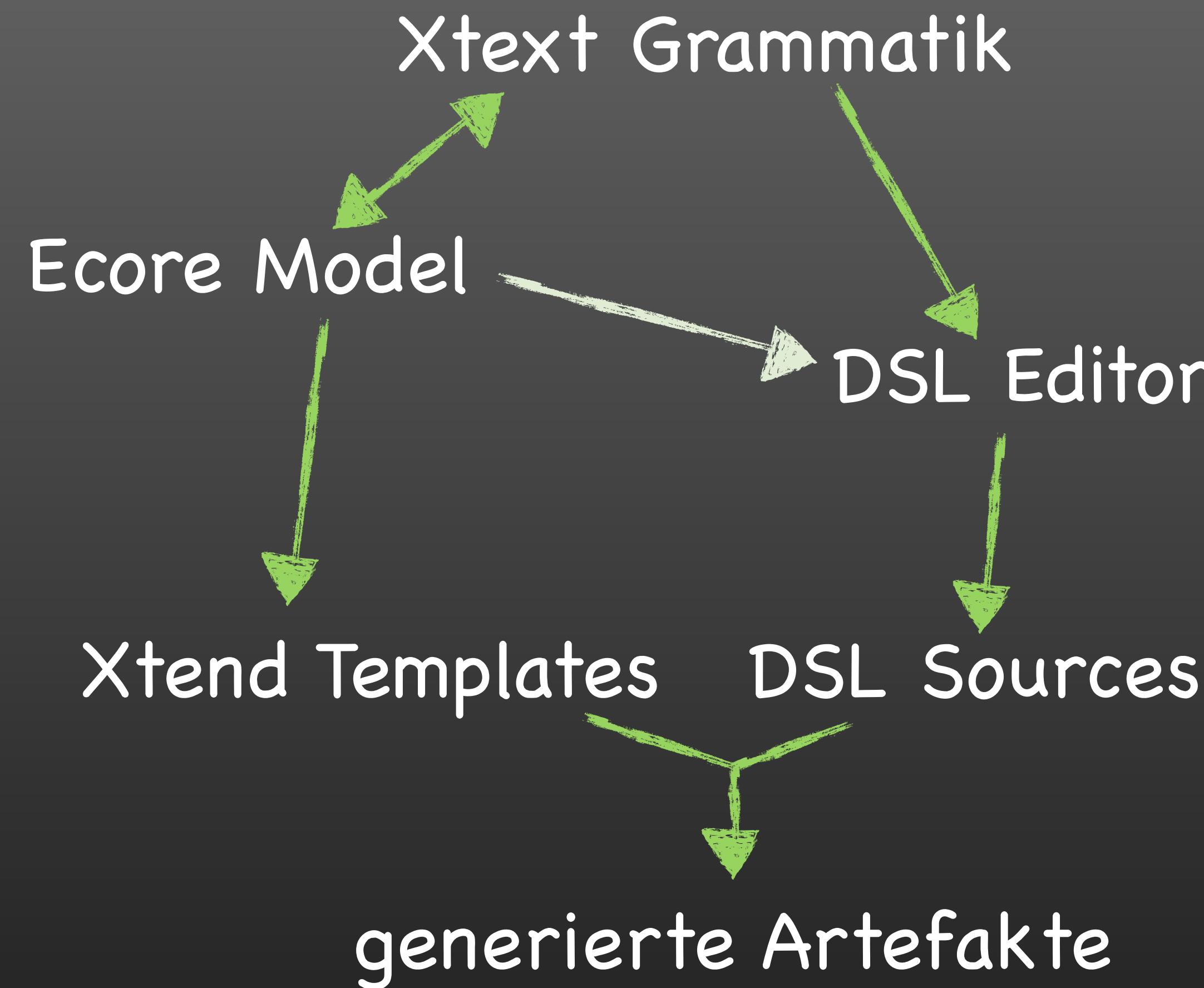
Umgebung & Technologie

- Xtext & Xtend
- Java basiert, Eclipse Projekte
- Definition textueller DSL Grammatiken & Templating Engine
- Automatisch generierter Editor für DSL Code
- enge Integration mit weiteren Eclipse Projekten

Use Cases

- ⌚ Hello World
- ⌚ Service/Logic/Entity

Xtext & Xtend Komponenten



Xtext Basics

Grammar:

```
'grammar' name=GrammarID
('with' usedGrammars+=[Grammar|GrammarID] (',' usedGrammars+=[Grammar|GrammarID])* )?
(definesHiddenTokens?='hidden' '(' (hiddenTokens+=[AbstractRule] (',' hiddenTokens
+=[AbstractRule])* )? ')')?
metamodelDeclarations+=AbstractMetamodelDeclaration*
(rules+=AbstractRule)+
;
```

GrammarID returns ecore::EString:

```
ID ('.' ID)*;
```

```
AbstractRule : ParserRule | TerminalRule | EnumRule;
```

```
AbstractMetamodelDeclaration :
GeneratedMetamodel | ReferencedMetamodel;
```

usw. ...

EBNF

Xtext Basics

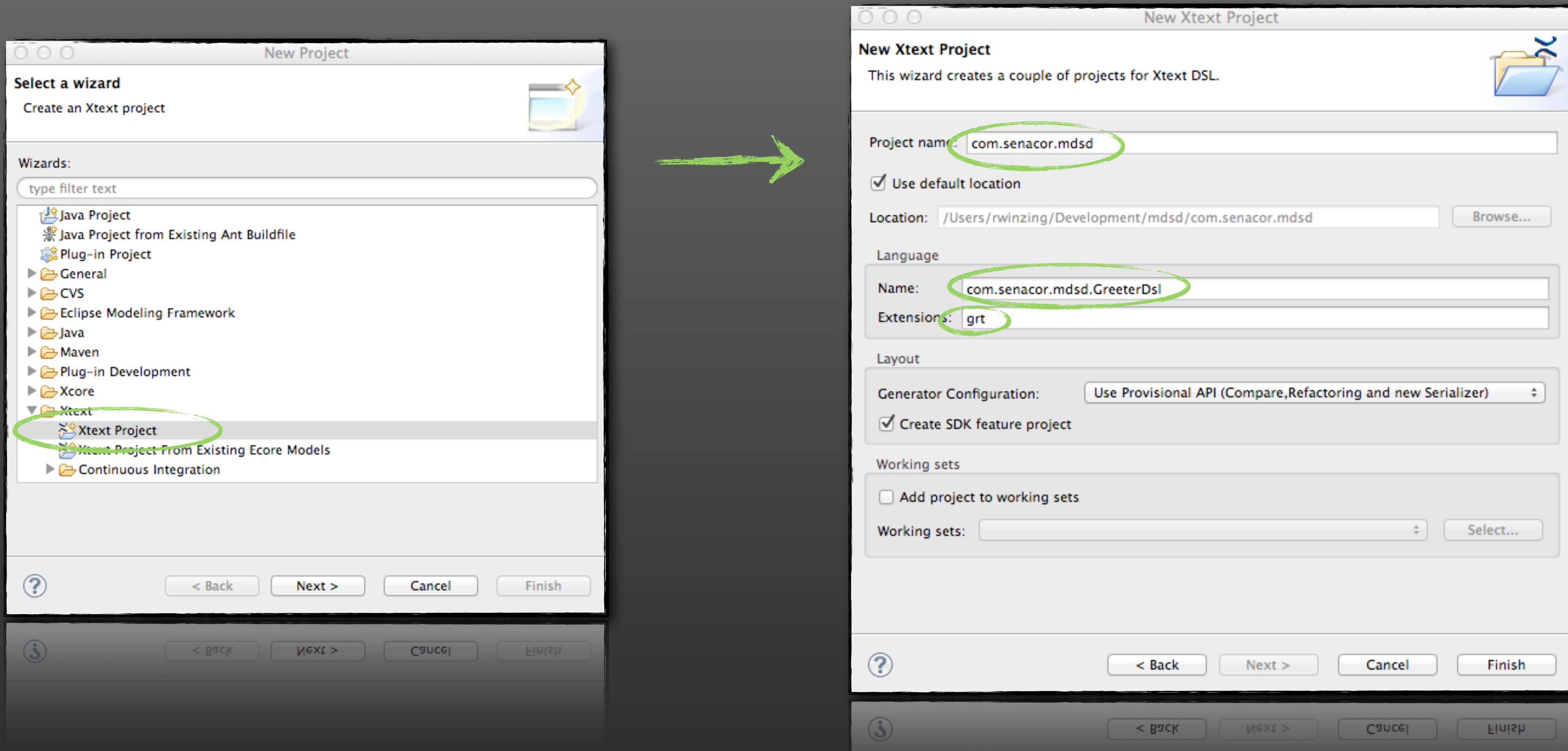
- ⌚ Rules
- ⌚ Terminal
- ⌚ DataType
- ⌚ Parser/Production/EObject
- ⌚ Keywords
- ⌚ Assignments
- ⌚ Referenzen

```
Model:  
greetings+=Greeting*;  
  
Greeting:  
'Hello' name=ID '!';
```

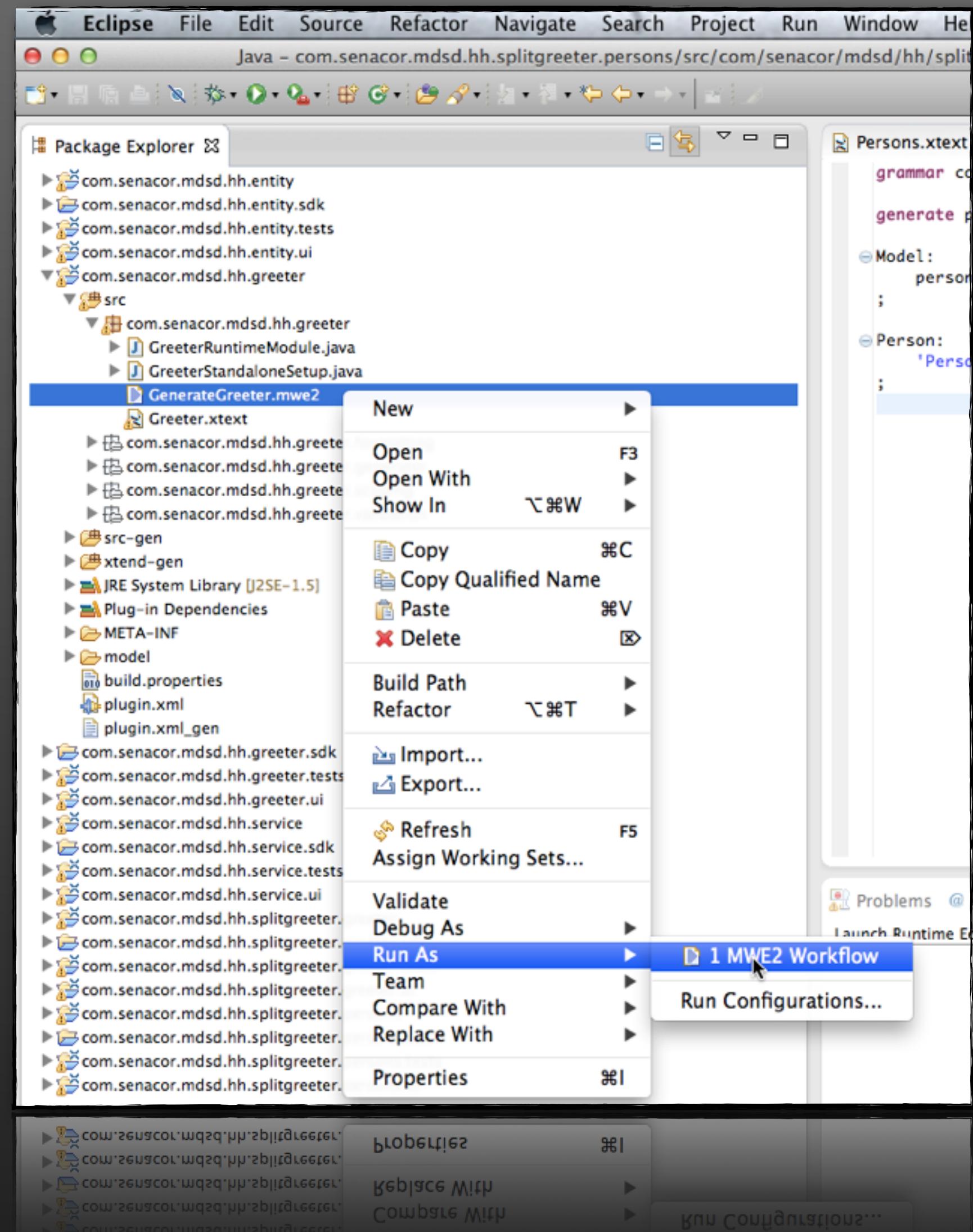
Hello, World! (1)

- ⌚ Projekt erzeugen
- ⌚ Editor testen
- ⌚ generierte Artefakte ansehen

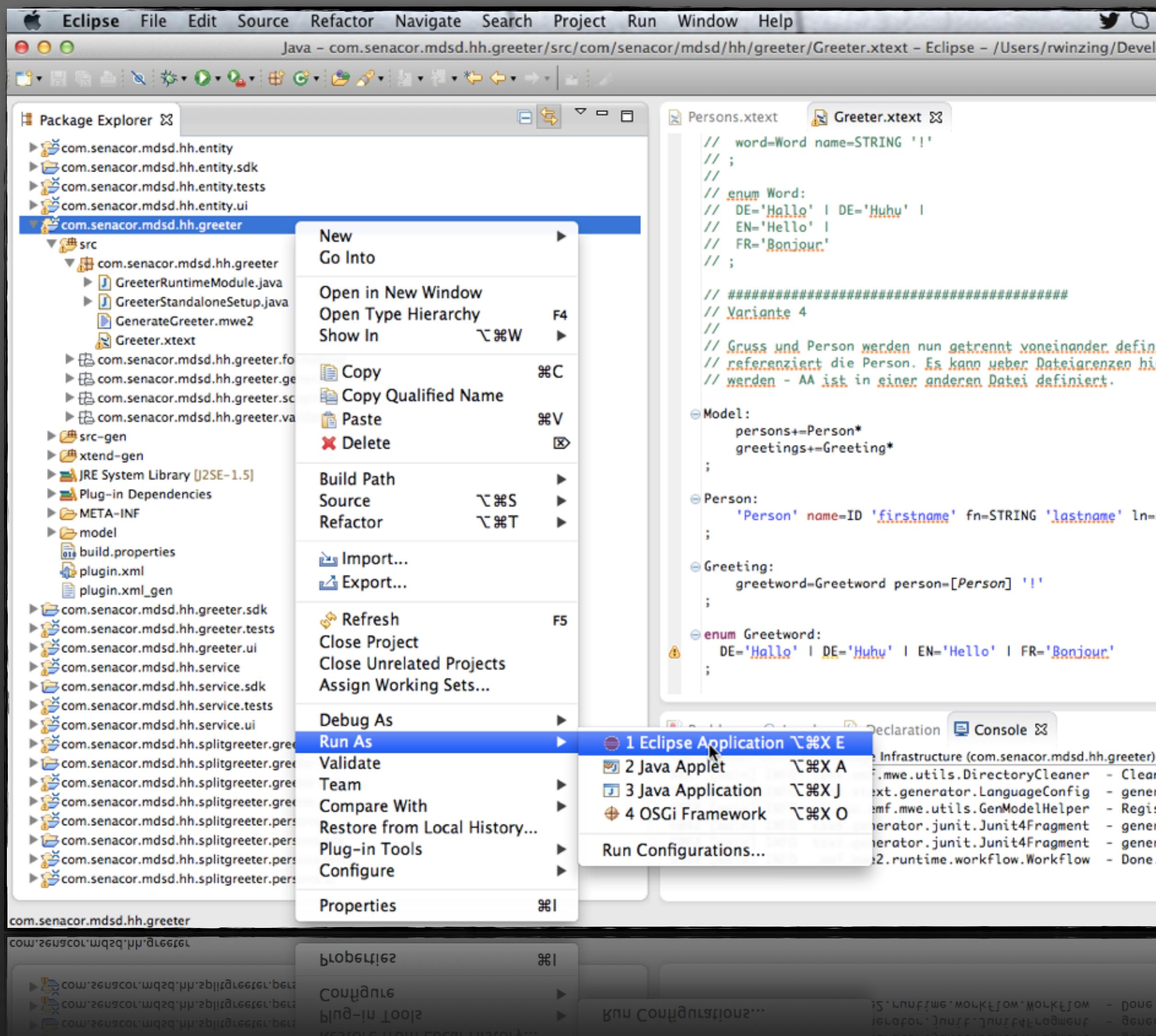
Ein erstes Xtext Projekt ...



... Artefakte generieren ...



... und Editor starten.



Hello, World! (2)

- etwas Struktur durch Klammern
- Name soll als String erfasst werden

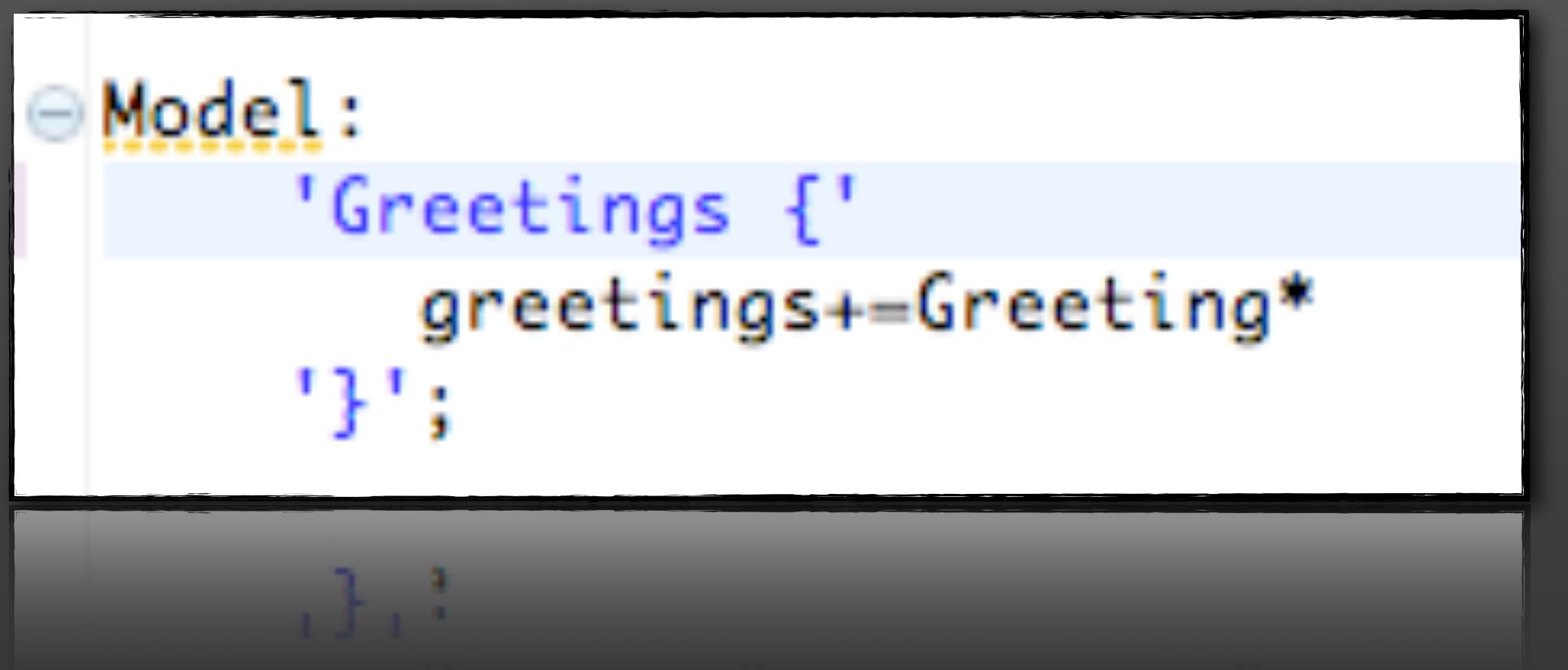


```
test.grt01 test.grt02
Greetings {
Hello "World"
}
```

```
test.grt01 test.grt02
Greetings {
Hello "World"
}
```

Hello, World! (2)

Wie wäre es mit



The screenshot shows a code editor window with a dark theme. A tooltip or callout box is overlaid on the screen, pointing to a section of the code. The code is a Java snippet within a class named 'Model':

```
Model:
    Greetings {
        greetings+=Greeting*
    };
```

The code uses inconsistent indentation: the opening brace '{' is aligned with the class name 'Model', while the closing brace '}' is aligned with the opening brace '{'. This inconsistency is highlighted by the tooltip.

?

SPACES ARE EVIL!

Hello, World! (3)

- ⌚ Alternative Grußtexte
- ⌚ Eine zusätzliche Rule



```
Greetings {
    Hello "World"!
    Hallo "Welt"
}
```

Hello, World! (4)

- ⌚ Grußworte als Enum
- ⌚ Kommentare



```
Greetings {
    Hello "World"!
    Hallo /* comment */ "Welt"!
}
```

Hello, World! (5)

- ☛ Personen & Grüße
- ☛ Cross-References



```
test.grt01 *test.grt02 ✎

Persons {
    rw firstname Ralph lastname Winzinger
}

Greetings {
    Hallo rw!
}

}
```

HOTTO UNI

Hello, World! (5)

```
grammar com.senacor.mdsd.greeter02.GreeterDsl02 with org.eclipse.xtext.common.Terminals

generate greeterDsl02 "http://www.senacor.com/mdsd/greeter02/GreeterDsl02"
```

Model:

```
'Persons' '{'
    persons+=Person*
}''
```

```
'Greetings' '{'
    greetings+=Greeting*
}'';
```

Person:

```
name=ID 'firstname' firstname=ID 'lastname' lastname=ID gender=('w' | 'm');
```

Greeting:

```
word=Greetword person=[Person] '!' ;
```

enum Greetword:

```
DE='Hallo' | EN='Hello' | FR='Bonjour';
```

```
DE='Hallo' | EN='Hello' | FR='Bonjour';
```

enum Greetword:

Hello, World! (6)

```
④ service.xtext ⑤ Greeter.xtext ⑥ com.senacor.mdsd.hh.splitgreeter.greeter ⑦ Persons.xtext
grammar com.senacor.mdsd.hh.splitgreeter.greeter with org.eclipse.xtext.common.Terminals

import "http://www.senacor.com/mdsd/hh/splitgreeter/persons/Persons" as Persons

generate greeter "http://www.senacor.com/mdsd/hh/splitgreeter/greeter/Greeter"

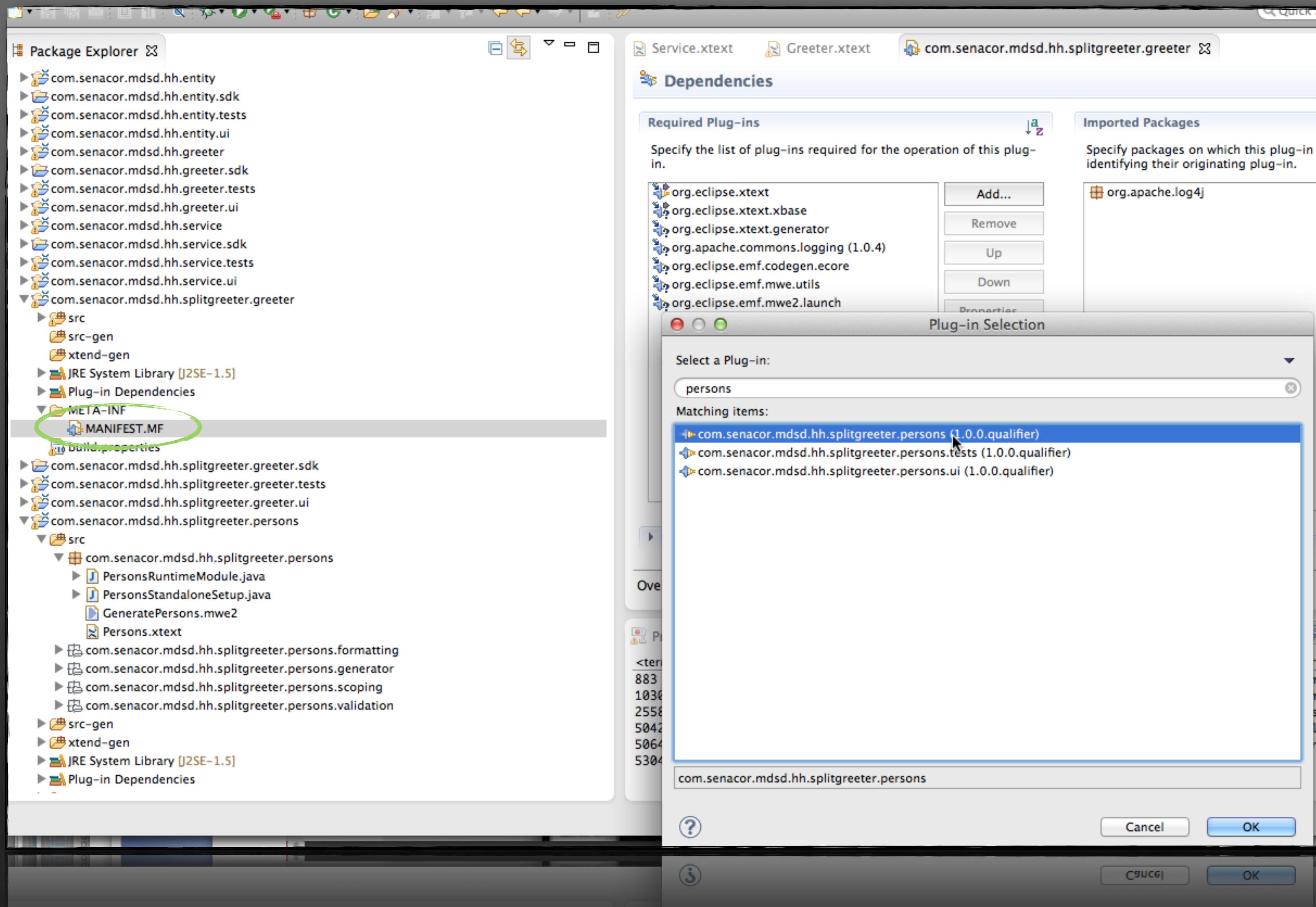
⑧ Model:
    greetings+=Greeting*
;

⑨ Greeting:
    greetword=Greetword person=[Persons::Person] !!
;

⑩ enum Greetword:
    DE='Hallo' | DE='Huhu' | EN='Hello' | FR='Bonjour'
;

?
    DE='Hello' | DE='Huhu' | EN='Hello' | FR='Bonjour',
    GREEK='Εύαλλος'
```

Hello, World! (6)



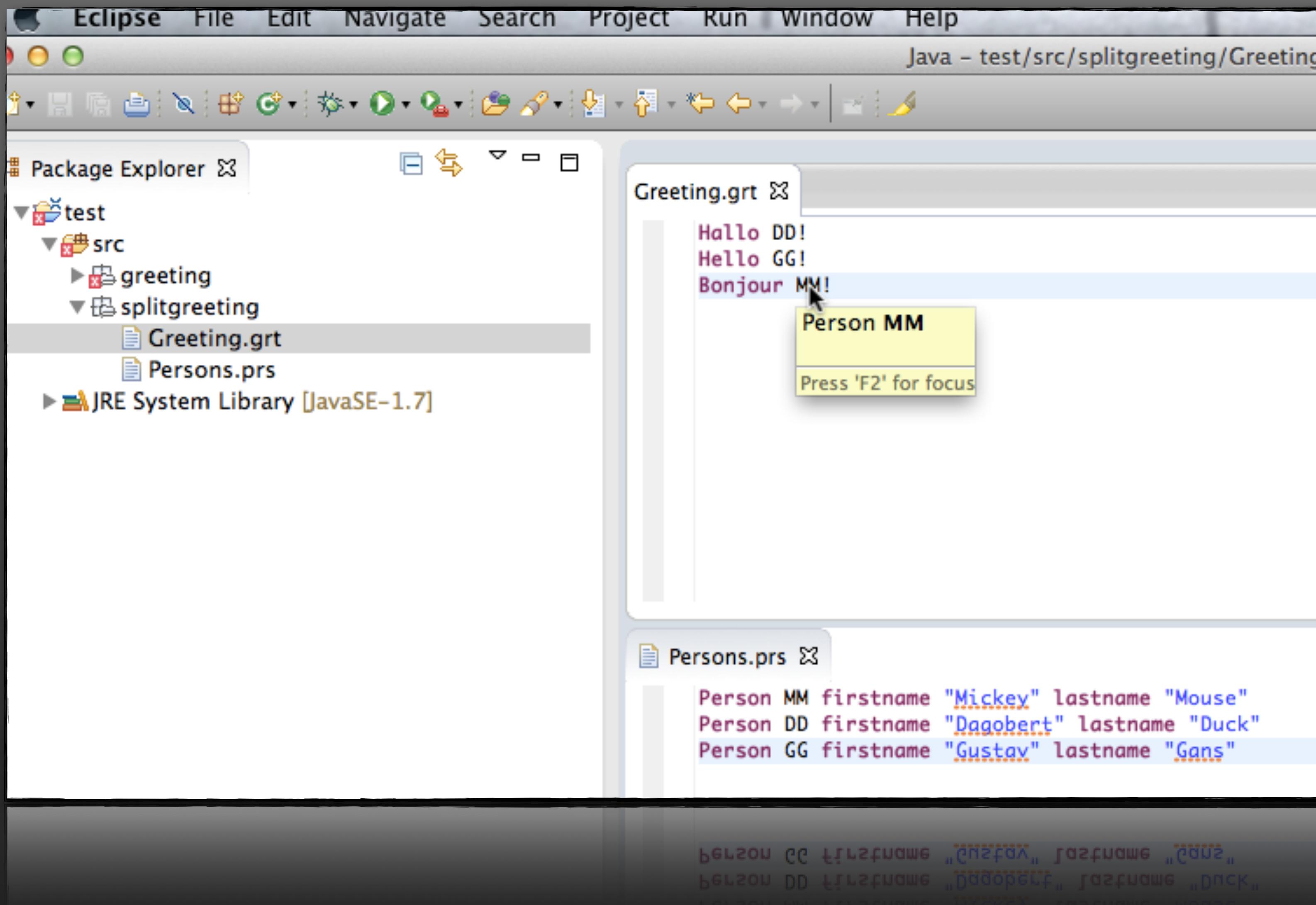
Hello, World! (6)

```
Workflow {
    bean = StandaloneSetup {
        scanClassPath = true
        platformUri = "${runtimeProject}..."
        // The following two lines can be removed, if Xbase is not used.
        // registerGeneratedEPackage = "org.eclipse.xtext.xbase.XbasePackage"
        // registerGenModelFile = "platform:/resource/org.eclipse.xtext.xbase/model/Xbase.genmodel"

        // register splitgreeter persons
        registerGeneratedEPackage = "com.senacor.mdsd.hh.splitgreeter.persons.persons.PersonsPackage"
        registerGenModelFile = "platform:/resource/com.senacor.mdsd.hh.splitgreeter.persons/model/generated/Persons.genmodel"
    }
}
```

Java code snippet showing configuration for a workflow setup. A green oval highlights the registration of the 'Persons' package and its genmodel file.

Hello, World! (6)



Agenda 21.01.2015

- ⌚ Recap
- ⌚ Übungsblock 2: some more Xtext and Xtend
 - ⌚ Xtend: Sprachkonstrukte
 - ⌚ (Editor: Validierungen, Hilfestellungen, Formatierung, ...)
 - ⌚ Xtend: (Code-)Generierung
 - ⌚ Build: Integration, Automatisierung & Best Practices
- ⌚ Q&A, Diskussion

Recap

Model:

```
'Greetings' '{'  
    greetings+=Greeting*  
'}';
```

Greeting:

```
('Hello' | 'Hallo') name=ID '!'  
;
```

:

```
('Hello' | 'Hallo') name=ID , i,
```

Recap

Model:

```
'Greetings' {'  
    greetings+=Greeting*  
'}';
```

Greeting:

```
('Hello'|'Hallo') name=ID '!';
```

2

Greetings {
Hello Ralph!
Hallo Hamburg!
}

```
public class GreetingImpl extends EClass
{
    protected String name = NAME_EDEFAULT;

    protected GreetingImpl() {
        super();
    }

    protected EClass eStaticClass() {
        return GreeterDsl02Package.Literals.GREETING;
    }

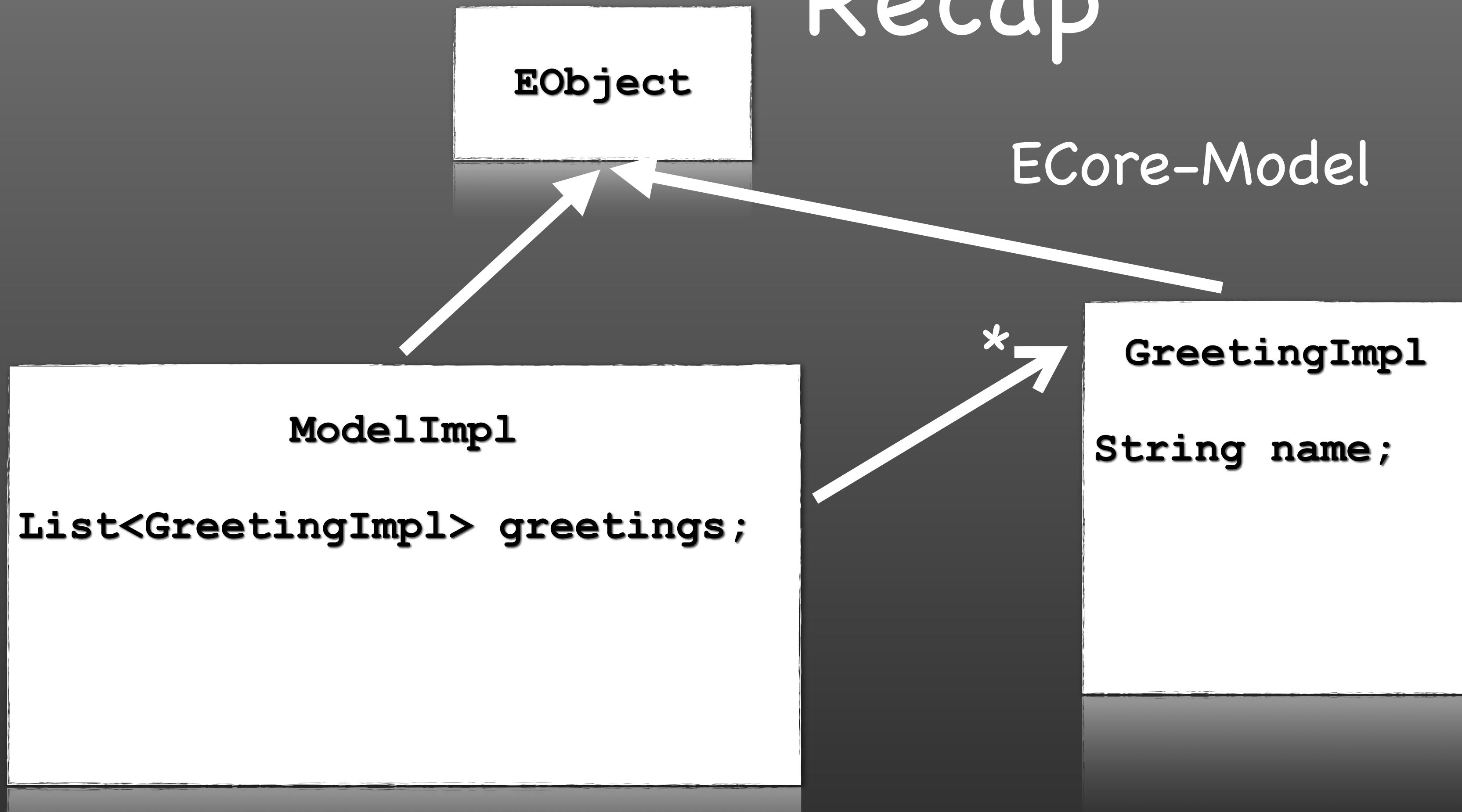
    public String getName() {
        return name;
    }
}

}

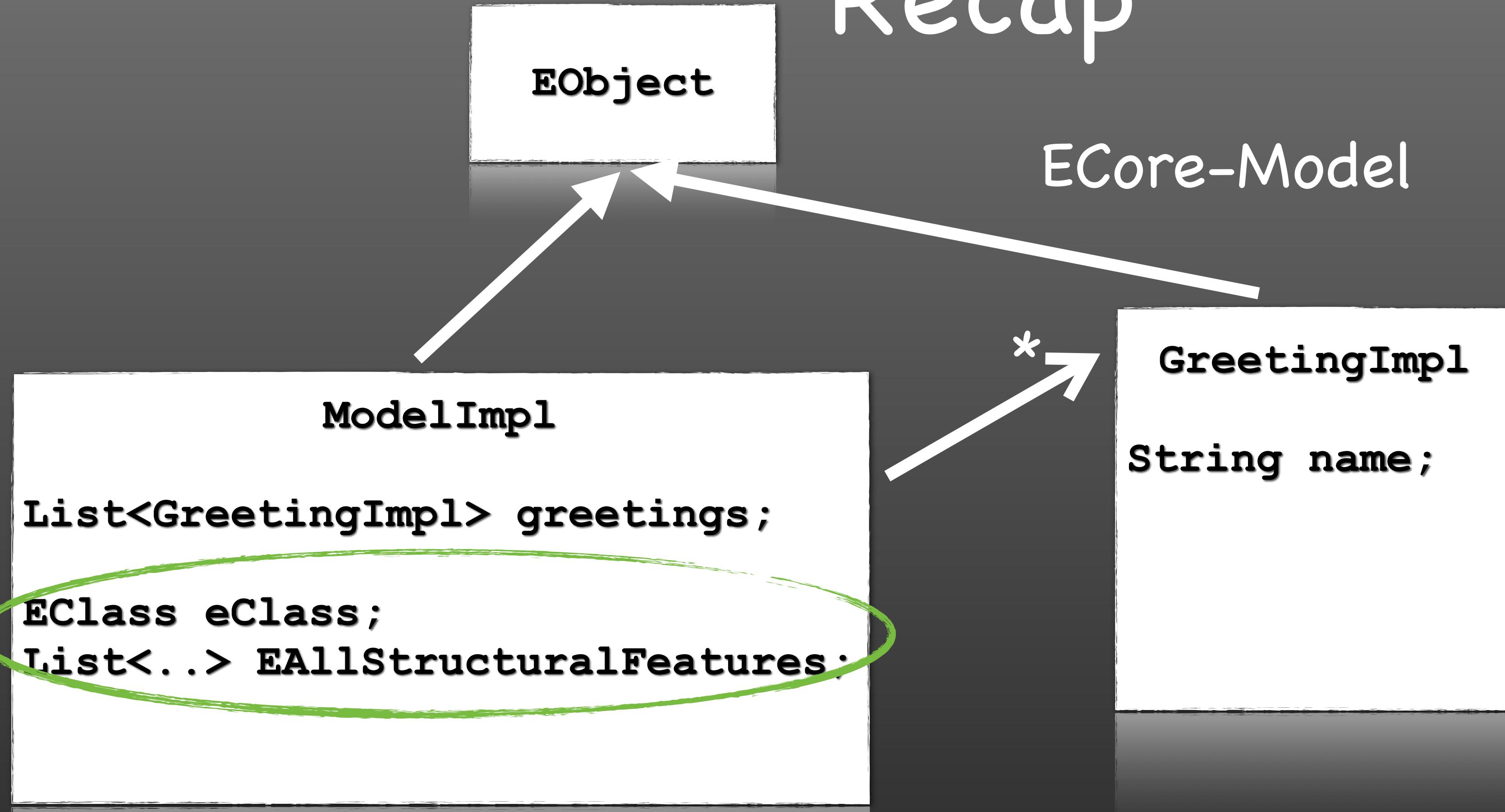
}

LEGALN NAME:
```

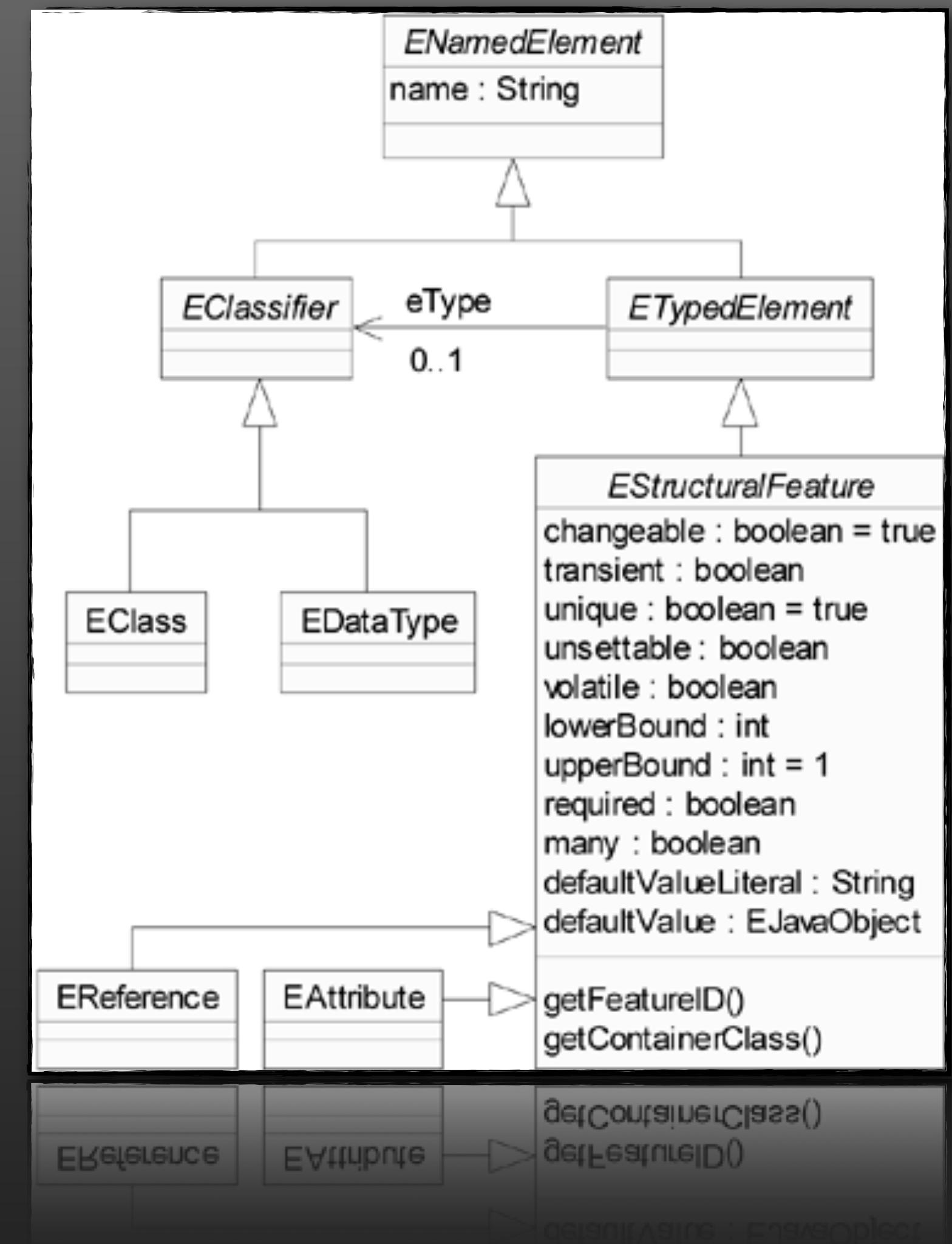
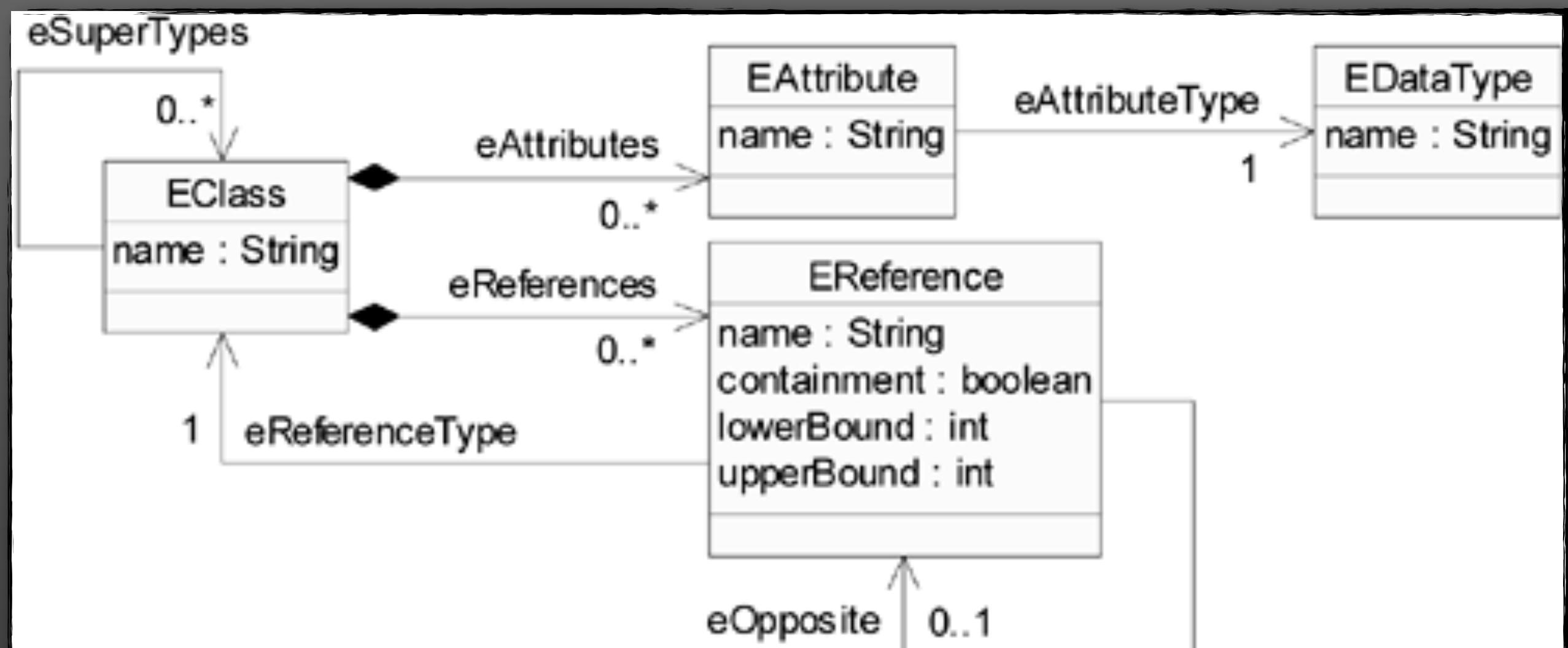
Recap



Recap



ecore Model



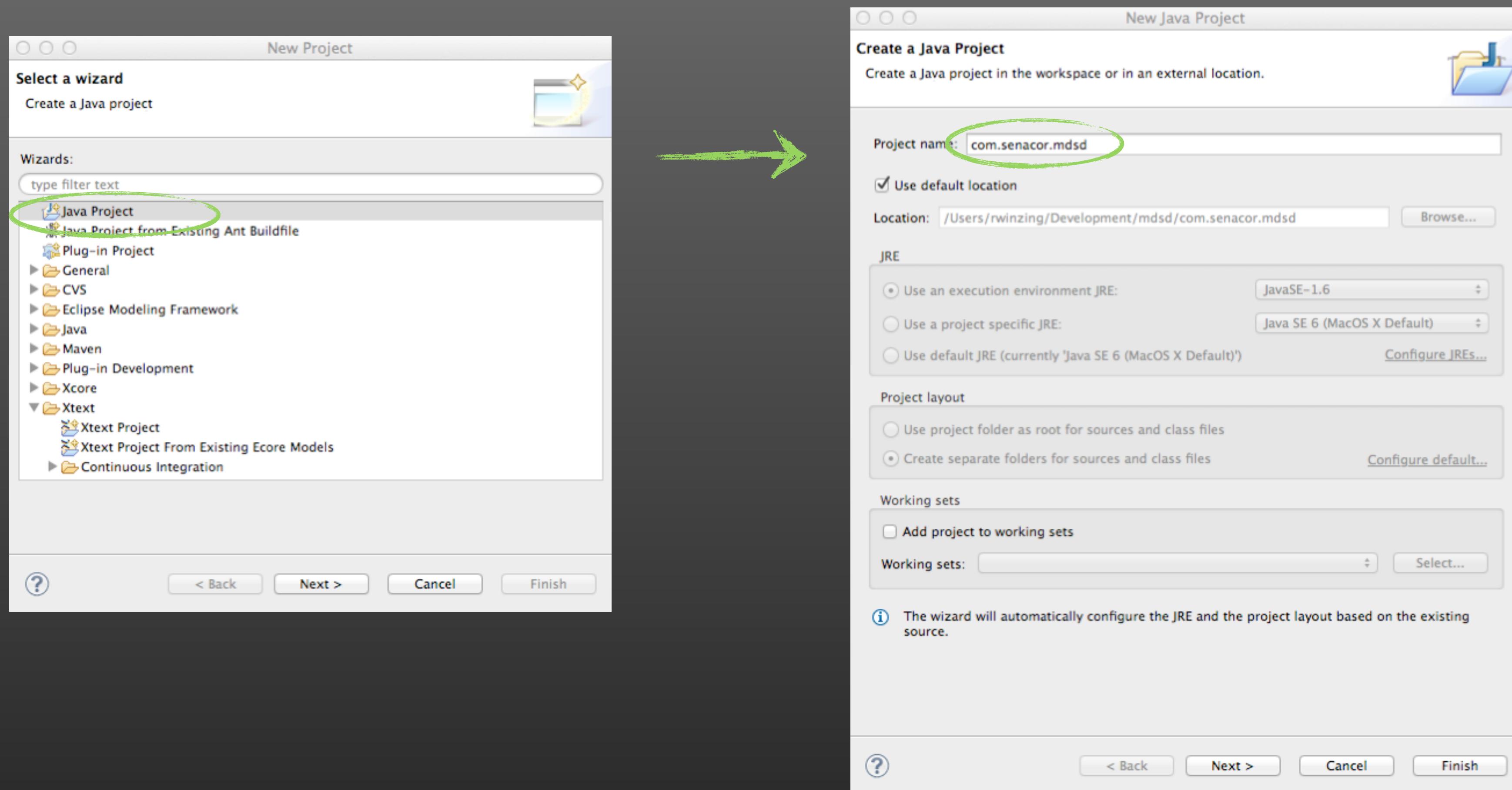
Xtend Basics

- Java Erweiterung (generiert wieder zu Java-Quellcode)
- Type inference
- Keywords def, val & var, dispatch
- Type extensions
- Lambdas
- Templates

Hello, World! (wieder mal)

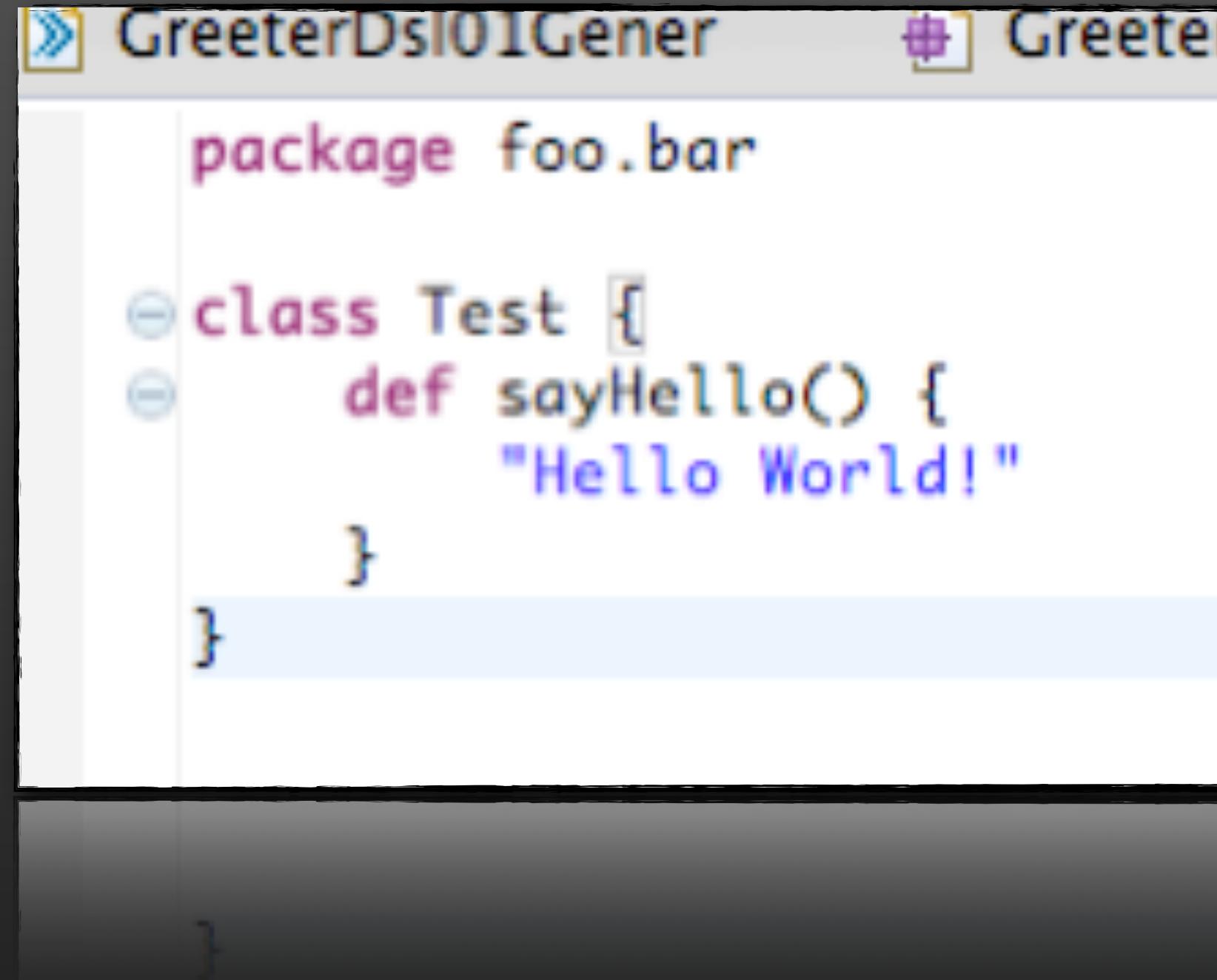
- Projekt erzeugen
- Xtend Klasse anlegen & speichern
- generierte Artefakte ansehen

Ein erstes Xtend Projekt ...



def, var & val

- „def“ deklariert eine Methode
- „var“ und „val“ deklarieren Variablen



```
GreeterDSL01Generator GreeterDSL01Generator
package foo.bar

class Test {
    def sayHello() =
        "Hello World!"
}
```

Transparenter Zugriff zwischen Java & Xtend

- „::“ für Zugriff auf statische Elemente



```
# GreeterDSI01.ec  GreeterDSI02.xt

package foo.bar

class Test {
    def printHello() {
        Util:::dump(sayHello)
    }

    def sayHello() {
        "Hello World!"
    }
}
```

Weshalb eigentlich „Xtend“?

foo(a, b) kann man schreiben als a.foo(b)

```
def dumpToConsole(String s) {  
    System.out.println(s);  
}  
  
dumpToConsole(„hello“);  
  
„hel  
for (gw: resource.allContents.toIterable.filter(typeof(GW))) {  
    generateFile(gw, fsa);  
}  
}
```

Templates

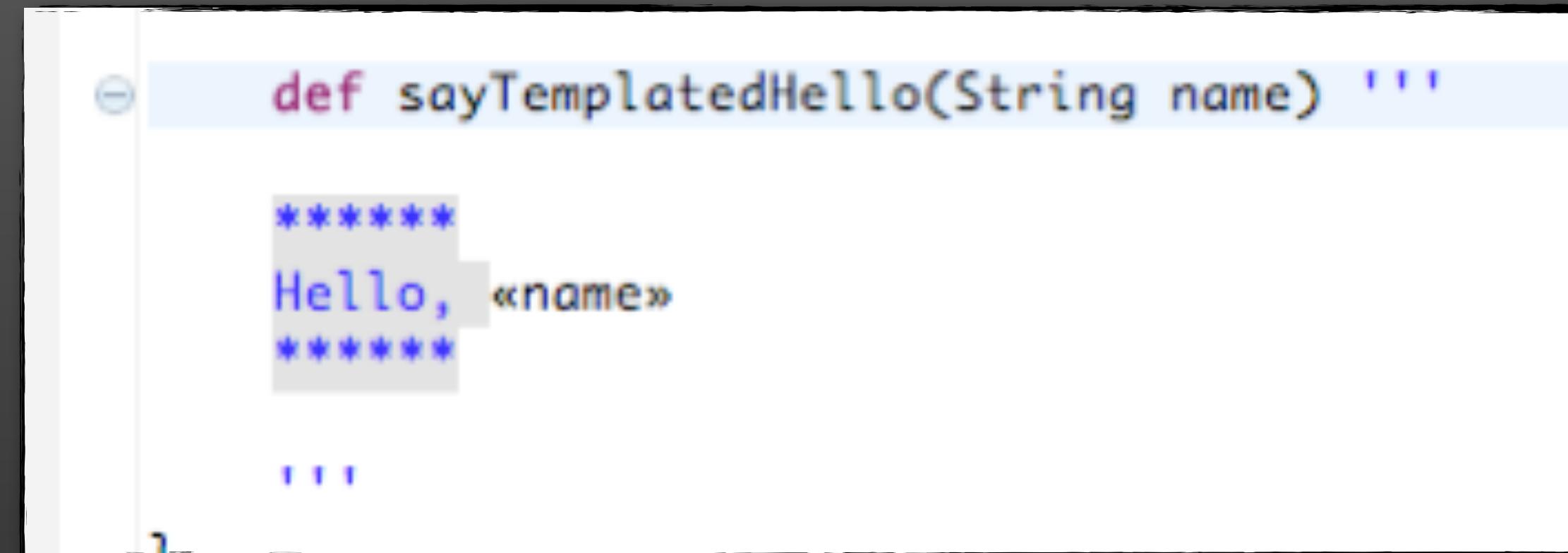
- Triple-Quotes leiten ein Template ein

- String-Ausdruck

- IF / ENDIF

- FOR / ENDFOR

- BEFORE / SEPARATOR / AFTER



A screenshot of a code editor showing a Python template definition. The code is:

```
def sayTemplatedHello(String name) ***  
    ****  
    Hello, «name»  
    ****  
    ...  
1
```

The string "Hello, «name»" is highlighted with a light gray background, indicating it is a placeholder or variable within the template.

editor tweaks

- Validatoren

- Labels, Icons

putting stuff together ...

- xtext und xtend verbinden

- und ein wenig generieren

Generation Gap

- Generierter und manuell implementierter Code müssen coexistieren können
- „Protected Regions“ reserviert Bereiche im generierten Code - leider fehleranfällig
- „Generation Gap“ verwendet OO-Mechanismen. Manueller Code wird in Subklassen verlagert, generiert werden nur (abstrakte) Oberklassen

<CODE>