



tutorial

Stefan Oehme, Moritz Eysholdt
itemis AG

Installation

- Grab a USB Key
- Install Eclipse
- Save the Zip Files to Your Disk
 - host_workspace.zip
 - runtime_workspace.zip

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate Models
- Cross-References and Scoping
- Advanced Grammar
- EMF / Ecore integration
- Index
- UI customizations
- Assigned Actions
- Testing
- Xbase
- Problem Solving
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

15:30 - 17:00

Outline

- **Example Application**
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate Models
- Cross-References and Scoping
- Advanced Grammar
- EMF / Ecore integration
- Index
- UI customizations
- Assigned Actions
- Testing
- Xbase
- Problem Solving
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

15:30 - 17:00

Eclipse Community Survey 2013

Eclipse Community Survey 2013 Evaluate

What is the *primary* computer language you typically use to develop software?

- C/C++
- C
- Lua
- Groovy
- Java
- Java Script

Ruby

- Ruby
- Scala
- Visual Basic/Visual Basic .Net
- None - I don't use a computer language for software development
- Don't know

Other (please specify)

Demo

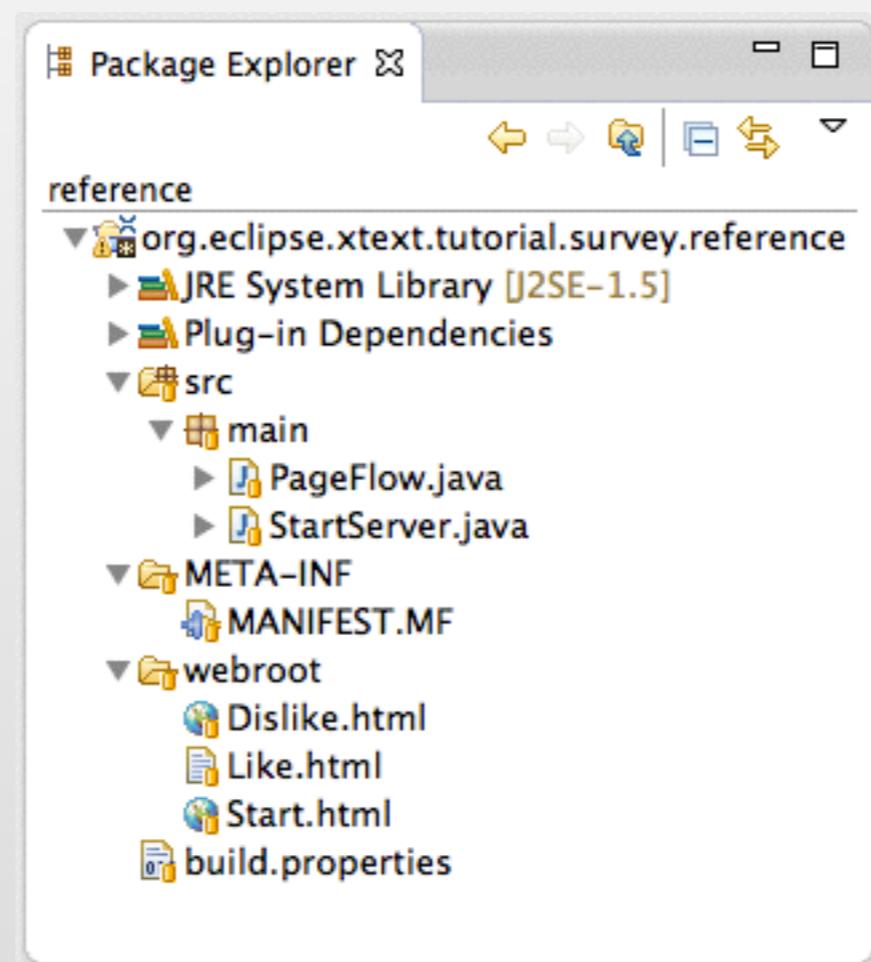
Example: Surveys

- ◉ A Web-Based App for Surveys
- ◉ Online Answering
- ◉ Multiple Pages
- ◉ Different Types of Questions
- ◉ Online Evaluation

Architecture

- HTML Forms
- Twitter Bootstrap CSS / JavaScript
- Jetty Server
- Simple Page-Flow Engine
- In-Memory Persistence

API View



Challenges

- ◉ A Heterogeneous Platform
 - ◉ Java, HTML, Database ...
- ◉ Difficult to Extend
 - ◉ More Questions, Other Surveys
 - ◉ Other Front-Ends (iOS, Android, PDF, ...)
- ◉ Hard to Maintain

The Domain

- The application is about **Surveys**.
- A **Survey** consists of **Pages**.
- A **Page** holds **Questions**.
- **Questions** are answered with **FreeText** or predefined **Choices**.
 - Some **Choices** are **exclusive**.
- A **Page** defines its **FollowUp** pages
 - **FollowUps** may depend on given answers.

DSL Approach

- Create a Domain-Specific Language
 - Describe the Data Formally
 - Generate Code from DSL Files

Survey

Page

TextQuestion

ChoiceQuestion (single)

Choice

Choice

FollowUp

FollowUp

Page

ChoiceQuestion

Choice

Choice

Choice

Page

ChoiceQuestion

Choice

Choice

Choice

```
survey tutorial "EclipseCon 2013 Tutorial Survey"

page Start (
    text name 'Your name'
    single choice like "Do you like the tutorial?" (
        yes "Yes"
        no "No"
        wat "Which tutorial? I'm waiting for the bus!"
    )
    if like=yes -> Like
    if like=no -> Dislike
)

page Like (
    choice particular "What do you like in particular?" (
        xtext 'Xtext is awesome'
        excercises 'The funny exercises'
        tutors 'The handsome tutors'
    )
)

page Dislike (
    choice particular "What do you hate in particular?" (
        xtext 'Xtext sucks'
        excercises 'The boring exercises'
        tutors 'The tutors stink'
    )
)
```

Advantages

- Addresses Heterogeneity
- Easy Design of Surveys
- Easy to Add New Front-Ends
- Separation of Roles During Development
- Speed-up for Development
- Improved Maintainability

Outline

- Example Application
- **Build your DSL with Xtext**
- Generate Code with Xtend
- Validate Models
- Cross-References and Scoping
- Advanced Grammar
- EMF / Ecore integration
- Index
- UI customizations
- Assigned Actions
- Testing
- Xbase
- Problem Solving
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

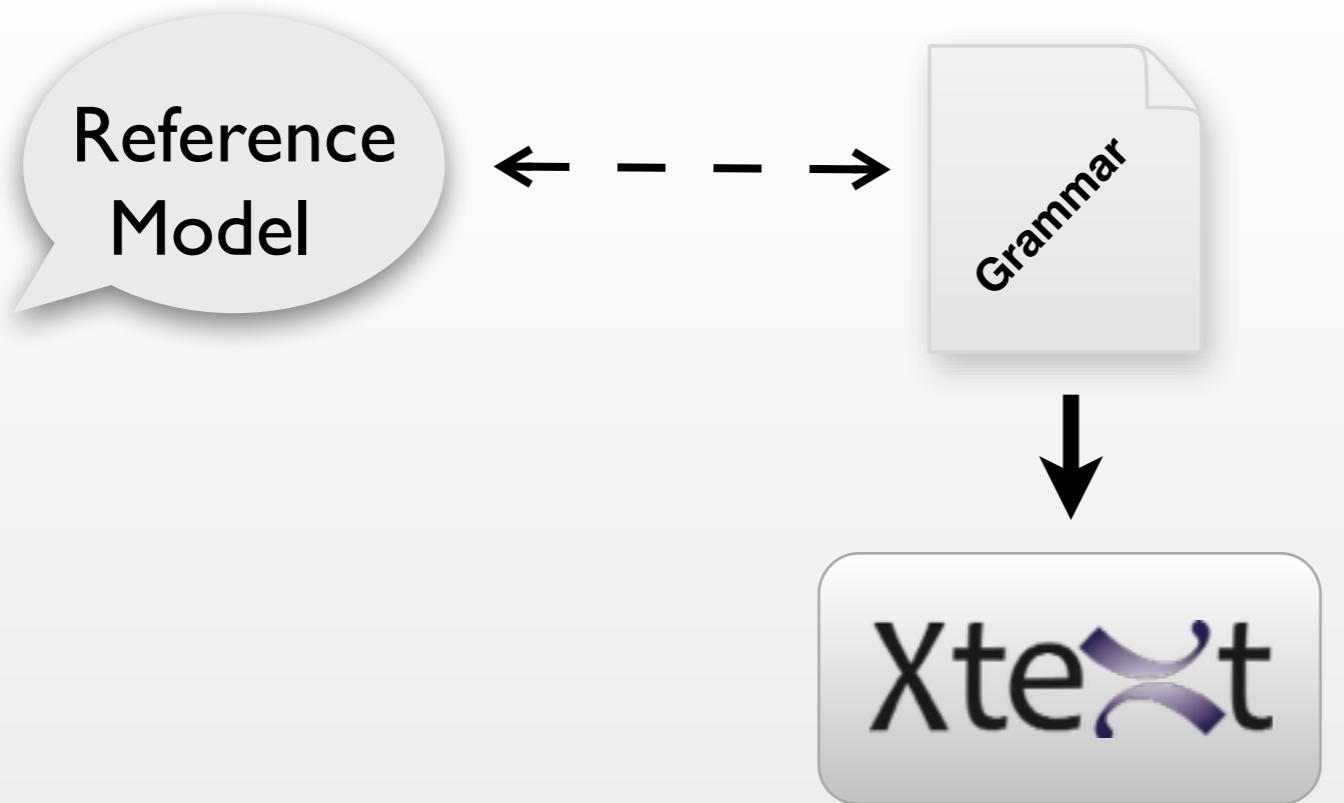
13:30 - 15:00

15:30 - 17:00



Reference
Model





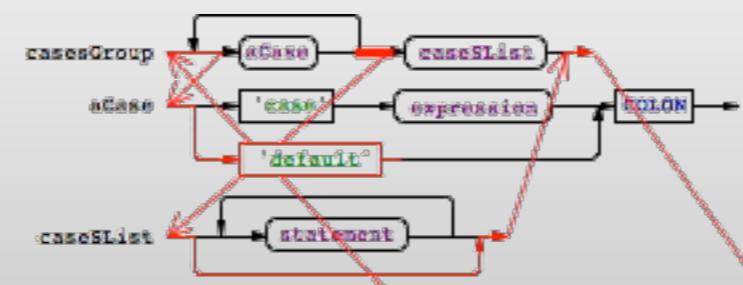
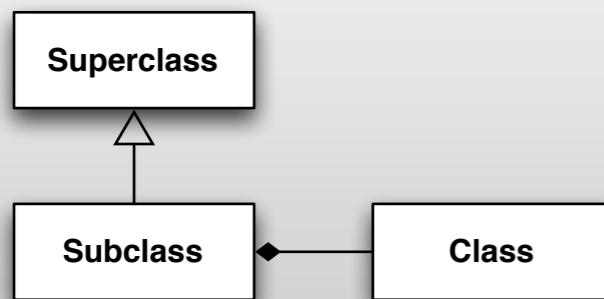
Reference Model



Grammar



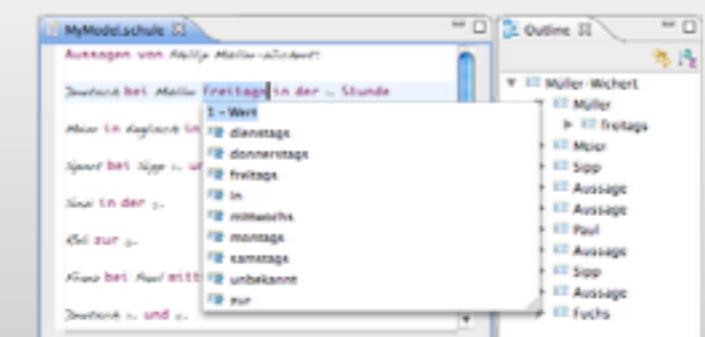
Xtext Runtime



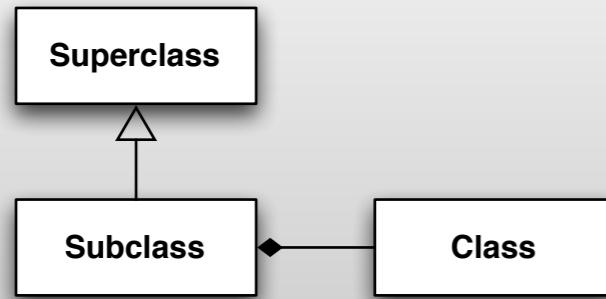
Ecore Package

Parser

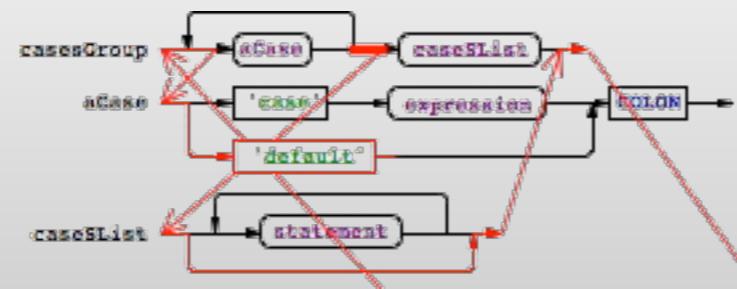
Editor



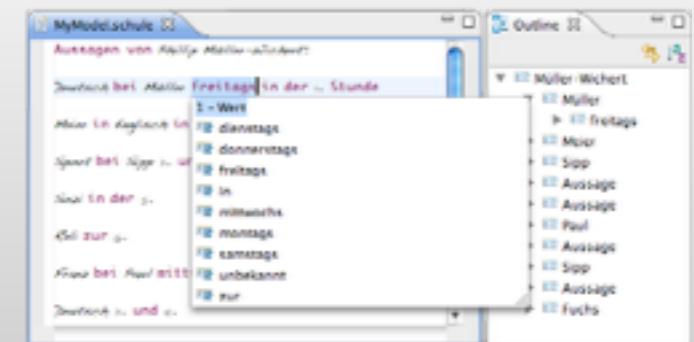
Xtext Runtime



Ecore model

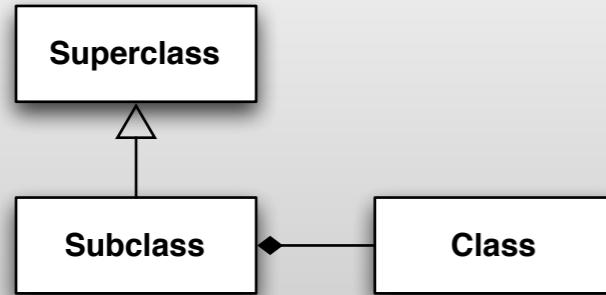


LL(*) Parser

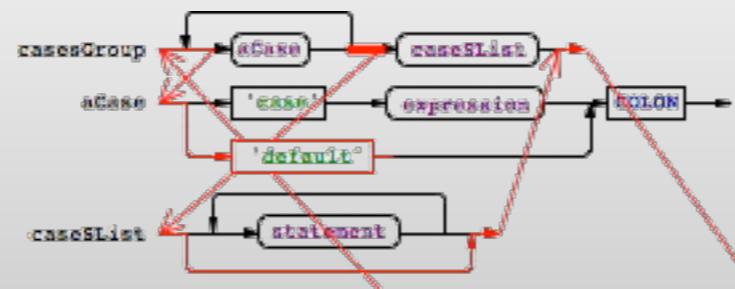


Editor

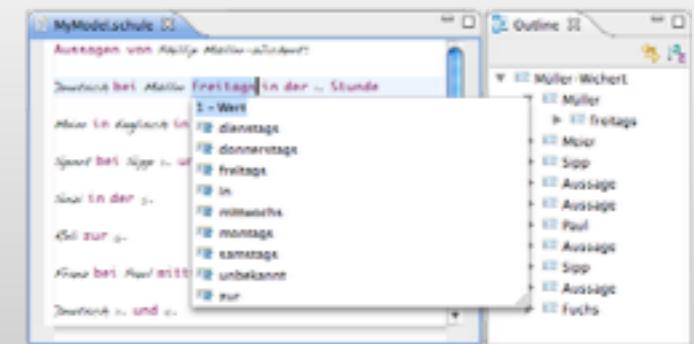
Xtext Runtime



Ecore model



LL(*) Parser

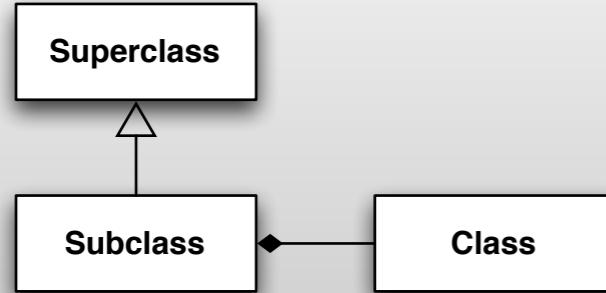


Editor

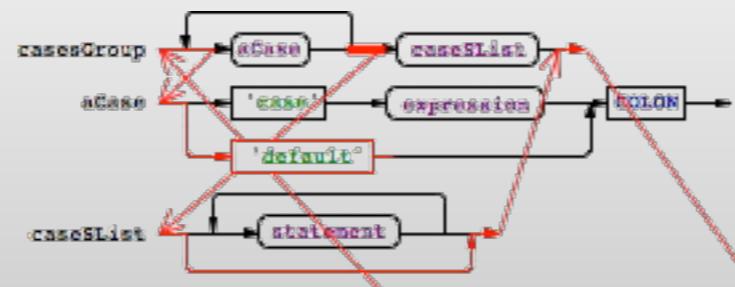
<creates>

Textual
Model

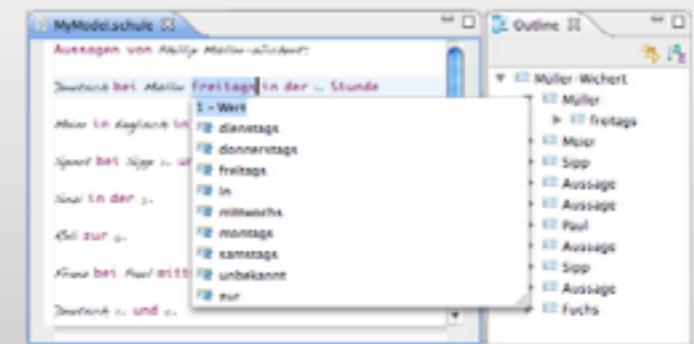
Xtext Runtime



Ecore model



LL(*) Parser



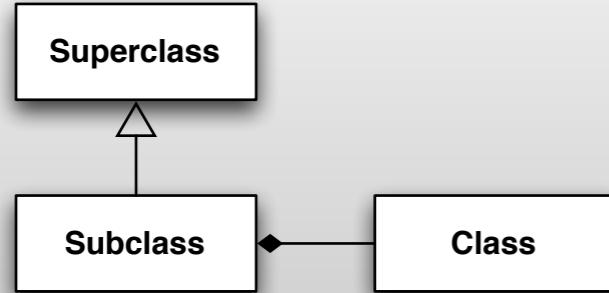
Editor

<parsed by>

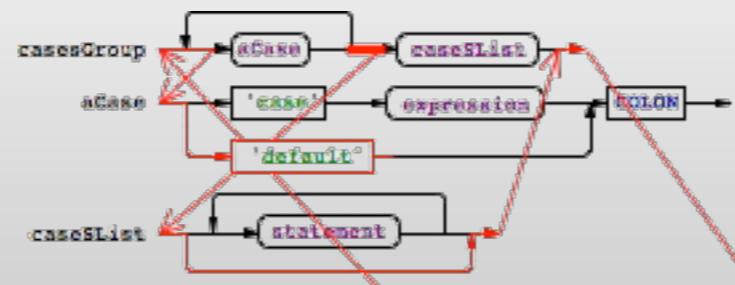
<creates>

Textual
Model

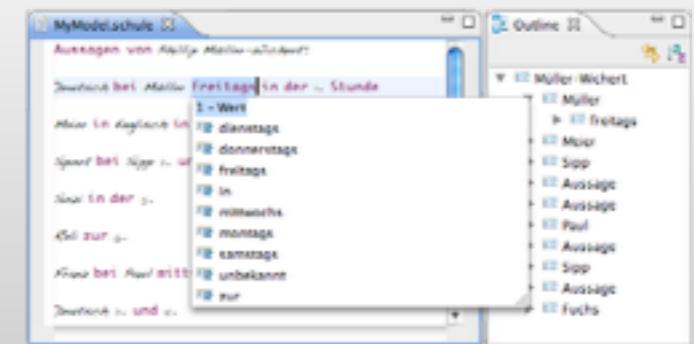
Xtext Runtime



Ecore model

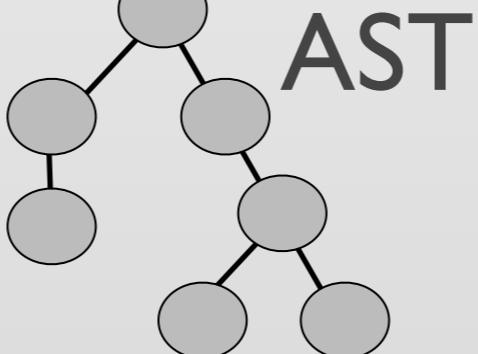


LL(*) Parser



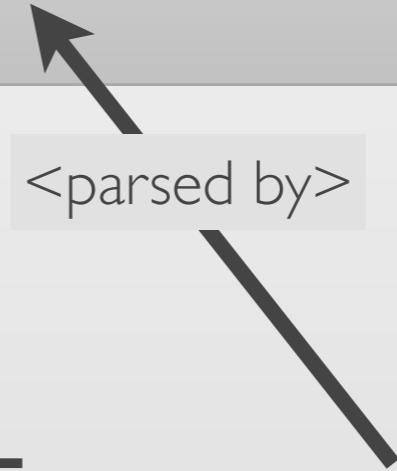
Editor

<instantiates>

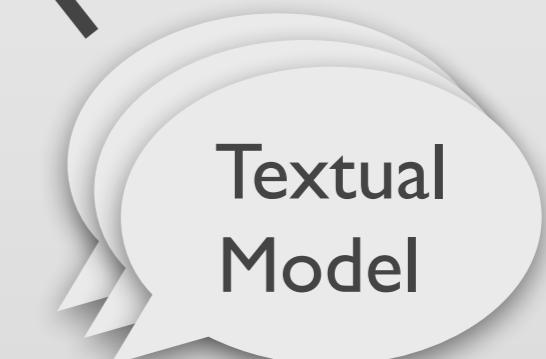


AST

<parsed by>

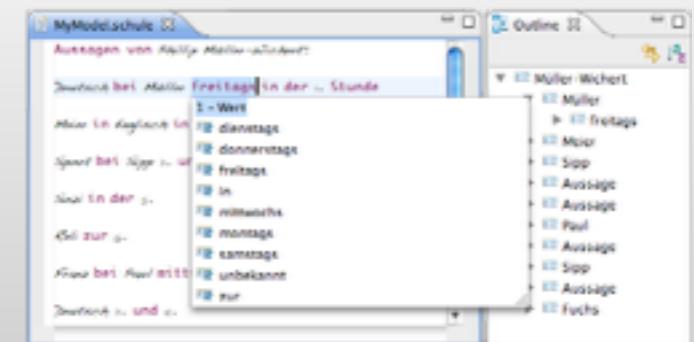
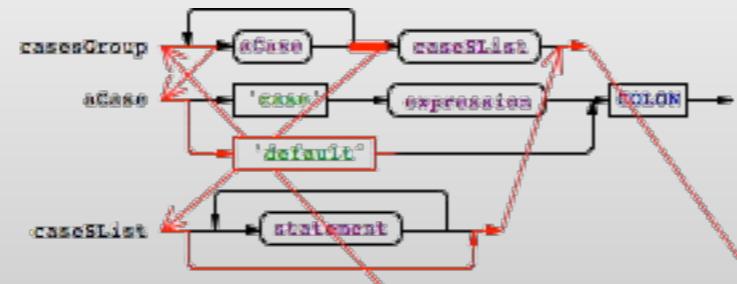
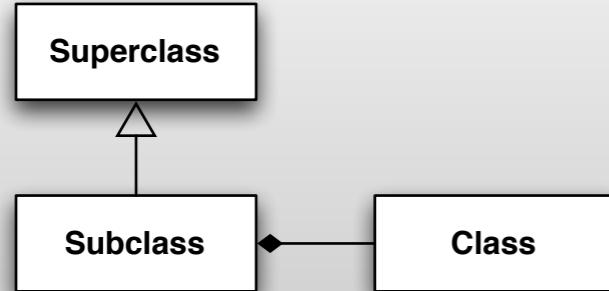


<creates>



Textual
Model

Xtext Runtime



Ecore model

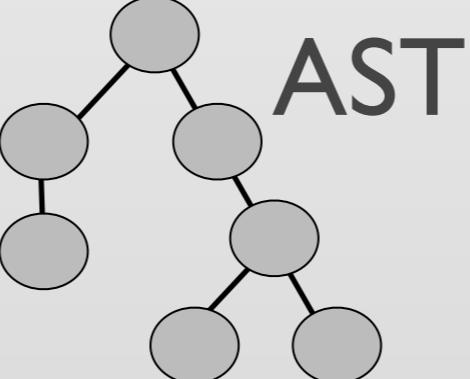
LL(*) Parser

Editor

<instanceof> <instantiates>

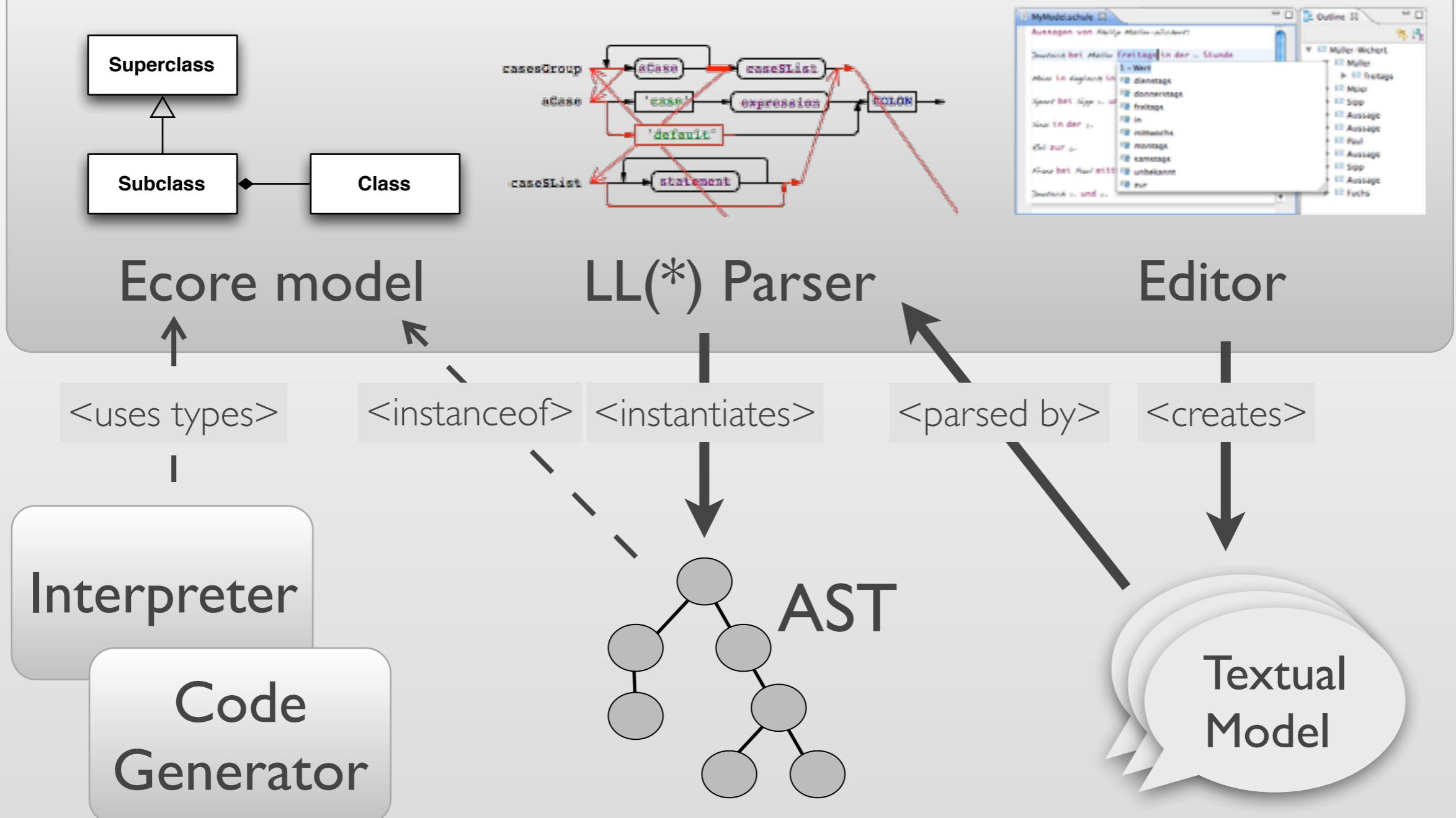
<parsed by>

<creates>

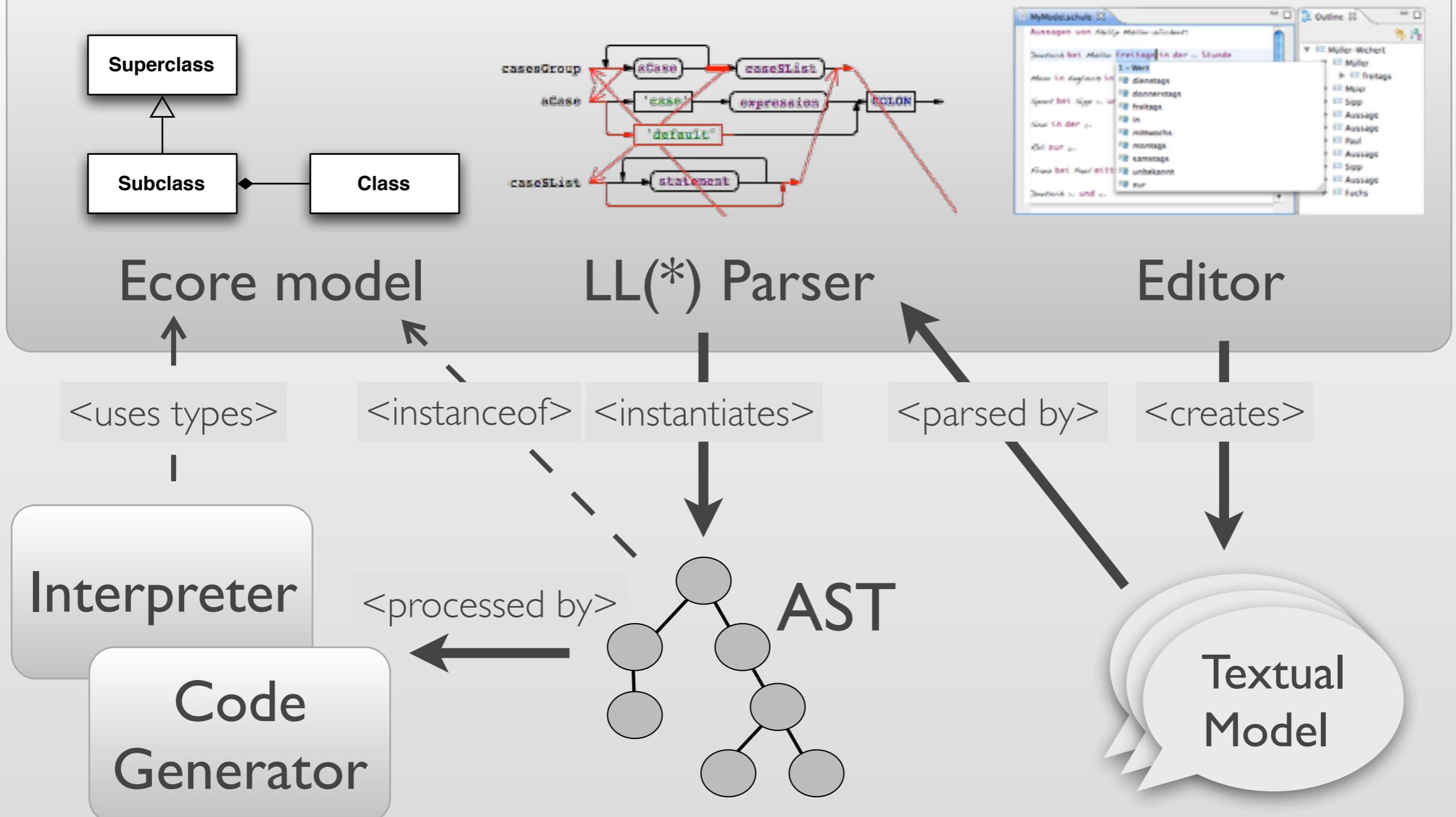


Textual
Model

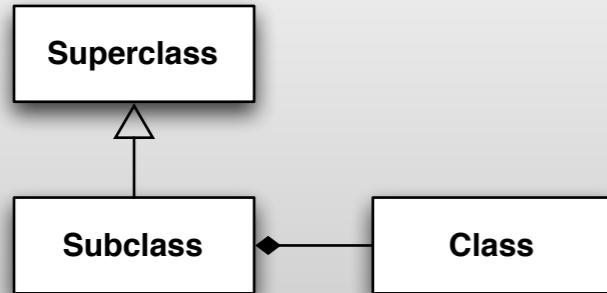
Xtext Runtime



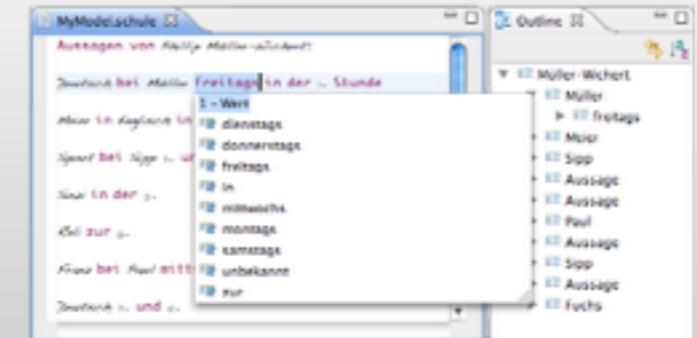
Xtext Runtime



Xtext Runtime

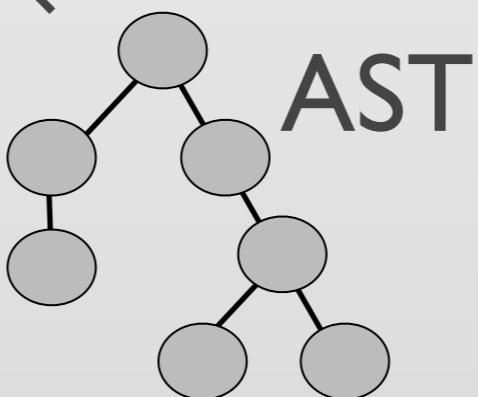


Ecore model



Editor

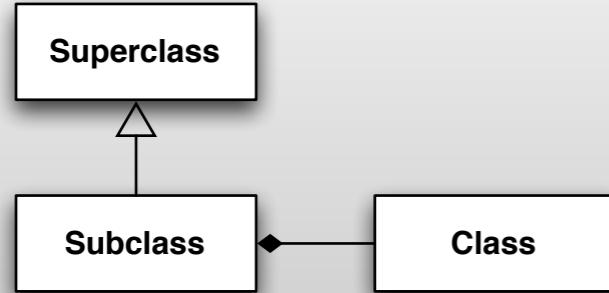
<instanceof>



AST

Textual
Model

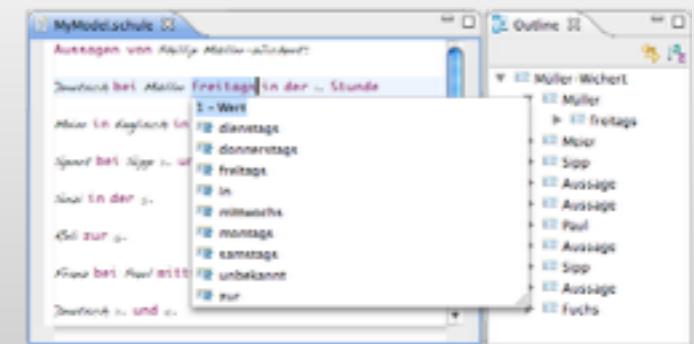
Xtext Runtime



Ecore model

<uses types>

<instanceof>

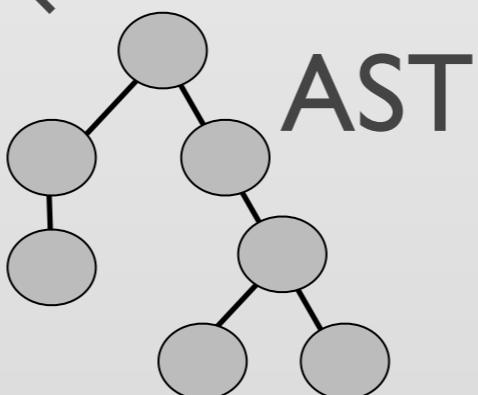


Editor

Quickfix

Refactoring

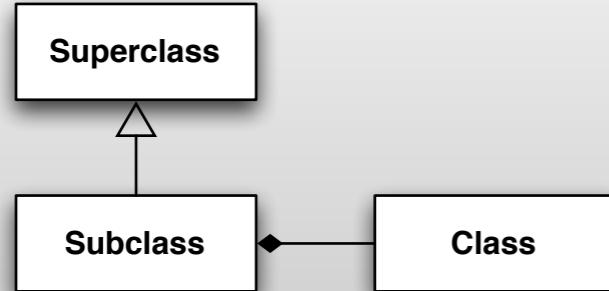
Graphical
Editor



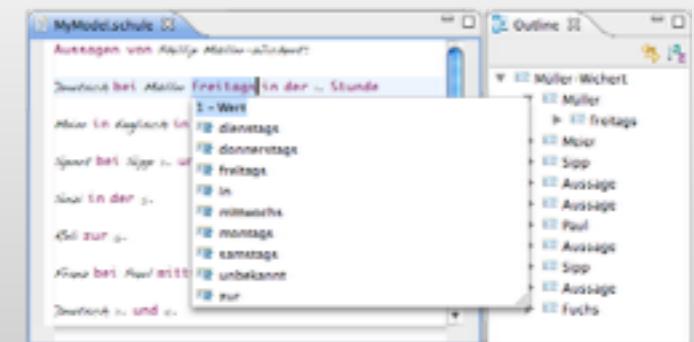
AST

Textual
Model

Xtext Runtime



Ecore model



Editor

<uses types>

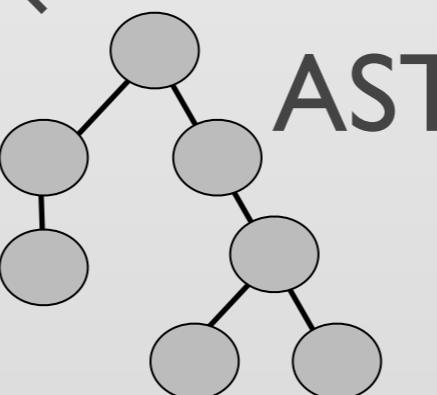
<instanceof>

Quickfix

Refactoring

Graphical
Editor

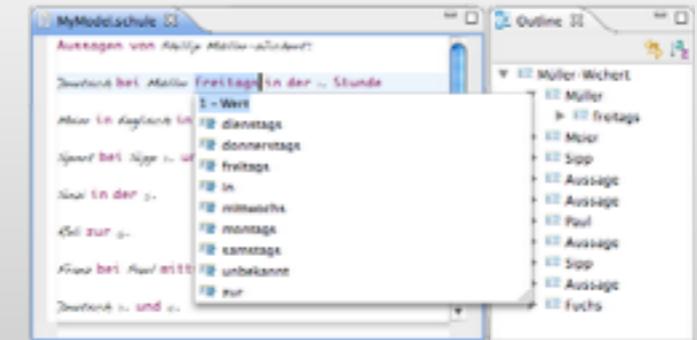
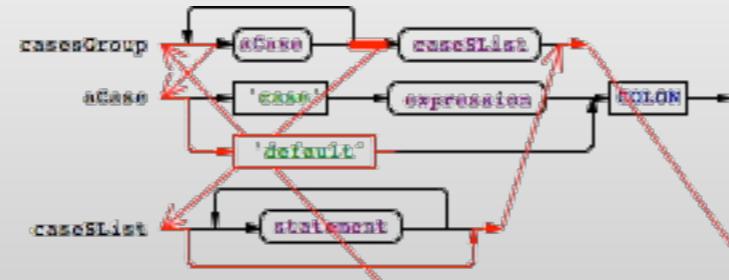
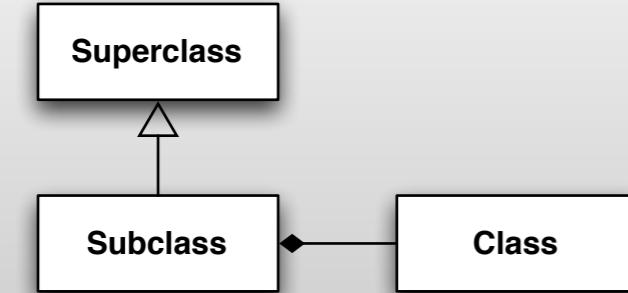
<processes>



AST

Textual
Model

Xtext Runtime



Ecore model



<uses types>

<instanceof>

<input for>

Serializer

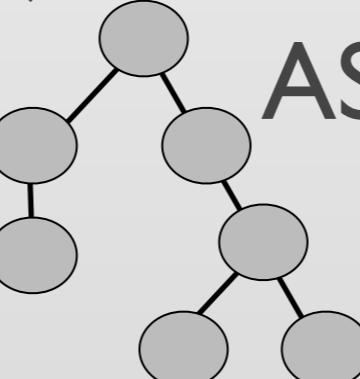
Editor

Quickfix

Refactoring

Graphical
Editor

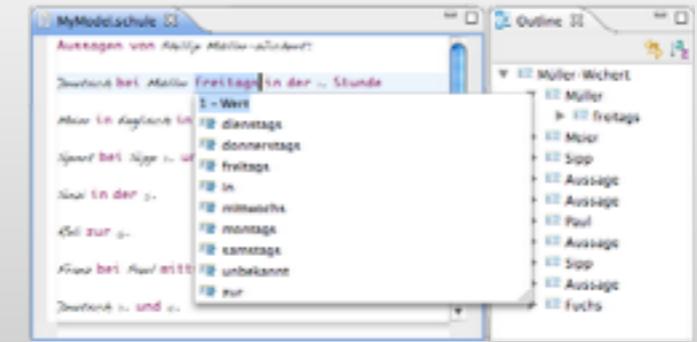
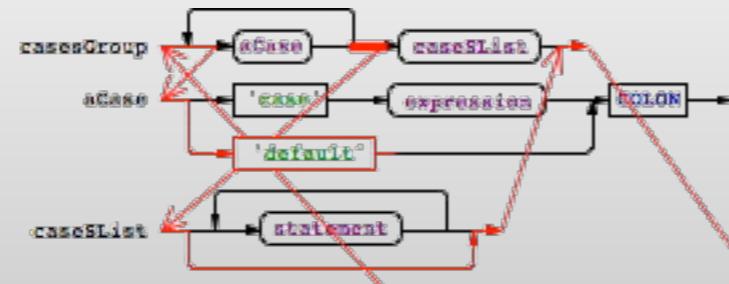
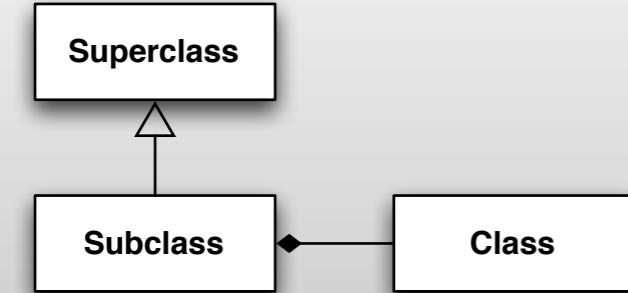
<processes>



AST

Textual
Model

Xtext Runtime



Ecore model

<uses types>

<instanceof>

<input for>

Editor

Quickfix

Refactoring

Graphical
Editor

Serializer

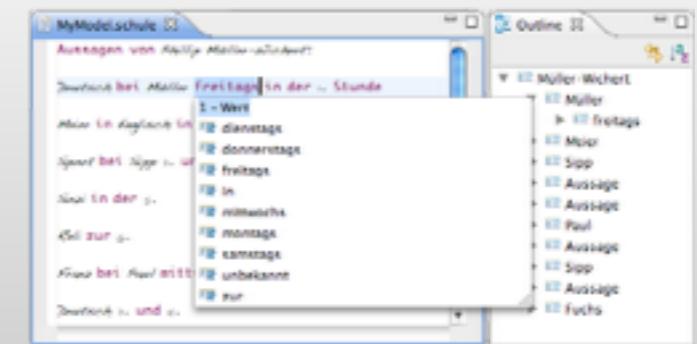
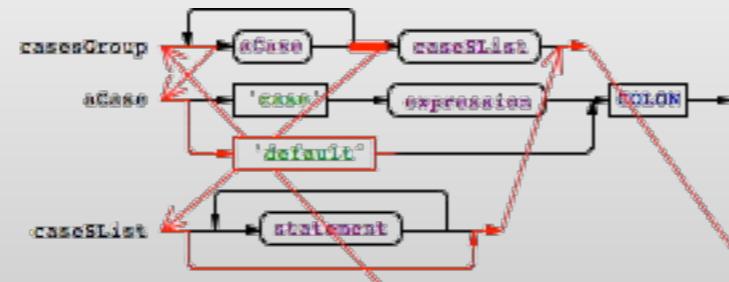
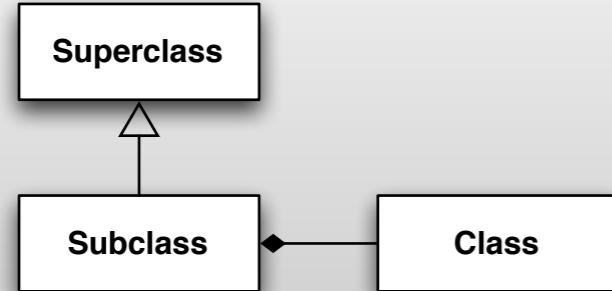
<creates/updates>

<processes>

AST

Textual
Model

Xtext Runtime



Ecore model

Serializer

Editor

<uses types>

<instanceof>

<input for>

<creates/updates>

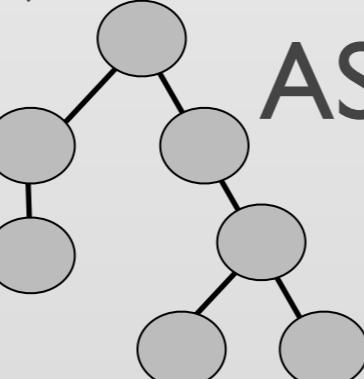
<populates>

Quickfix

Refactoring

Graphical
Editor

<processes>



AST

Textual
Model

EXERCISE: NEW PROJECT

```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals

generate companyModel "http://www.xtext.org/workshop/CompanyModel"
```

Company:

```
(assets+=Asset)+;
```

Asset:

```
Employee | Project;
```

Employee:

```
'employee' name=ID;
```

Project:

```
(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';
```

Task returns WorkItem:

```
'task' name=ID 'assigned to'
assignee=[Employee];
```

```
employee Peter
inhouse project Survey {
    task kick_off assigned to Peter
}
```

Grammar Definition

```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals

generate companyModel "http://www.xtext.org/workshop/CompanyModel"

Company:
(assets+=Asset)+;

Asset:
Employee | Project;

Employee:
'employee' name=ID;

Project:
(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';

Task returns WorkItem:
'task' name=ID 'assigned to'
assignee=[Employee];
```

```
employee Peter
inhouse project Survey {
    task kick_off assigned to Peter
}
```

```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals
```

```
generate companyModel "http://www.xtext.org/workshop/CompanyModel"
```

Company:

```
(assets+=Asset)+;
```

Asset:

```
Employee | Project;
```

Employee:

```
'employee' name=ID;
```

Project:

```
(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';
```

Task returns WorkItem:

```
'task' name=ID 'assigned to'
assignee=[Employee];
```

```
employee Peter
inhouse project Survey {
    task kick_off assigned to Peter
}
```

```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals
```

Derived Metamodel  generate companyModel "http://www.xtext.org/workshop/CompanyModel"

Company:

```
(assets+=Asset)+;
```

Asset:

```
Employee | Project;
```

Employee:

```
'employee' name=ID;
```

Project:

```
(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';
```

Task returns WorkItem:

```
'task' name=ID 'assigned to'
assignee=[Employee];
```

```
employee Peter
inhouse project Survey {
    task kick_off assigned to Peter
}
```

```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals

generate companyModel "http://www.xtext.org/workshop/CompanyModel"
```

Parser Rules

```
Company:
(assets+=Asset)+;

Asset:
Employee | Project;

Employee:
'employee' name=ID;

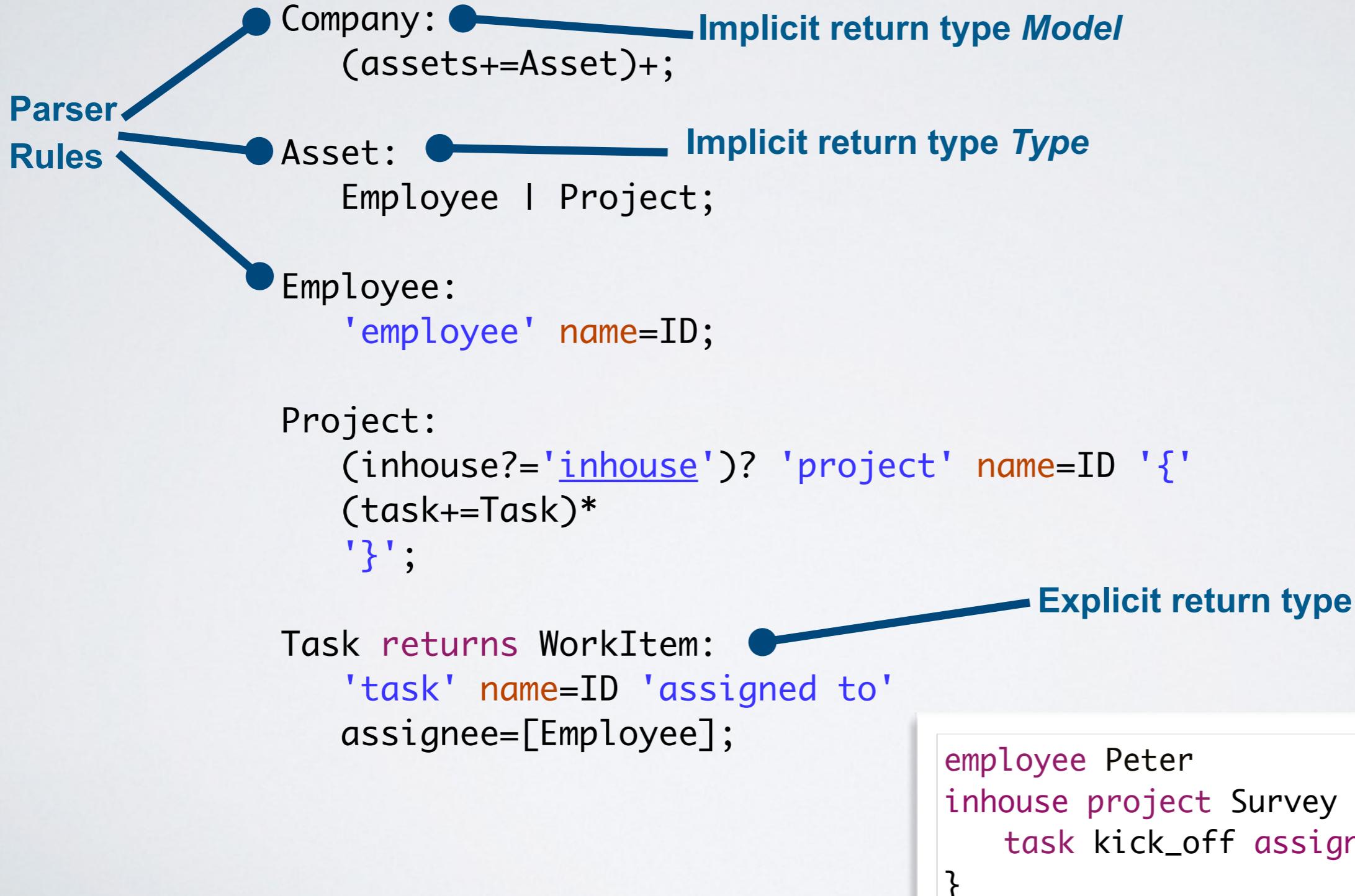
Project:
(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';
```

```
Task returns WorkItem:
'task' name=ID 'assigned to'
assignee=[Employee];
```

```
employee Peter
inhouse project Survey {
    task kick_off assigned to Peter
}
```

```
grammar org.xtext.workshop.DomainModel  
with org.eclipse.xtext.common.Terminals
```

```
generate companyModel "http://www.xtext.org/workshop/CompanyModel"
```



```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals

generate companyModel "http://www.xtext.org/workshop/CompanyModel"
```

Company:

(assets+=Asset)+;

Asset:

Employee | Project;

Employee:

'employee' name=ID;

Project:

(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';

Keywords

Task returns WorkItem:

'task' name=ID 'assigned to'
assignee=[Employee];

```
employee Peter
inhouse project Survey {
    task kick_off assigned to Peter
}
```

```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals

generate companyModel "http://www.xtext.org/workshop/CompanyModel"
```

Company:
(assets+=Asset)+;

Multivalue Assignment

Asset:
Employee | Project;

Employee:
'employee' name=ID;

Simple Assignment

Project:
(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';

Boolean Assignment

Task returns WorkItem:
'task' name=ID 'assigned to'
assignee=[Employee];

employee Peter
inhouse project Survey {
task kick_off assigned to Peter
}

```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals

generate companyModel "http://www.xtext.org/workshop/CompanyModel"
```

Company:

(assets+=Asset)+;

Cardinality (one or more)

Asset:

Employee | Project;

Employee:

'employee' name=ID;

Project:

(inhouse?='inhouse')? 'project' name=ID '{'

Optional (zero or one)

(task+=Task)* '}';

Cardinality (zero or more)

Task returns WorkItem:

'task' name=ID 'assigned to'
assignee=[Employee];

```
employee Peter
inhouse project Survey {
    task kick_off assigned to Peter
}
```

```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals

generate companyModel "http://www.xtext.org/workshop/CompanyModel"
```

Company:

```
(assets+=Asset)+;
```

Asset:

```
Employee | Project;
```

Alternative

Employee:

```
'employee' name=ID;
```

Project:

```
(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';
```

Task returns WorkItem:

```
'task' name=ID 'assigned to'
assignee=[Employee];
```

```
employee Peter
inhouse project Survey {
    task kick_off assigned to Peter
}
```

```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals

generate companyModel "http://www.xtext.org/workshop/CompanyModel"
```

Company:

(assets+=Asset)+;

Asset:

Employee | Project;

Employee:

'employee' name=ID;  **Attribute**

Project:

(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*  **Containment Reference**
'}';

Task returns WorkItem:

'task' name=ID 'assigned to'
assignee=[Employee];



Cross Reference

```
employee Peter
inhouse project Survey {
    task kick_off assigned to Peter
}
```

```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals
```

```
generate companyModel
"http://www.xtext.org/workshop/CompanyModel"
```

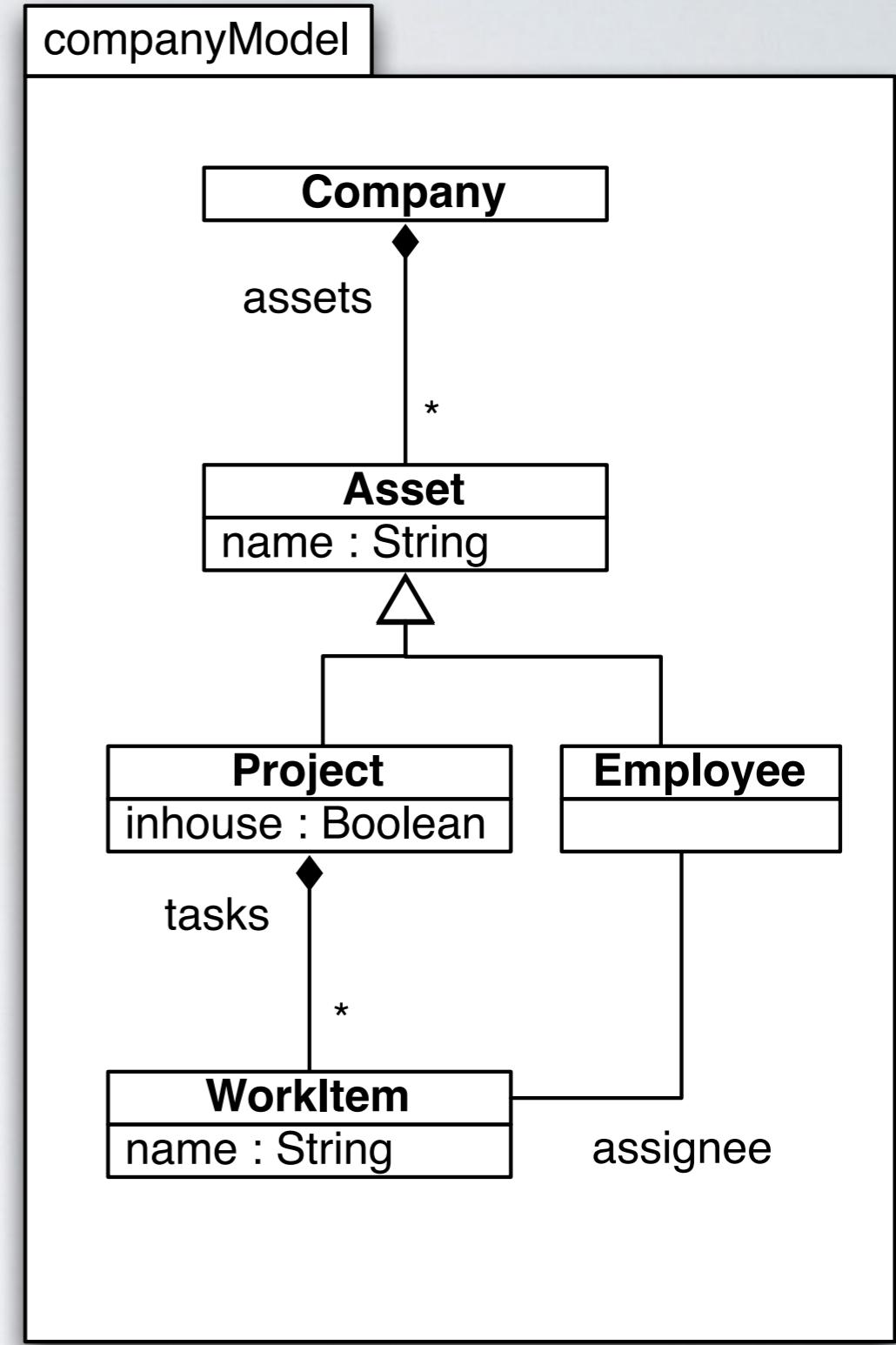
```
Company:
(assets+=Asset)+;
```

```
Asset:
Project | Employee;
```

```
Employee:
'employee' name=ID;
```

```
Project:
(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';
```

```
Task returns WorkItem:
'task' name=ID 'assigned to'
assignee=[Employee];
```



```

grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals

generate companyModel
"http://www.xtext.org/workshop/CompanyModel"

```

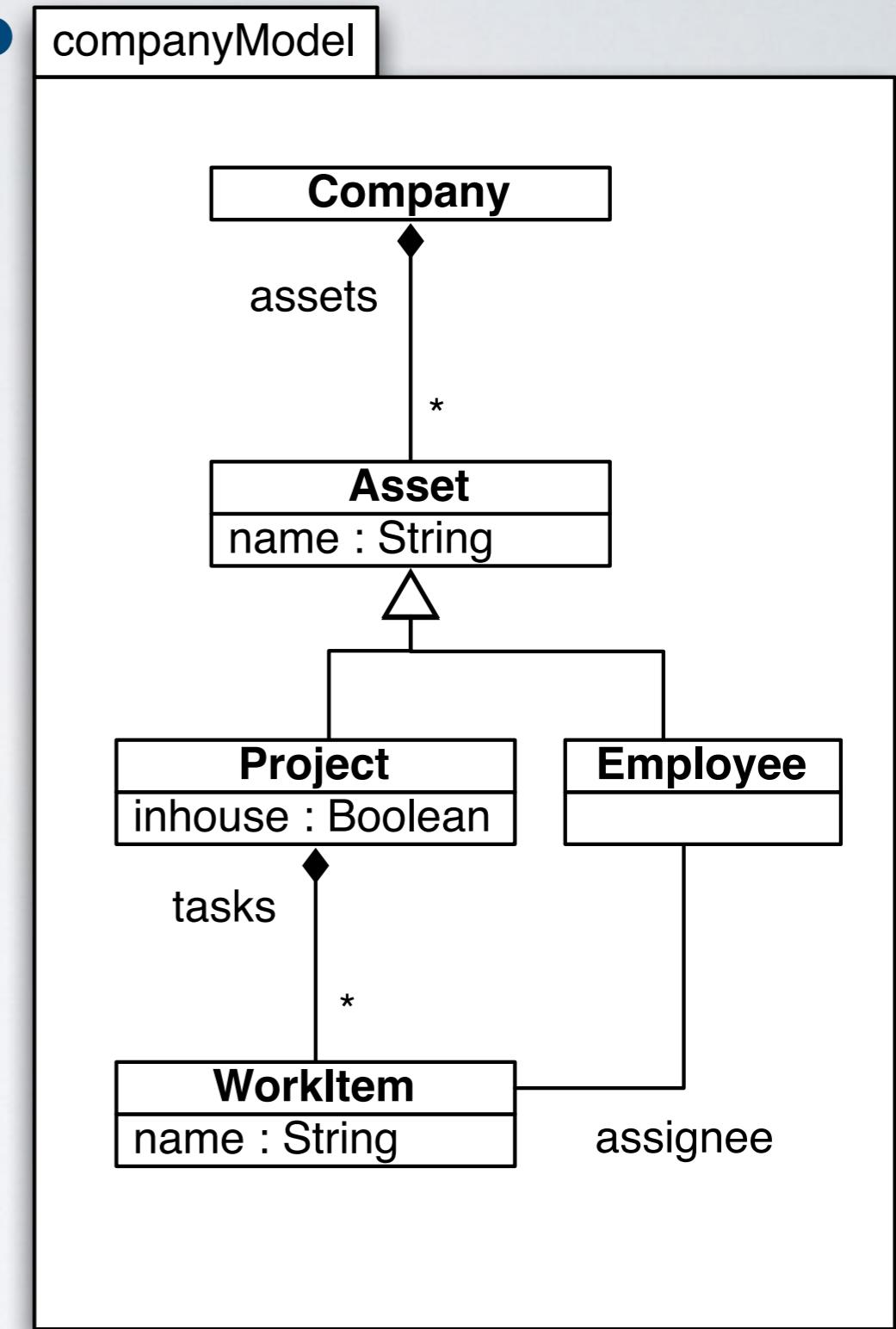
Company:
`(assets+=Asset)+;`

Asset:
`Project | Employee;`

Employee:
`'employee' name=ID;`

Project:
`(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';`

Task **returns** WorkItem:
`'task' name=ID 'assigned to'
assignee=[Employee];`



```
grammar org.xtext.workshop.DomainModel  
with org.eclipse.xtext.common.Terminals
```

```
generate companyModel  
"http://www.xtext.org/workshop/CompanyModel"
```

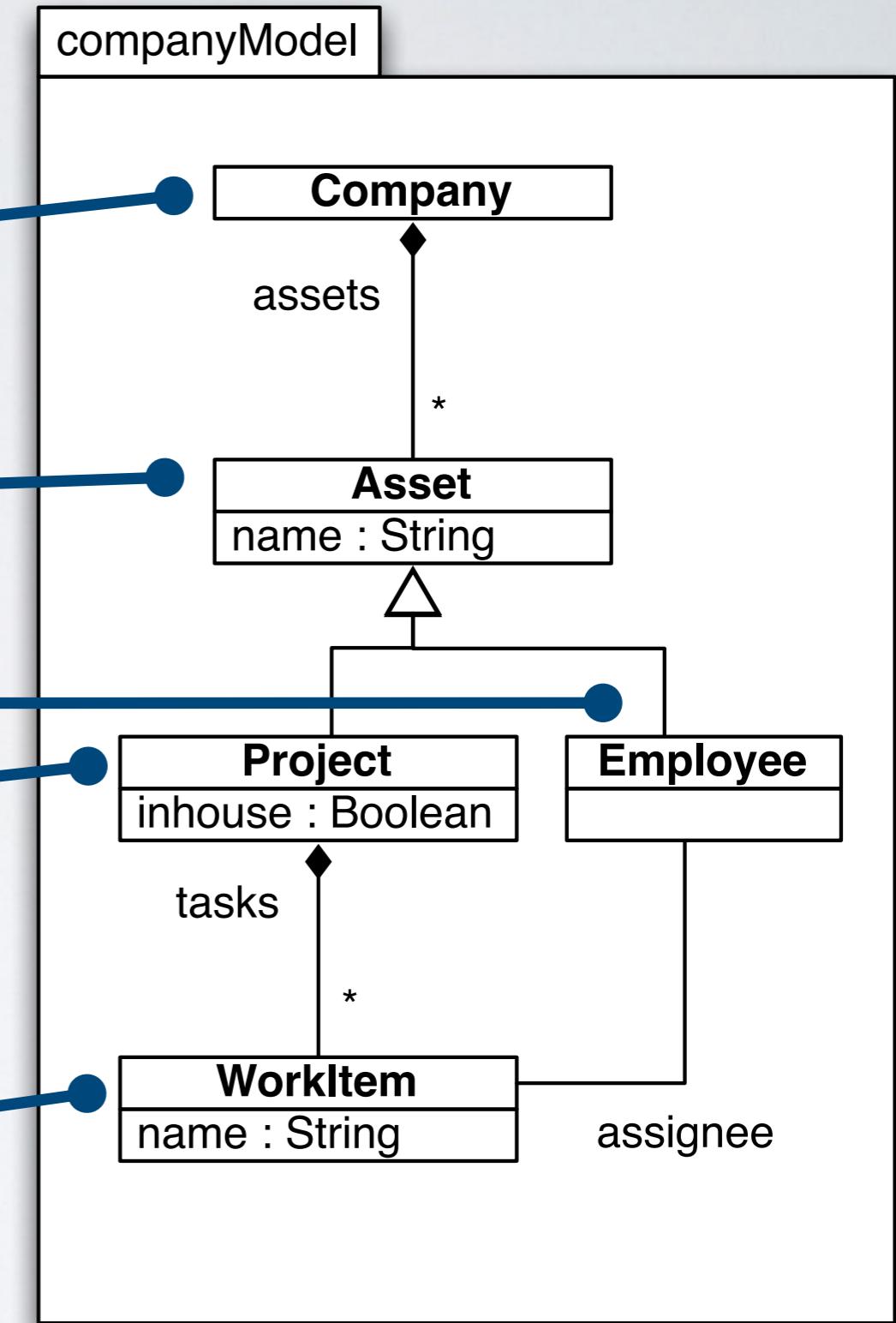
```
Company: •  
(assets+=Asset)+;
```

```
Asset: •  
Project | Employee;
```

```
Employee: •  
'employee' name=ID;
```

```
Project: •  
(inhouse?='inhouse')? 'project' name=ID '{'  
(task+=Task)*  
'}';
```

```
Task returns WorkItem:  
'task' name=ID 'assigned to'  
assignee=[Employee];
```



```
grammar org.xtext.workshop.DomainModel
```

```
with org.eclipse.xtext.common.Terminals
```

```
generate companyModel
```

```
"http://www.xtext.org/workshop/CompanyModel"
```

```
Company:
```

```
(assets+=Asset)+;
```

```
Asset:
```

```
Project | Employee;
```

```
Employee:
```

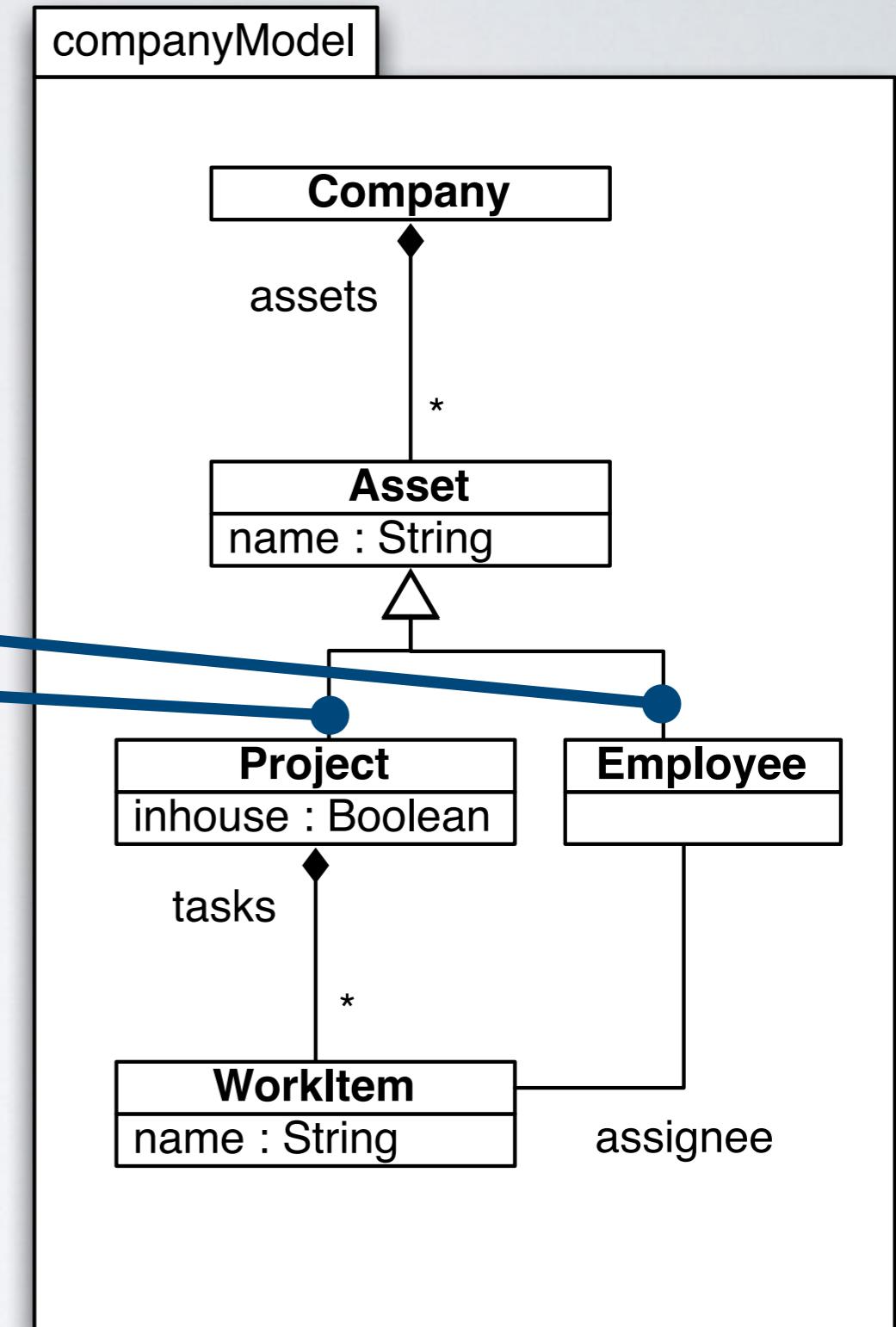
```
'employee' name=ID;
```

```
Project:
```

```
(inhouse?='inhouse')? 'project' name=ID '{'  
(task+=Task)*  
'}';
```

```
Task returns WorkItem:
```

```
'task' name=ID 'assigned to'  
assignee=[Employee];
```



```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals
```

```
generate companyModel
"http://www.xtext.org/workshop/CompanyModel"
```

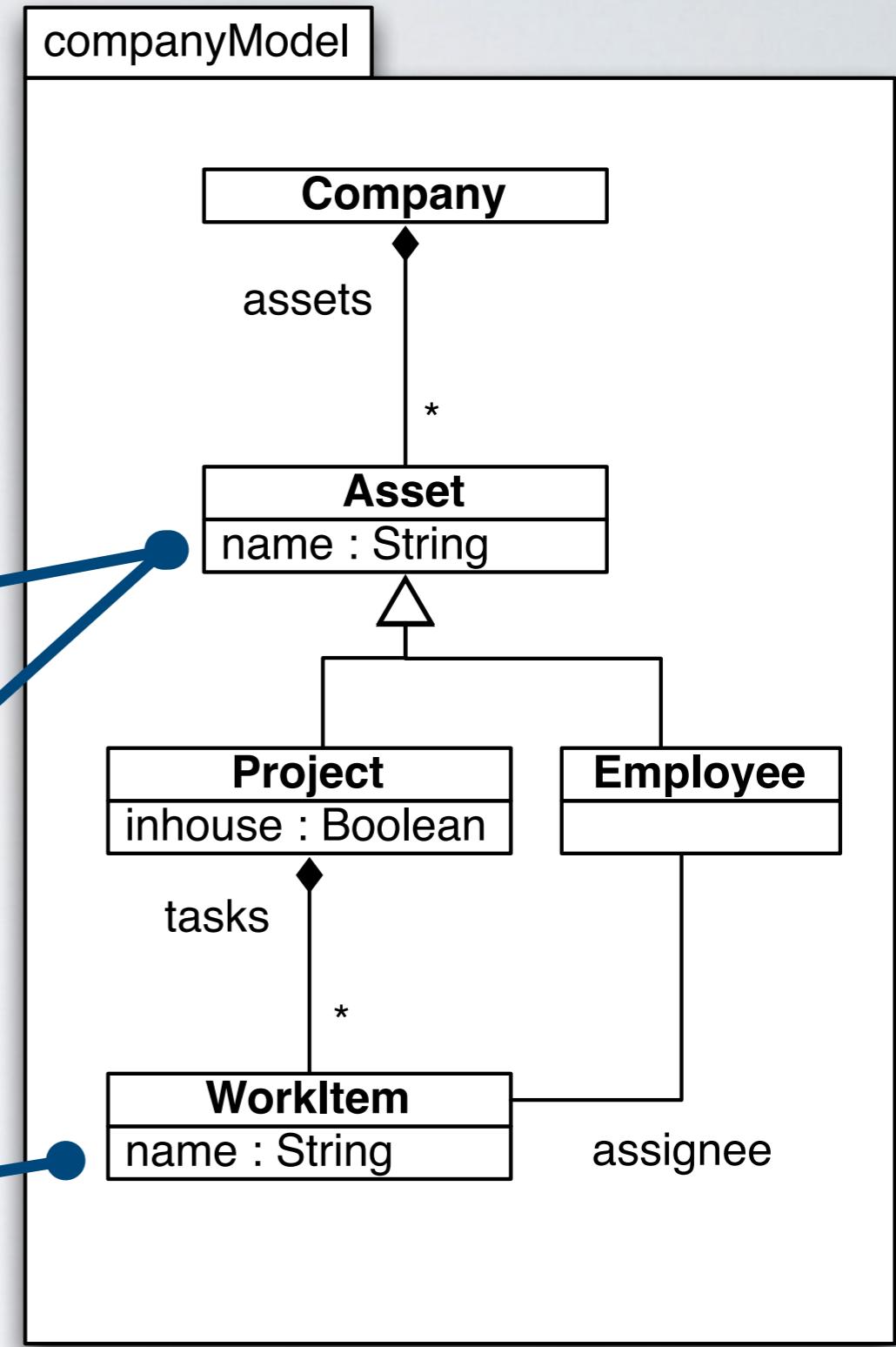
```
Company:
(assets+=Asset)+;
```

```
Asset:
Project | Employee;
```

```
Employee:
'employee' name=ID;
```

```
Project:
(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';
```

```
Task returns WorkItem:
'task' name=ID 'assigned to'
assignee=[Employee];
```



```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals
```

```
generate companyModel
"http://www.xtext.org/workshop/CompanyModel"
```

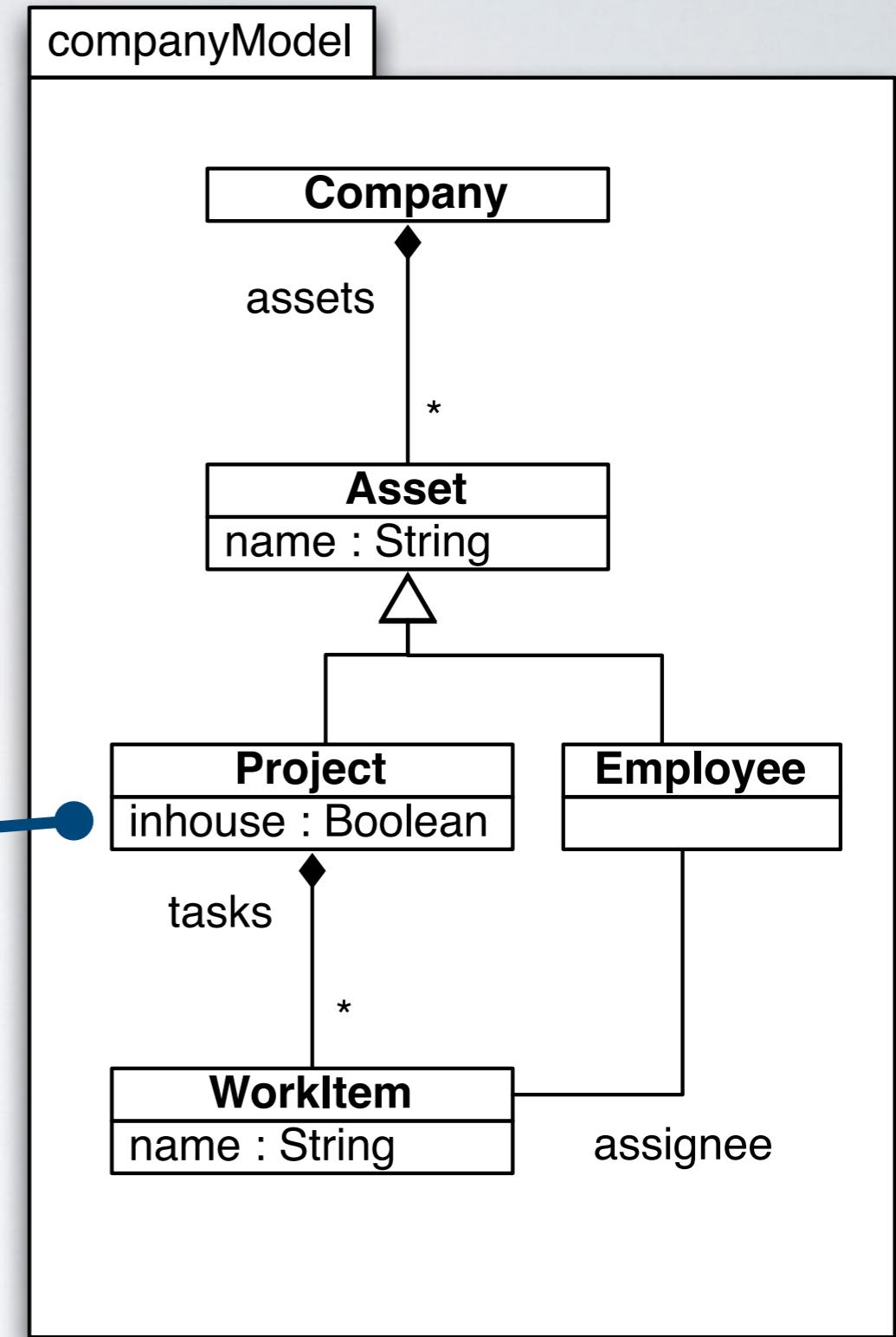
```
Company:
(assets+=Asset)+;
```

```
Asset:
Project | Employee;
```

```
Employee:
'employee' name=ID;
```

```
Project:
(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';
```

```
Task returns WorkItem:
'task' name=ID 'assigned to'
assignee=[Employee];
```



```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals
```

```
generate companyModel
"http://www.xtext.org/workshop/CompanyModel"
```

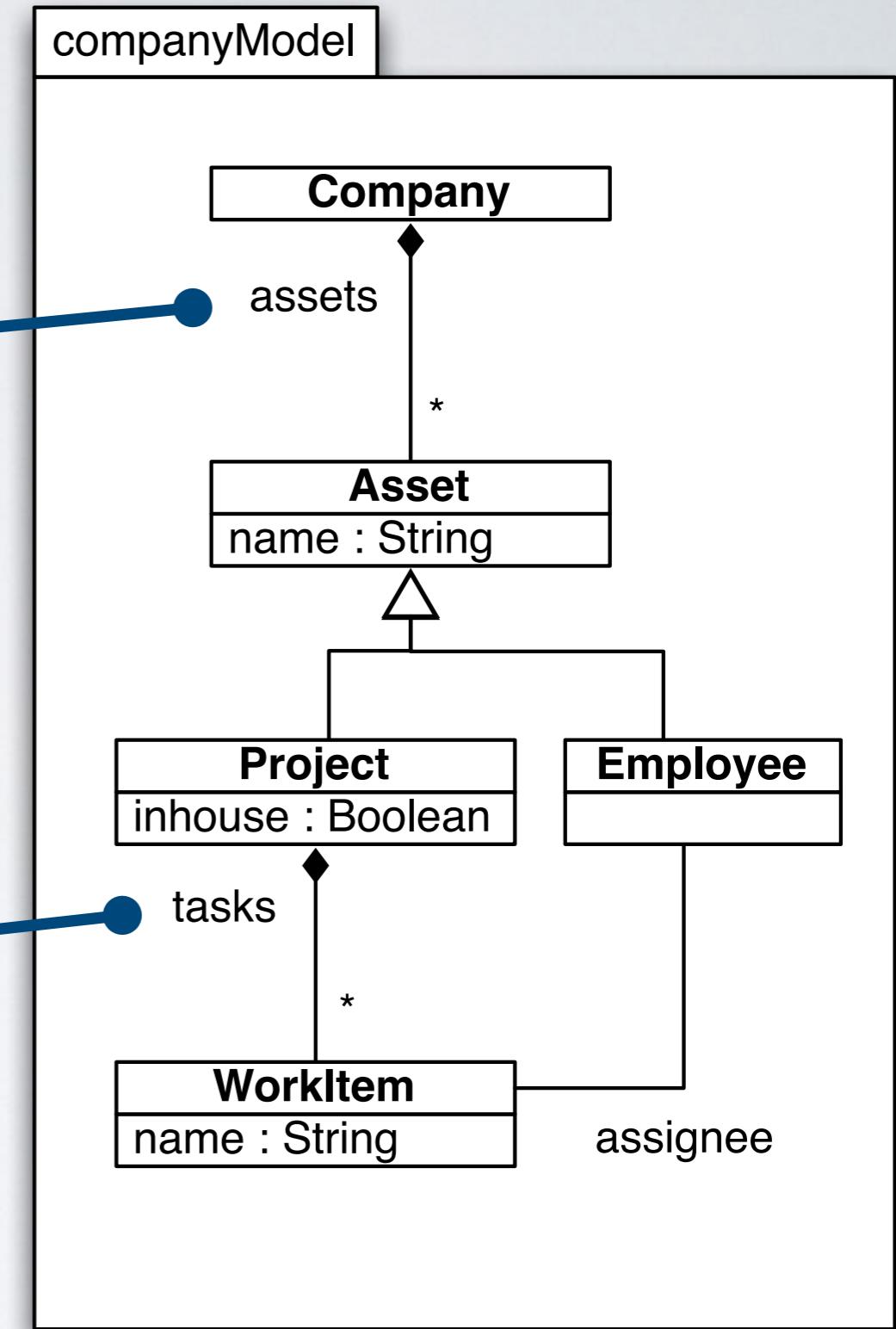
```
Company:
(assets+=Asset)+;
```

```
Asset:
Project | Employee;
```

```
Employee:
'employee' name=ID;
```

```
Project:
(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';
```

```
Task returns WorkItem:
'task' name=ID 'assigned to'
assignee=[Employee];
```



```
grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals
```

```
generate companyModel
"http://www.xtext.org/workshop/CompanyModel"
```

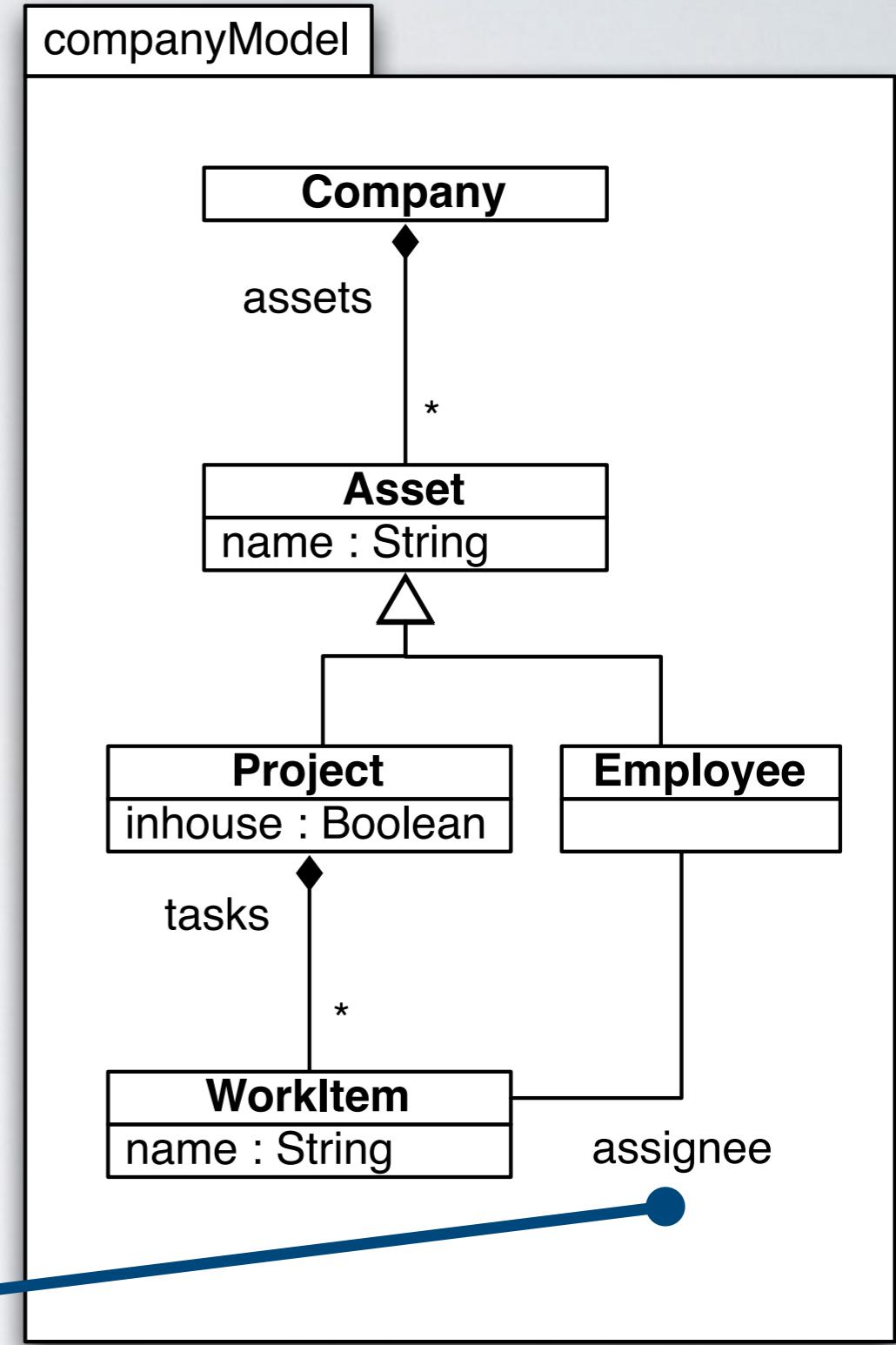
```
Company:
(assets+=Asset)+;
```

```
Asset:
Project | Employee;
```

```
Employee:
'employee' name=ID;
```

```
Project:
(inhouse?='inhouse')? 'project' name=ID '{'
(task+=Task)*
'}';
```

```
Task returns WorkItem:
'task' name=ID 'assigned to'
assignee=[Employee];
```



EXERCISE: GRAMMAR

Outline

- Example Application
- Build your DSL with Xtext
- **Generate Code with Xtend**
- Validate Models
- Cross-References and Scoping
- Advanced Grammar
- EMF / Ecore integration
- Index
- UI customizations
- Assigned Actions
- Testing
- Xbase
- Problem Solving
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

15:30 - 17:00



Hello Xtend

```
class Greeter {  
    def static void main(String... args) {  
        println('Hello EclipseCon')  
    }  
}
```

Templates

```
def example(List<String> elements) """  
    Usually a template consists mainly of text spanning  
    multiple lines.  
    If you want to evaluate an expression you have to write it  
    in french quotes «7*3*2». Code assist inserts a pair of  
    these.  
    You can also iterate a collection with FOR  
    «FOR element: elements»  
        Found another element: «element»  
    «ENDFOR»  
    For decisions there is the IF statement  
    «IF elements.empty»  
        no elements.  
    «ENDIF»  
    »»
```

Dispatch Methods

```
// Xtend code
def dispatch area(Circle c) {
    c.radius * c.radius * Math.PI
}
def dispatch area(Rectangle r) {
    r.width * r.height
}

def someCalculation(Object o) {
    area(o) // polymorphic call
}
```

Dispatch Methods

```
// Xtend code
def dispatch area(Circle c) {
    c.radius * c.radius * Math.PI
}
def dispatch area(Rectangle r) {
    r.width * r.height
}

def someCalculation(Object o) {
    area(o) // polymorphic call
}
```

```
// generated Java code (dispatcher)
public double area(final Object c) {
    if (c instanceof Circle) {
        return _area((Circle)c);
    } else if (c instanceof Rectangle) {
        return _area((Rectangle)c);
    } else {
        throw new IllegalArgumentException();
    }
}
```

EXERCISE: CODE GENERATOR

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- **Validate Models**
- Cross-References and Scoping
- Advanced Grammar
- EMF / Ecore integration
- Index
- UI customizations
- Assigned Actions
- Testing
- Xbase
- Problem Solving
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

15:30 - 17:00

Validator

```
class SurveyValidator extends AbstractSurveyValidator {  
    @Check  
    def textMustNotBeEmpty(Question question) {  
        if(question.getText().isEmpty()) {  
            error("Empty question is illegal",  
                  question,  
                  SurveyPackage.Literals.QUESTION__TEXT)  
        }  
    }  
}
```

EXERCISE: VALIDATOR

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate Models
- **Cross-References and Scoping**
 - Advanced Grammar
 - EMF / Ecore integration
 - Index
 - UI customizations
 - Assigned Actions
 - Testing
 - Xbase
 - Problem Solving
 - Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

15:30 - 17:00

Cross References

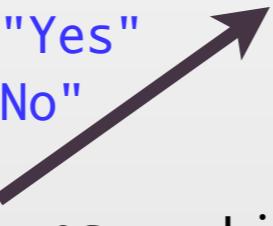
```
page Start (
    single choice like "Do you like the tutorial?" (
        yes "Yes"
        no "No"
    )
    if like=yes -> Like
)

page Like (
    choice particular "What do you like in particular?" (
        xtext 'Xtext is awesome'
        excercises 'The funny exercises'
        tutors 'The handsome tutors'
    )
)
```

Cross References

```
page Start (
    single choice like "Do you like the tutorial?" (
        yes "Yes"
        no "No"
    )
    if like=yes -> Like
)

page Like (
    choice particular "What do you like in particular?" (
        xtext 'Xtext is awesome'
        excercises 'The funny exercises'
        tutors 'The handsome tutors'
    )
)
```



Cross References

```
page Start (
    single choice like "Do you like the tutorial?" (
        yes "Yes"
        no "No"
        if like=yes -> Like
    )
)

page Like (
    choice particular "What do you like in particular?" (
        xtext 'Xtext is awesome'
        excercises 'The funny exercises'
        tutors 'The handsome tutors'
    )
)
```

The diagram illustrates the flow between two pages. A self-loop arrow originates from the 'Start' page and points back to it. Another arrow originates from the 'Start' page and points to the 'Like' page.

Cross References

```
page Start (
    single choice like "Do you like the tutorial?" (
        yes "Yes"
        no "No"
        if like=yes -> Like
    )
)
page Like (
    choice particular "What do you like in particular?" (
        xtext 'Xtext is awesome'
        excercises 'The funny exercises'
        tutors 'The handsome tutors'
    )
)
```

Grammar

Page:

```
'page' name=ID '('  
// questions  
'->' next=[Page|ID]  
)';
```

WHAT'S A SCOPE?

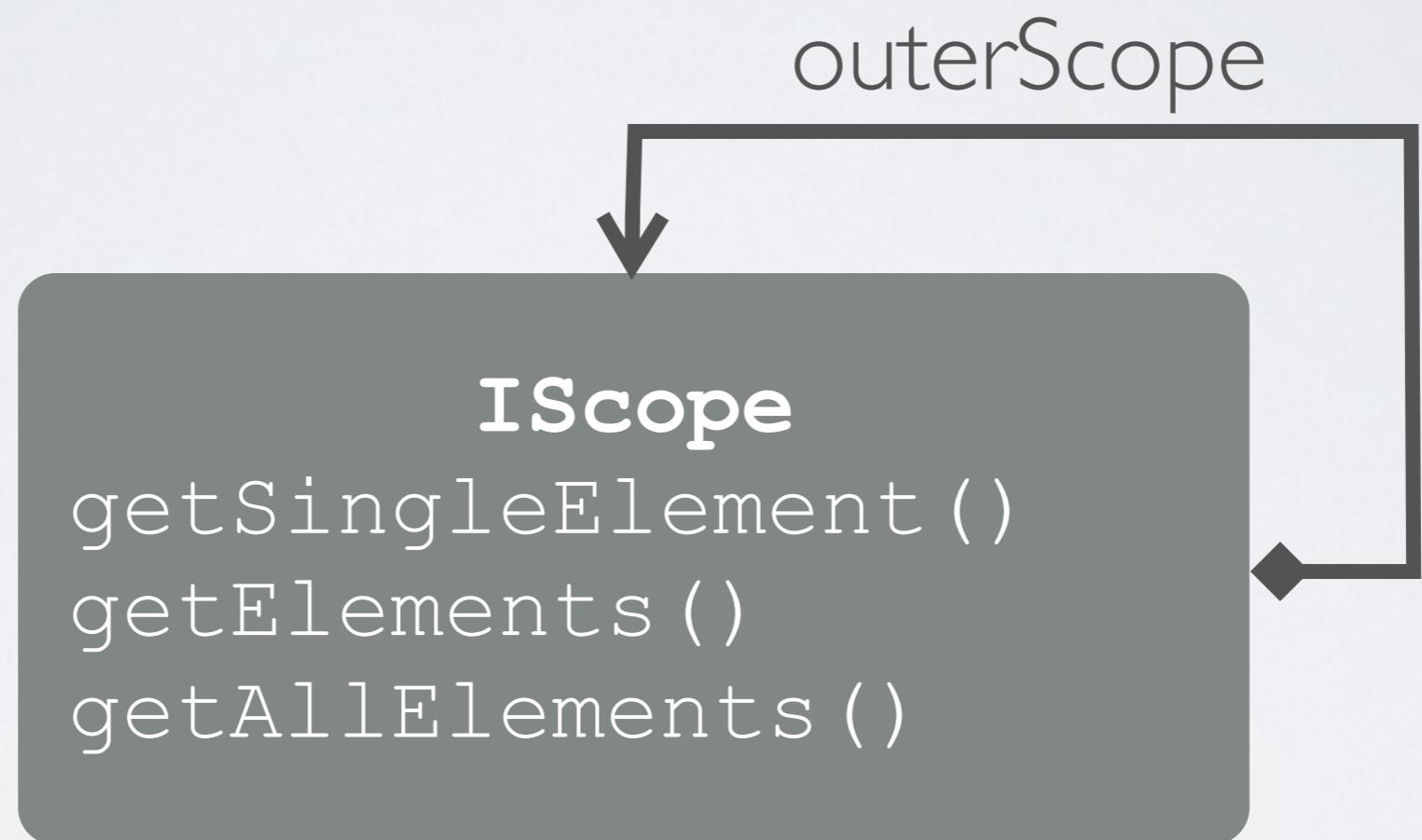
Describes the
visible Elements

Depends on
the context

Mapping:
Name to EObject

Chain of
Responsibility

WHAT'S A SCOPE?



CROSS REFERENCES

```
...  
<children xsi:type="application:PerspectiveStack"  
xmi:id="_NGiG4DGjEd-zge_czUaGVQ"  
id="_NGiG4DGjEd-zge_czUaGVQ"  
selectedElement="_NGjVADGjEd-zge_czUaGVQ">  
<children xsi:type="application:Perspective"  
xmi:id="_NGjVADGjEd-zge_czUaGVQ"  
id="_NGjVADGjEd-zge_czUaGVQ">  
...
```

UUIDs

Names

PerspectiveStack :myDslPerspectiveStack
selectedElement :myDslPerspective

Perspective :myDslPerspective

QUALIFIED NAMES

```
application contacts {  
    | command save "Save"  
}
```

Default
QualifiedNamesProvider

QUALIFIED NAMES

```
application contacts {  
    | command save "Save"  
    } }
```

qualifiedName(|)

Default
QualifiedNamespaceProvider

QUALIFIED NAMES

```
application contacts {  
    | command save "Save"  
}
```

EAttribute „name“

qualifiedName(|)

Default
QualifiedNameProvider

QUALIFIED NAMES

```
application contacts {  
    | command save "Save"  
    }  
}
```

EAttribute „name“

qualifiedName(|)

Default
QualifiedNameProvider

QUALIFIED NAMES

```
application contacts {  
    | command save "Save"  
    } }
```

qualifiedName(|)

Default
QualifiedNomeProvider

save

QUALIFIED NAMES

```
application contacts {  
    | command save "Save"  
}
```

EAttribute „name“

qualifiedName(|)

Default
Qualified Name Provider

save

QUALIFIED NAMES

```
application contacts {  
    | command save "Save"  
}
```

EAttribute „name“

↓
qualifiedName(|)

Default
QualifiedNomeProvider

save

QUALIFIED NAMES

```
application contacts {  
    | command save "Save"  
    } }
```

qualifiedName(|)

Default
QualifiedNomeProvider

contacts.save

OBJECTS CAN HAVE
MULTIPLE DIFFERENT
NAMES DEPENDING ON
THE CONTEXT

SCOPES

```
application contacts {  
    |  
    2 command save "Save"  
    |  
    3 command exit "Exit"  
    bind "CTRL+S" to save  
    bind "CTRL+X" to contacts.exit  
}
```

SCOPES

```
application contacts { |  
  command save "Save"  
  command exit "Exit" 3  
  bind "CTRL+S" to save  
  bind "CTRL+X" to contacts.exit  
}
```

2

3

SCOPES

```
application contacts {  
    |  
    2 command save "Save"  
    |  
    3 command exit "Exit"  
    bind "CTRL+S" to save  
    bind "CTRL+X" to contacts.exit  
}  
|
```

Container's Scope

name	EObject
„contacts“	1
„save“	2
„exit“	3

SCOPES

```
application contacts {  
    |  
    2 command save "Save"  
    |  
    3 command exit "Exit"  
    bind "CTRL+S" to save  
    bind "CTRL+X" to contacts.exit  
}  
|
```

Container's Scope

name	EObject
„contacts“ „save“ „exit“	 2 3

SCOPES

```
application contacts {  
    |  
    2 command save "Save"  
    |  
    command exit "Exit"  
    |  
    bind "CTRL+S" to 3 save  
    |  
    bind "CTRL+X" to contacts.exit  
}  
|
```

Container's Scope

name	EObject
„contacts“ „save“ „exit“	1 2 3

SCOPES

```
application contacts {  
    |  
    2 command save "Save"  
    |  
    3 command exit "Exit"  
    bind "CTRL+S" to save  
    bind "CTRL+X" to contacts.exit  
}  
|
```

Container's Scope

name	EObject
„contacts“	1
„save“	2
„exit“	3

SCOPES

```
application contacts {  
    |  
    2 command save "Save"  
    |  
    3 command exit "Exit"  
    bind "CTRL+S" to save  
    bind "CTRL+X" to contacts.exit  
}
```

Container's Scope

name	EObject
„contacts“	1
„save“	2
„exit“	3

SCOPES

```
application contacts {  
    |  
    2 command save "Save"  
    |  
    3 command exit "Exit"  
    bind "CTRL+S" to save  
    bind "CTRL+X" to contacts.exit  
}
```

Container's Scope

name	EObject
„contacts“	1
„save“	2
„exit“	3

outerScope →

SCOPES

```
application contacts {  
    |  
    2 command save "Save"  
    |  
    3 command exit "Exit"  
    bind "CTRL+S" to save  
    bind "CTRL+X" to contacts.exit  
}
```

Container's Scope

name	EObject
„contacts“	1
„save“	2
„exit“	3



File Scope

name	EObject
„contacts“	1
„contacts.save“	2
„contacts.exit“	3

SCOPES

```
application contacts {  
    |  
    2 command save "Save"  
    |  
    3 command exit "Exit"  
    bind "CTRL+S" to save  
    bind "CTRL+X" to contacts.exit  
}
```

Container's Scope

name	EObject
„contacts“	1
„save“	2
„exit“	3



File Scope

name	EObject
„contacts“	1
„contacts.save“	2
„contacts.exit“	3

SCOPES

```

application contacts {
    | 1
    command save "Save"
    | 2
    command exit "Exit" 3
    bind "CTRL+S" to save
    bind "CTRL+X" to contacts.exit
}

```

Container's Scope

name	EObject
„contacts“	1
„save“	2
„exit“	3



File Scope

name	EObject
„contacts“	1
„contacts.save“	2
„contacts.exit“	3

...

GlobalScope

outerScope
...

name	EObject
FQNs	all visible elements

GlobalScope

outerScope
...

name	EObject
FQNs	all visible elements

DefaultGlobalScopeProvider

GlobalScope

outerScope
...

name	EObject
FQNs	all visible elements

DefaultGlobalScopeProvider

- all in workspace

GlobalScope

outerScope
...

name	EObject
FQNs	all visible elements

DefaultGlobalScopeProvider

- all in workspace
- backed by an Eclipse builder

GlobalScope

outerScope
...

name	EObject
FQNs	all visible elements

DefaultGlobalScopeProvider

- all in workspace
- backed by an Eclipse builder

ResourceSetGlobalScopeProvider

GlobalScope

outerScope
...

name	EObject
FQNs	all visible elements

DefaultGlobalScopeProvider

- all in workspace
- backed by an Eclipse builder

ResourceSetGlobalScopeProvider

- all in ResourceSet

Scoping

```
page Start (
    text name 'Your name'
    single choice like '...' (
        yes 'Yes'
        no 'No'
    )
    if like=yes -> Like
)
```

```
page Like (
    choice particular '...' (
        xtext '...'
        excercises '...'
        tutors '...'
    )
...
)
```

Questions (Default Scoping)	
Global Name	Local Name
Start.name	name
Start.like	like
Like.particular	

Scoping

```
page Start (
    text name 'Your name'
    single choice like '...' (
        yes 'Yes'
        no 'No'
    )
    if like=yes -> Like
)
```

```
page Like (
    choice particular '...' (
        xtext '...'
        excercises '...'
        tutors '...'
    )
    ...
)
```

Questions (Default Scoping)	
Global Name	Local Name
Start.name	name
Start.like	like
Like.particular	

Choices
Customized
yes
no

EXERCISE: SCOPING

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate Models
- Cross-References and Scoping
- **Advanced Grammar**
- EMF / Ecore integration
- Index
- UI customizations
- Assigned Actions
- Testing
- Xbase
- Problem Solving
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

15:30 - 17:00

GRAMMAR LANGUAGE

- ADVANCED FEATURES

GRAMMAR LANGUAGE - ADVANCED FEATURES

```
terminal ID      : '^'?('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'_'|'0'..'9')*;  
terminal INT returns ecore::EInt: ('0'..'9')+;
```

GRAMMAR LANGUAGE - ADVANCED FEATURES

```
terminal ID      : '^'?('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'_'|'0'..'9')*;  
terminal INT returns ecore::EInt: ('0'..'9')+;
```

QualifiedName:

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

Double returns ecore::EDouble:

```
    INT ('.' INT)?;
```

GRAMMAR LANGUAGE - ADVANCED FEATURES

```
terminal ID      : '^'?('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'_'|'0'..'9')*;  
terminal INT returns ecore::EInt: ('0'..'9')+;
```

QualifiedName:

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

Double returns ecore::EDouble:

```
    INT ('.' INT)?;
```

enum Visibility:

```
    public="public" | private="private" | protected="protected";
```

GRAMMAR LANGUAGE - ADVANCED FEATURES

```
terminal ID      : '^'?('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'_'|'0'..'9')*;  
terminal INT returns ecore::EInt: ('0'..'9')+;
```

QualifiedName:

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

Double returns ecore::EDouble:

```
INT ('.' INT)?;
```

enum Visibility:

```
public | private | protected;
```

GRAMMAR LANGUAGE - ADVANCED FEATURES

```
terminal ID      : '^'?('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'_'|'0'..'9')*;  
terminal INT returns ecore::EInt: ('0'..'9')+;
```

QualifiedName:

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

Double returns ecore::EDouble:

```
INT ('.' INT)?;
```

enum Visibility:

```
public="public" | public="+" |  
private="private" | private="-" |  
protected="protected" | protected="#" ;
```

AST CONSTRUCTION

Class :

(abstract?='abstract')? 'class' name=ID ('extends' super=[Class])?;

Example I: abstract class A

AST CONSTRUCTION

Class :

(abstract?='abstract')? 'class' name=ID ('extends' super=[Class])?;

Example I:

abstract class A



First Feature assignment in rule

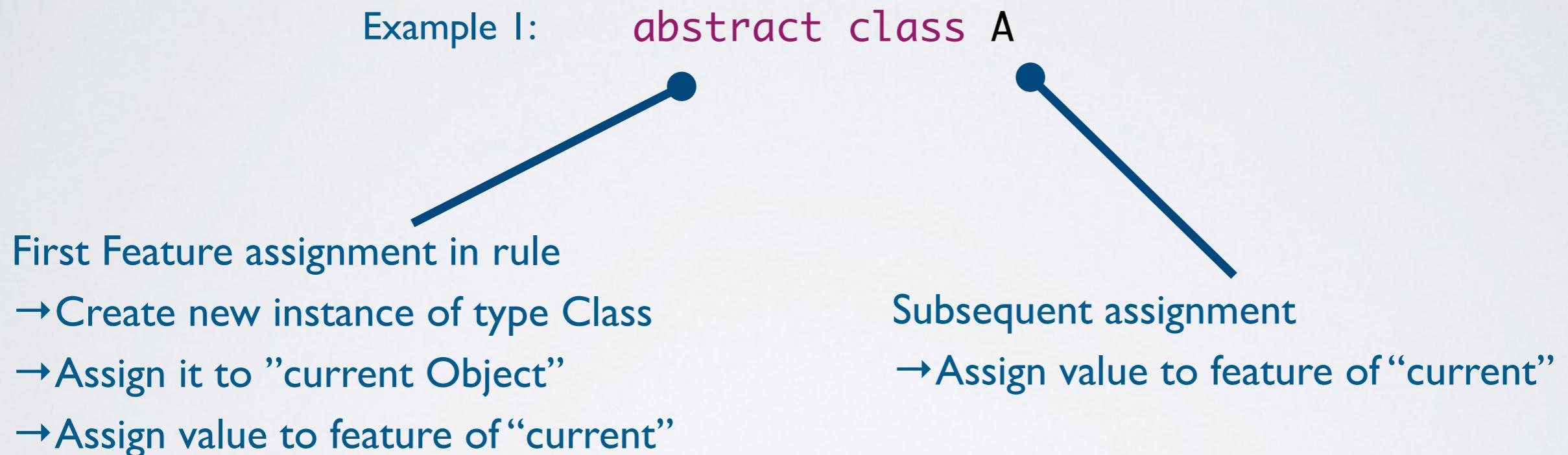
- Create new instance of type Class
- Assign it to "current Object"
- Assign value to feature of "current"

AST CONSTRUCTION

Class :

(abstract?='abstract')? 'class' name=ID ('extends' super=[Class])?;

Example I:



AST CONSTRUCTION

Class :

(abstract?='abstract')? 'class' name=ID ('extends' super=[Class])?;

Example 1:



First Feature assignment in rule

- Create new instance of type Class
- Assign it to "current Object"
- Assign value to feature of "current"

Subsequent assignment

- Assign value to feature of "current"

Example 2:

class B extends A

AST CONSTRUCTION

Class :

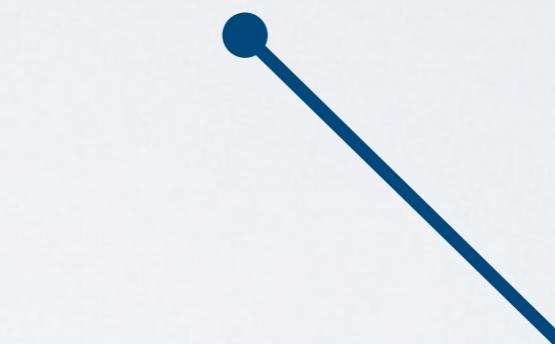
(abstract?='abstract')? 'class' name=ID ('extends' super=[Class])?;

Example 1:

abstract class A

First Feature assignment in rule

- Create new instance of type Class
- Assign it to "current Object"
- Assign value to feature of "current"



Subsequent assignment

- Assign value to feature of "current"

Example 2:

class B extends A



AST CONSTRUCTION

Class :

(abstract?='abstract')? 'class' name=ID ('extends' super=[Class])?;

Example 1:

abstract class A

First Feature assignment in rule

- Create new instance of type Class
- Assign it to "current Object"
- Assign value to feature of "current"

Subsequent assignment

- Assign value to feature of "current"

Example 2:

class B extends A

ACT CONSTRUCTION

```
Class CreateObject() {
    EObject current = null; // result
    if (token=='abstract') {
        if (current == null) current = new Class();
        current.setAbstract(true);
        advance();
    }
    advance('class');
    if (token == ID) {
        if (current == null) current = new Class();
        current.setName(tokenValue);
        advance();
    }
    ...
    return current;
}
```

First F
→Create
→Assi
→Assi

ass)?;

AST CONSTRUCTION II

Type :

(Class | DataType) 'is' visibility='public' | 'private');

DataType :

'datatype' name=ID;

Class :

'class' name=ID;

AST CONSTRUCTION II

Type :

(Class | DataType) 'is' visibility='public' | 'private');

DataType :

'datatype' name=ID;

Class :

'class' name=ID;

Example datatype A is public

AST CONSTRUCTION II

Type :

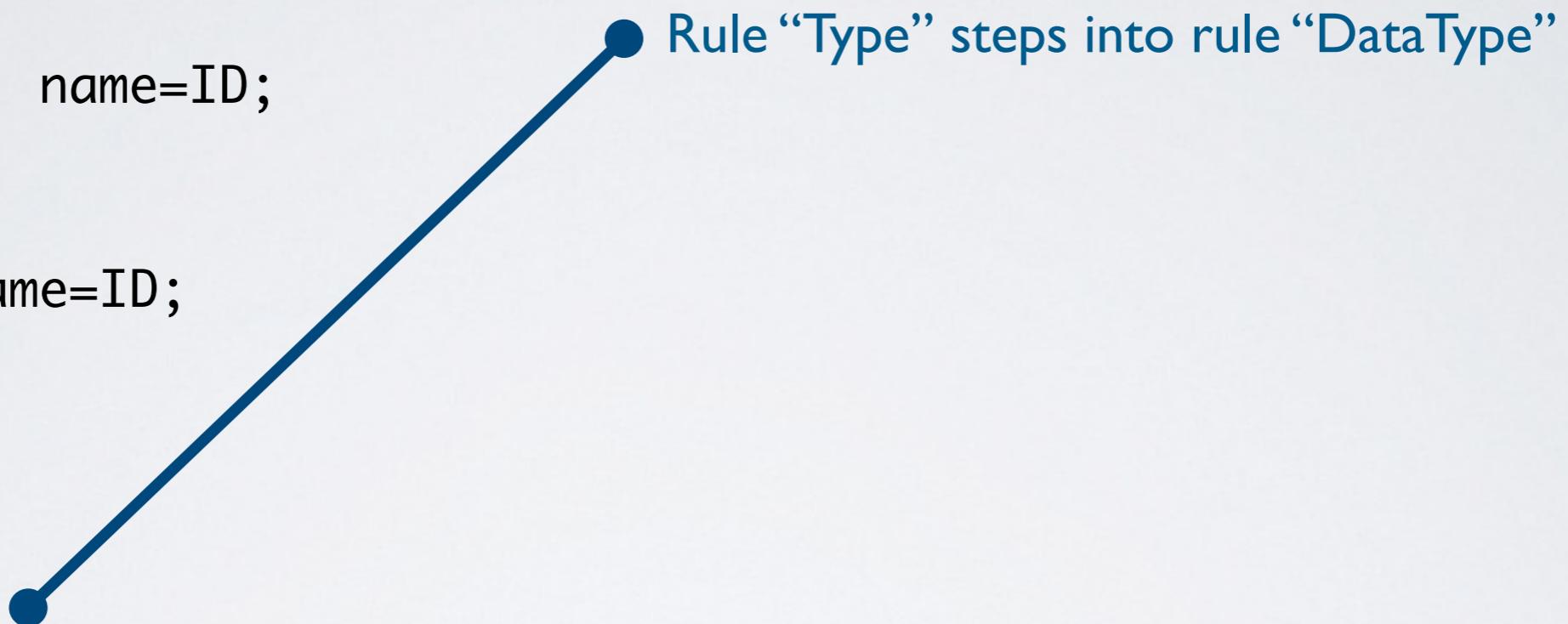
(Class | DataType) 'is' visibility='public' | 'private');

DataType :

'datatype' name=ID;

Class :

'class' name=ID;



Example datatype A is public

AST CONSTRUCTION II

Type :

```
(Class | DataType) 'is' visibility='public' | 'private');
```

DataType :

```
'datatype' name=ID;
```

Class :

```
'class' name=ID;
```

Example datatype A is public



AST CONSTRUCTION II

Type :

(Class | DataType) 'is' visibility='public' | 'private');

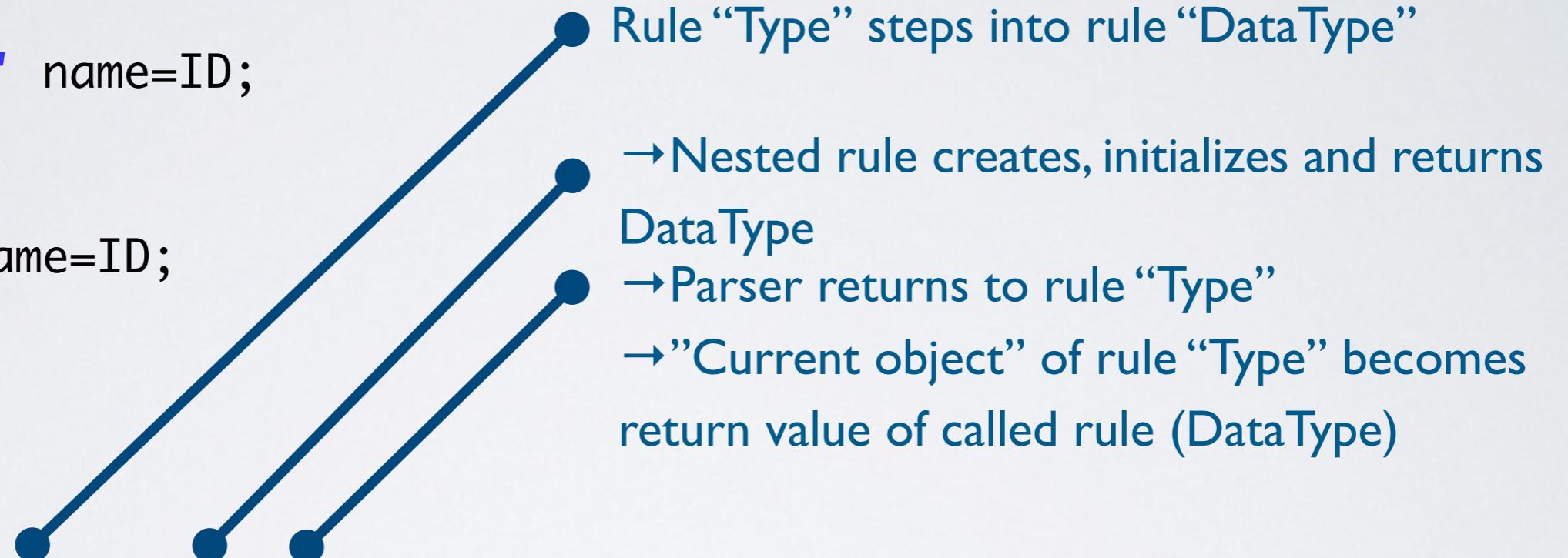
DataType :

'datatype' name=ID;

Class :

'class' name=ID;

Example datatype A is public



AST CONSTRUCTION II

Type :

```
(Class | DataType) 'is' visibility='public' | 'private');
```

DataType :

```
'datatype' name=ID;
```

Class :

```
'class' name=ID;
```

Example

datatype A is public



Rule “Type” steps into rule “DataType”

→Nested rule creates, initializes and returns
DataType
→Parser returns to rule “Type”
→“Current object” of rule “Type” becomes
return value of called rule (DataType)

→Subsequent assignments set
features of “current object”

AST CONSTRUCTION II

Type :

(Class | DataType) 'is' visibility='public' | 'private');

DataType :

'datatype' name=ID;

Class :

'class' name=ID;

Example

datatype A is public



Rule “Type” steps into rule “DataType”

→ Nested rule creates, initializes and returns
 DataType
→ Parser returns to rule “Type”
→ “Current object” of rule “Type” becomes
 return value of called rule (DataType)

→ Subsequent assignments set
 features of “current object”

- Rule “Type” returns either a Class or a DataType

- In this example only one instance has been created

ACT CONCRETE LEXICAL ANALYSIS

Type
CC
EOBJECT current = null; // result
Data
'd
if (token=='datatype') {
 current = parseDatatype();
} else { .. }
Class
'c
advance('is');
if (token == 'public' || token=='private') {
 if (current ==null) current = new Type();
 current.setVisibility(tokenValue);
 advance();
}
Example
...
return current;
• Rules
• Int

UNORDERED GROUPS

Employee:

```
'firstName' firstName=ID  
('age' age=INT)?  
'lastName' lastName=ID  
('yearOfBirth' yearOfBirth=INT)?  
;
```

UNORDERED GROUPS

Employee:

```
'firstName' firstName=ID  
('age' age=INT)?  
'lastName' lastName=ID  
('yearOfBirth' yearOfBirth=INT)?  
;
```

firstName	Holger
lastName	Schil
yearOfBirth	

UNORDERED GROUPS

Employee:

```
('firstName' firstName=ID  
('age' age=INT)?  
'lastName' lastName=ID  
('yearOfBirth' yearOfBirth=INT)?)*
```

;

UNORDERED GROUPS

Employee:

```
('firstName' firstName=ID  
('age' age=INT)?  
'lastName' lastName=ID  
('yearOfBirth' yearOfBirth=INT)?)*  
;
```

```
5 Employee:  
6   ('firstName' firstName=ID  
7   ('age' age=INT)?  
8   'lastName' lastName=ID  
9   ('yearOfBirth' yearOfBirth=INT)?)*  
10 ;
```

The assigned value of feature 'firstName' will possibly override itself because it is used inside of a loop.

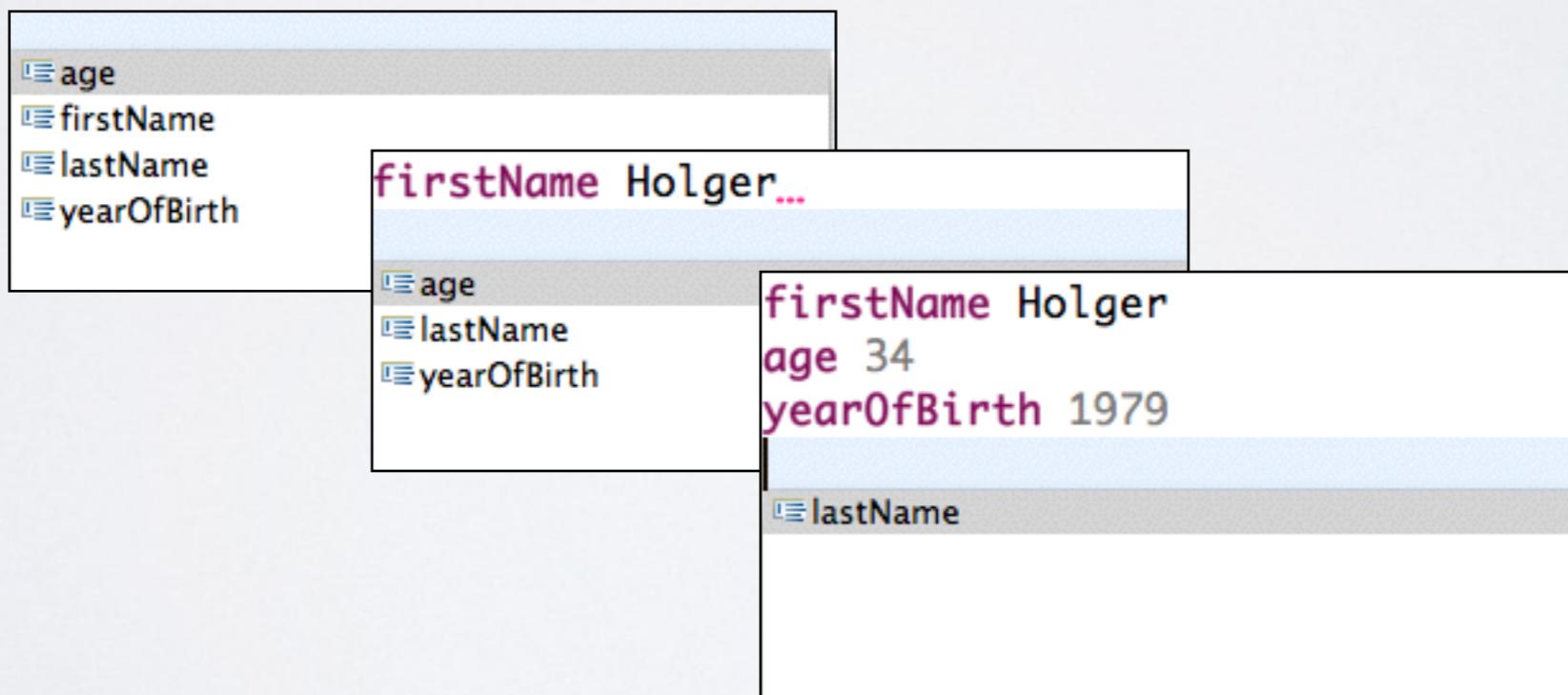
...

UNORDERED GROUPS

Employee:

```
(,firstName' firstName=ID &  
('age' age=INT)? &  
'lastName' lastName=ID &  
('yearOfBirth' yearOfBirth=INT)?)  
;
```

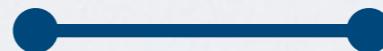
Each element can
only appear once but
in any order



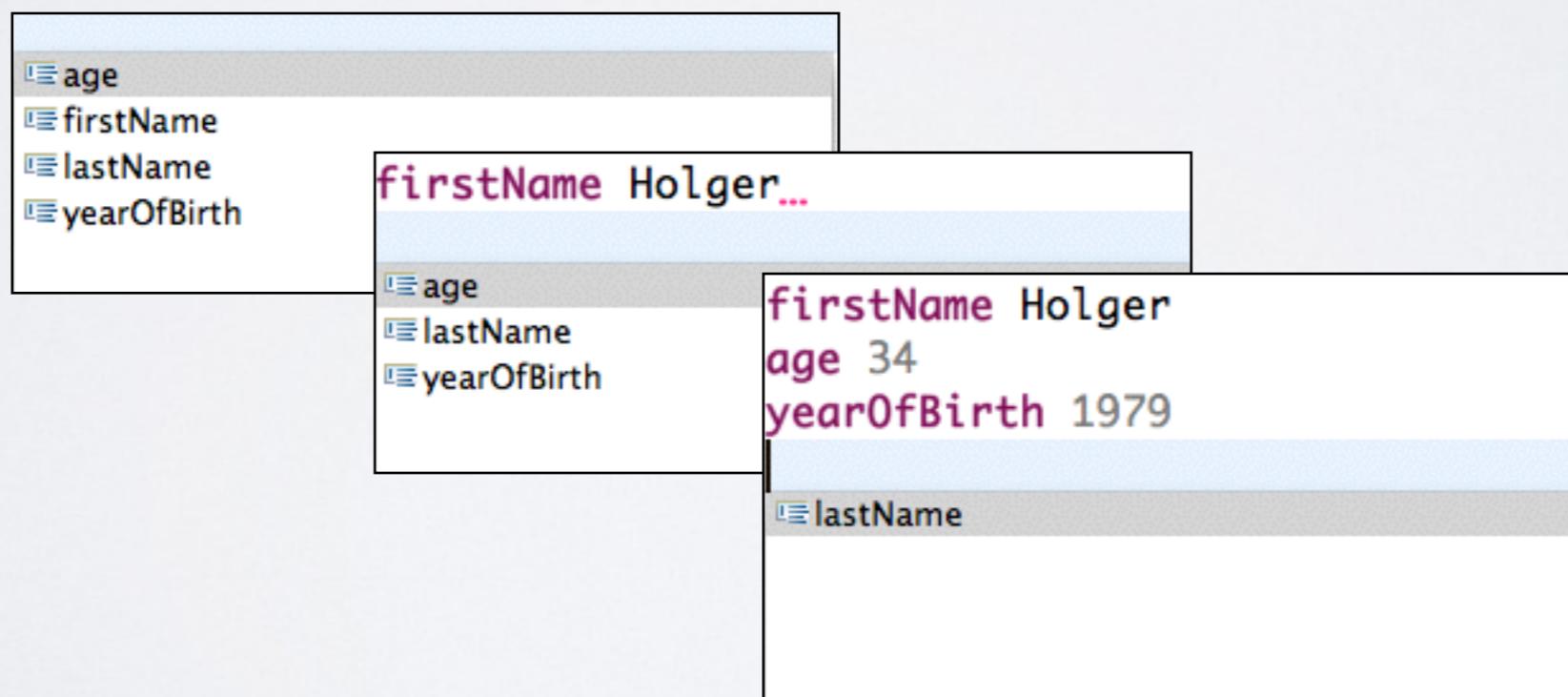
UNORDERED GROUPS

Employee:

```
(,firstName' firstName=ID &  
('age' age=INT)? &  
'lastName' lastName=ID &  
('yearOfBirth' yearOfBirth=INT)?)  
;
```



Each element can
only appear once but
in any order



Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate Models
- Cross-References and Scoping
- Advanced Grammar
- **EMF / Ecore integration**
- Index
- UI customizations
- Assigned Actions
- Testing
- Xbase
- Problem Solving
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

15:30 - 17:00

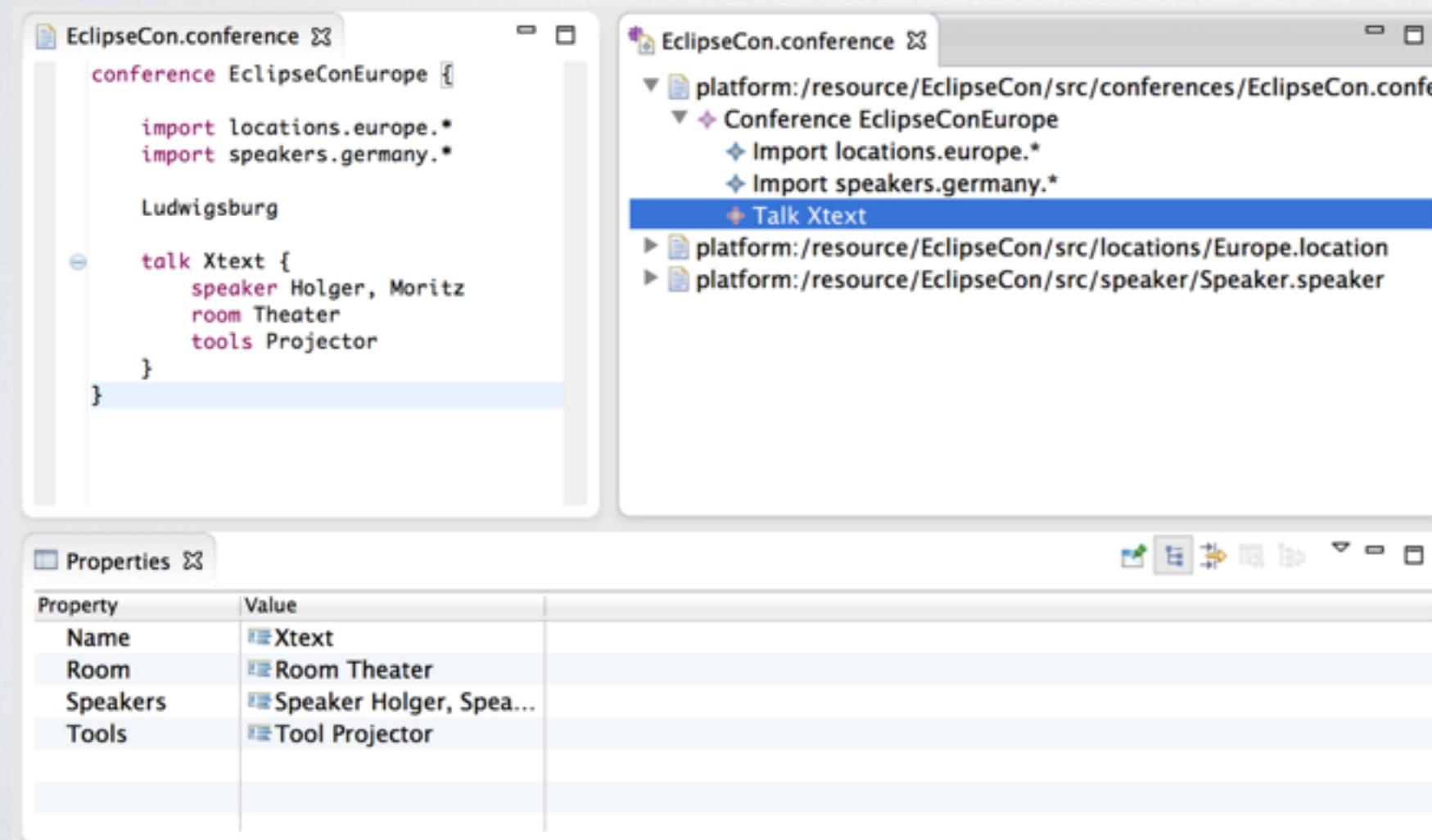
EMF INTEGRATION

EMF ECLIPSE MODELING ECOSYSTEM

your text files are EMF models!

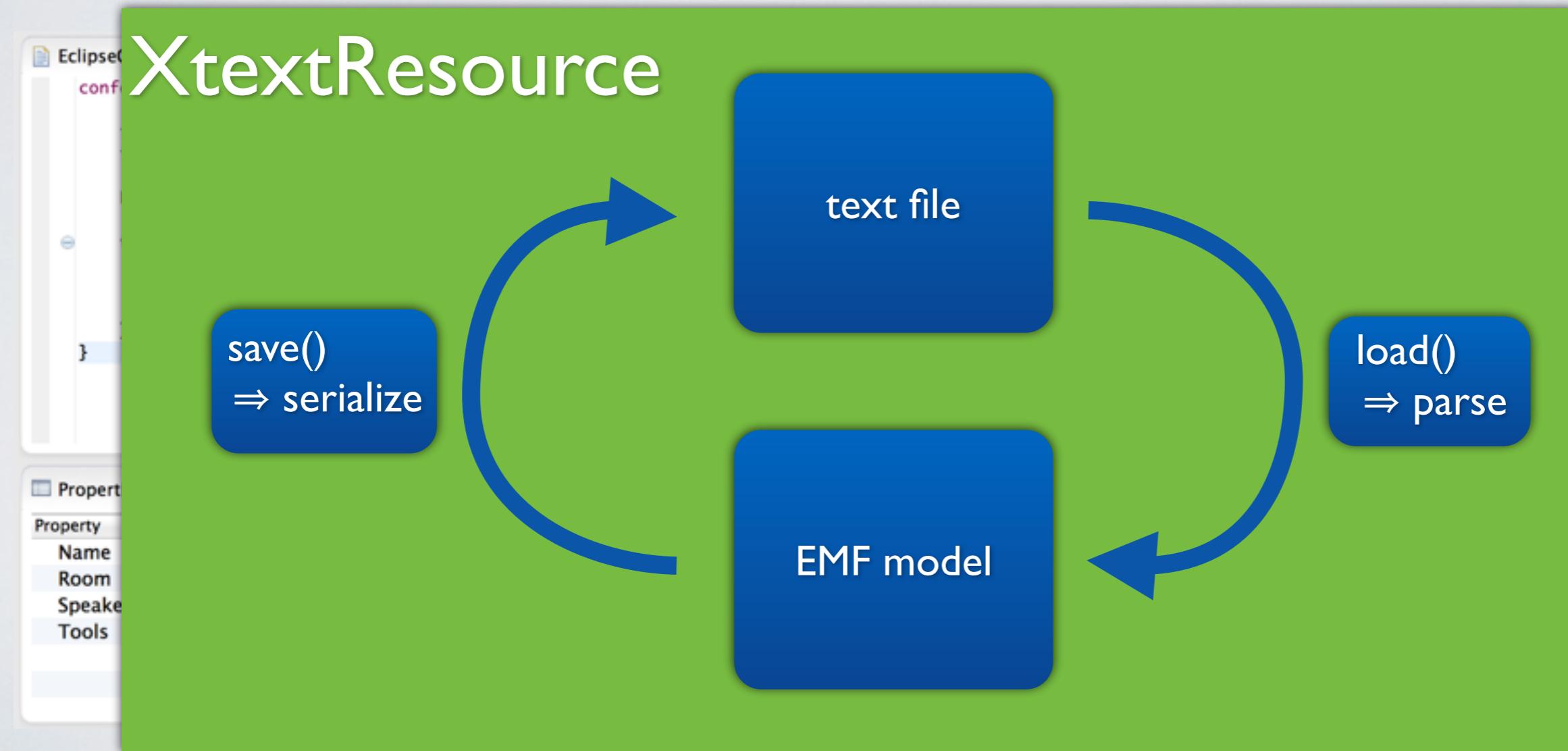
EMF ECLIPSE MODELING ECOSYSTEM

your text files are EMF models!



EMF ECLIPSE MODELING ECOSYSTEM

your text files are EMF models!

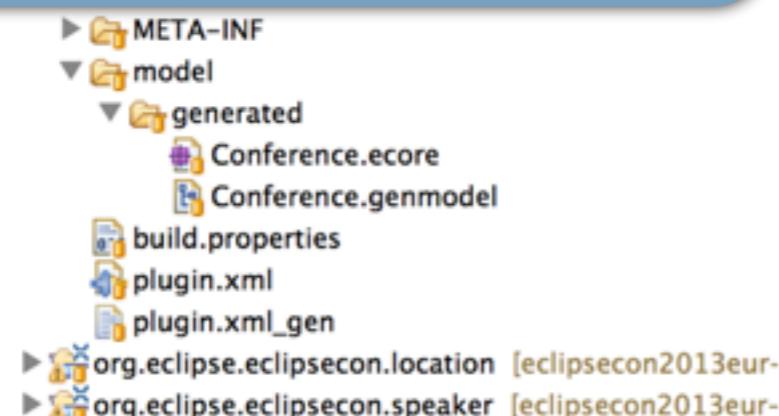


EMF ECLIPSE MODELING ECOSYSTEM

generate vs. import Ecore model

Generated Ecore Model

- rapid prototyping
- no sync issues
- not all Ecore features



The screenshot shows the Eclipse IDE's Xtext editor window titled 'Conference.xtext'. The code in the editor is:

```
grammar org.eclipse.eclipsecon.Conference with org.eclipse.xtext.common.Terminals

generate conference "http://www.eclipse.org/eclipsecon/Conference"

import "http://www.eclipse.org/eclipsecon/Location"
import "http://www.eclipse.org/eclipsecon/Speaker"
```

A blue callout bubble points from this section towards the 'Imported Ecore Model' text.

Imported Ecore Model

- fine-grained control
- manual sync
- all Ecore features
- use Xcore!

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate Models
- Cross-References and Scoping
- Advanced Grammar
- EMF / Ecore integration
- **Index**
- UI customizations
- Assigned Actions
- Testing
- Xbase
- Problem Solving
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

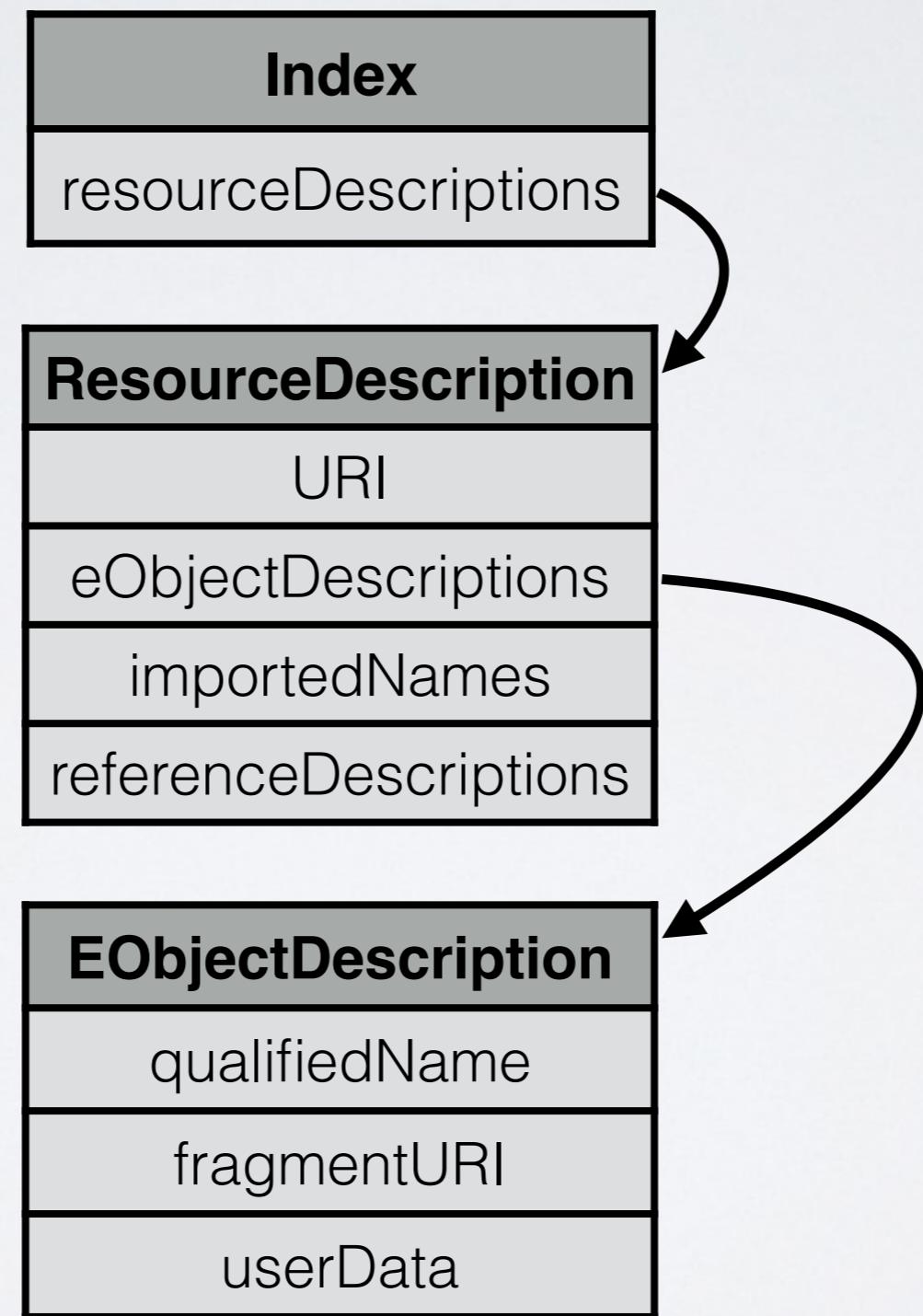
13:30 - 15:00

15:30 - 17:00

INDEX?

- enables filename-agnostic resolution of names
- enables validation of cross-references without loading targets
- enables computation of affected files during incremental build
- reverse-reference lookup

INDEX STRUCTURE



INDEX POPULATION

resource change

<triggers>

incremental Build

```
for(resource : changedOrAffected) {  
    val description = new ResourceDescription(resource.uri)  
    for(eObject : resource.eAllContents) {  
        val name = qualifiedNameProvider.get(eObject)  
        if(name != null) {  
            val fragment = resource.getFragmentURI(eObject)  
            description.add(new EObjectDescription(name, fragment))  
        }  
    }  
    index.add(description)  
}  
for(resource : changedOrAffected) {  
    EcoreUtil.resolveAll(resource)  
    validator.validate(resource)  
}
```

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate Models
- Cross-References and Scoping
- Advanced Grammar
- EMF / Ecore integration
- Index
- **UI customizations**
- Assigned Actions
- Testing
- Xbase
- Problem Solving
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

15:30 - 17:00

CODE COMPLETION

From Grammar

Survey:

```
'survey'  
name=ID  
title=STRING  
pages+=Page*;
```

Page:

```
'page' name=ID '('  
    questions+=Question*  
    followUps+=FollowUp*  
)';
```

From Grammar

```
Survey:  
  'survey'  
  name=ID  
  title=STRING  
  pages+=Page*;  
  
Page:  
  'page' name=ID '{'  
  questions+=Question*  
  followUps+=FollowUp*  
  '}';
```

Keywords

From Grammar

```
Survey:  
  'survey'  
  name=ID  
  title=STRING  
  pages+=Page*;  
  
Page:  
  'page' name=ID '{'  
  questions+=Question*  
  followUps+=FollowUp*  
  '}';
```

Keywords

From Scope

```
FollowUp:  
  guard=Guard? '-'> next=[Page];  
  
Guard:  
  'if'  
  question=[ChoiceQuestion|QualifiedName]  
  '='  
  answer=[Choice];
```

```
class SurveyProposalProvider extends AbstractSurveyProposalProvider {
```

CO
NEXUS

TEMPLATE PROPOSALS

FORMATTING

FORMATTING

Pretty Printing
Code Beautification

```
import
<Literature.entities> entity
City {Int id; String
name;}
entity Store { Int
id; String name;
Int
city
references
City.
id; }
```

FORMATTING

Pretty Printing
Code Beautification

```
import
<Literature.entities> entity
City {Int id; String
name;}
entity Store { Int
id; String name;
Int
city
references
City.
id; }
```

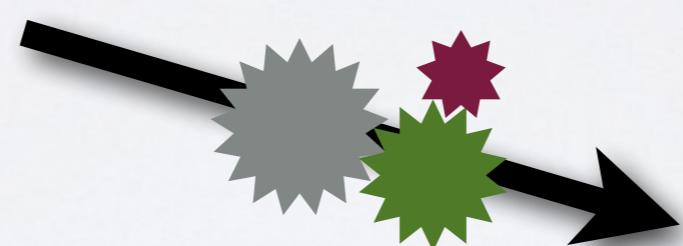
```
import <Literature.entities>
entity City {
    Int id;
    String name;
}

entity Store {
    Int id;
    String name;
    Int city references
    City.id;
}
```

FORMATTING

Pretty Printing
Code Beautification

```
import
<Literature.entities> entity
City {Int id; String
name;}
entity Store { Int
id; String name;
Int
city
references
City.
id; }
```



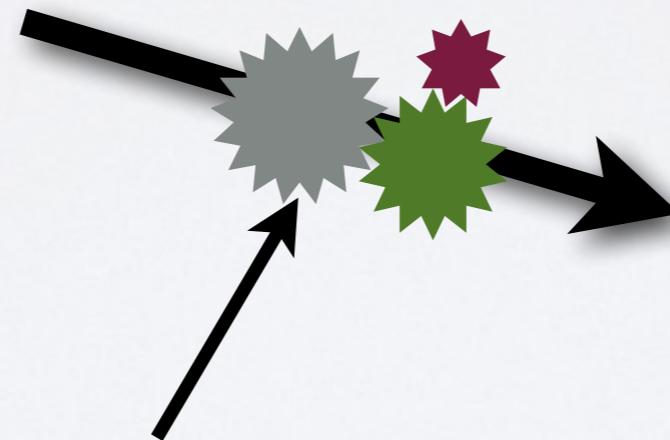
```
import <Literature.entities>
entity City {
    Int id;
    String name;
}

entity Store {
    Int id;
    String name;
    Int city references
    City.id;
}
```

FORMATTING

Pretty Printing
Code Beautification

```
import
<Literature.entities> entity
City {Int id; String
name;}
entity Store { Int
id; String name;
Int
city
references
City.
id; }
```



§ 1. At ... do ...
§ 2. At ... do ...
§ 3. At ... do ...
§ 4. At ... do ...
...

```
import <Literature.entities>

entity City {
    Int id;
    String name;
}

entity Store {
    Int id;
    String name;
    Int city references
    City.id;
}
```

TWO IMPLEMENTATIONS

`org.eclipse.xtext.formatting` // old

- will be deprecated soon
- rule based

`org.eclipse.xtext.formatting2` // new

- access to AST
- creates text replacements
- format tokens of Terminal/Datatype rules
- conditional formatting
- pattern-aware formatting
- column-based or tabular formatting

FORMATTING V2

PackageDeclaration:

```
'package' name=QualifiedName '{'  
    elements+=AbstractElement*  
'}';
```

Grammar

```
def dispatch void format(PackageDeclaration pkg, extension IFormattableDocument doc) {  
    pkg.regionForFeature(ABSTRACT_ELEMENT__NAME).surround[oneSpace]  
    pkg.regionForKeyword("{").append[newLine; increaseIndentation]  
    for (AbstractElement element : pkg.elements) {  
        format(element, doc);  
        element.append[setNewLines(1, 1, 2)]  
    }  
    pkg.regionForKeyword("}").prepend[decreaseIndentation]  
}
```

Formatter

QUICK FIX

Validation

```
public static final String INVALID_NAME = "org.xtext.quickfix.InvalidTypeName";  
  
@Check  
def void checkTypeNameStartsWithCapital(City city) {  
    if (city.getName() == null || city.getName().length() == 0) return;  
    if (!Character.isUpperCase(city.getName().charAt(0))) {  
        warning("Name should start with a capital letter.",  
               ExampleQuickfixPackage.Literals.CITY__NAME, INVALID_NAME,  
               city.getName());  
    }  
}
```

Fix

```
@Fix(ExampleQuickfixValidator.INVALID_NAME)  
def void capitalizeName(Issue issue, IssueResolutionAcceptor acceptor) {  
    acceptor.accept(issue, "Capitalize name",  
                  "Capitalize the name \\""\n// retrieve the 'user data' from the validation warning  
+ issue.getData().head + "\".", // icon to display (in the icons folder)  
                  "upcase.png",  
                  [context |  
                   val xtextDocument = context.getXtextDocument();  
                   val firstLetter = xtextDocument.get(issue.getOffset() + 1, 1);  
                   xtextDocument.replace(issue.getOffset() + 1, 1, firstLetter.toUpperCase());  
                  ]);  
}
```

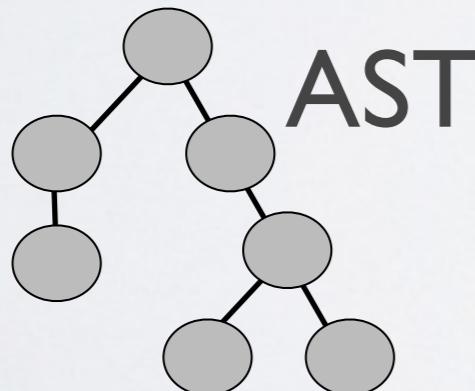
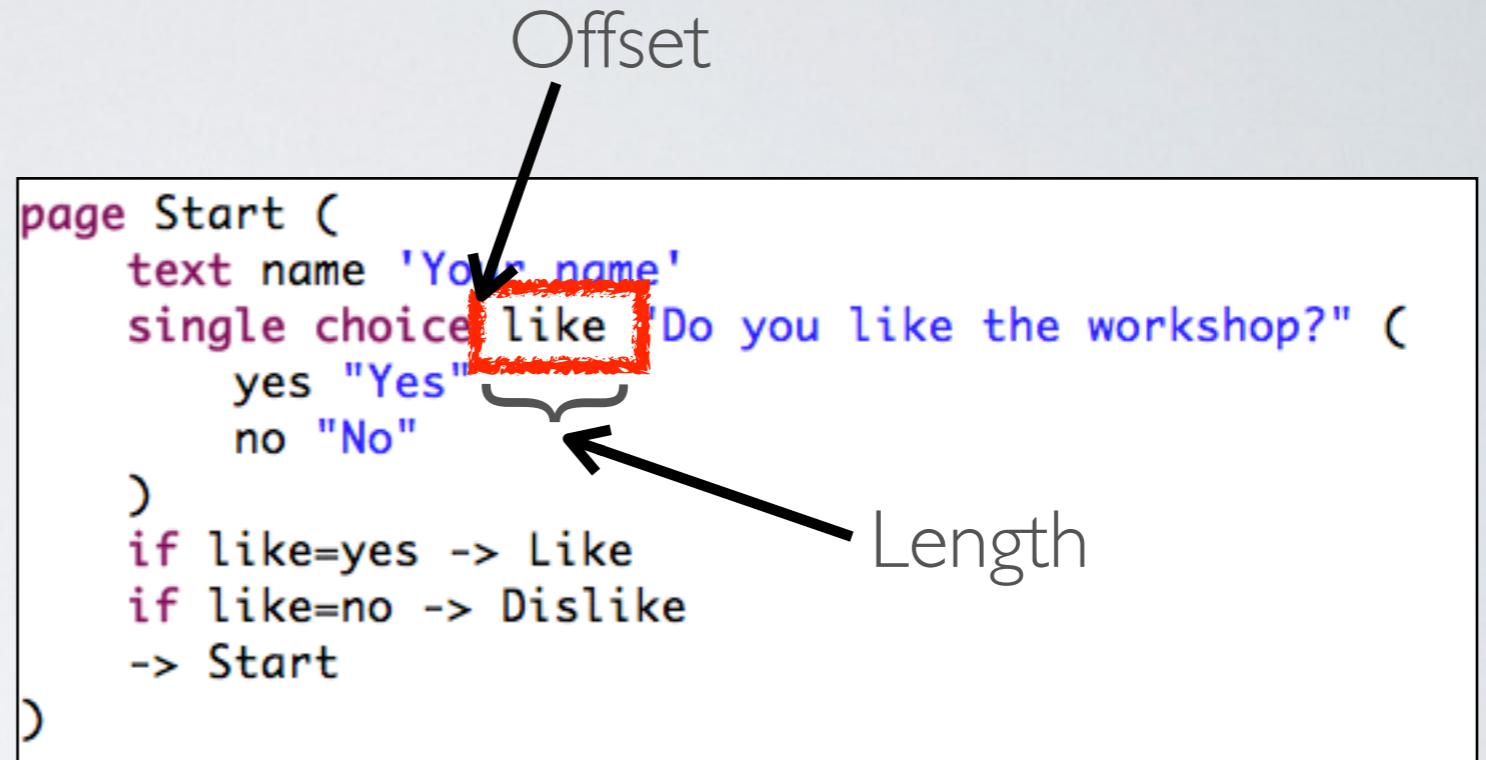
Different types of quick fixes

On a textual level

Offset

```
page Start (
    text name 'Your name'
    single choice like "Do you like the workshop?" (
        yes "Yes"
        no "No"
    )
    if like=yes -> Like
    if like=no -> Dislike
    -> Start
)
```

Length



On AST level

SYNTAX HIGHLIGHTING

Demo

OUTLINEVIEW

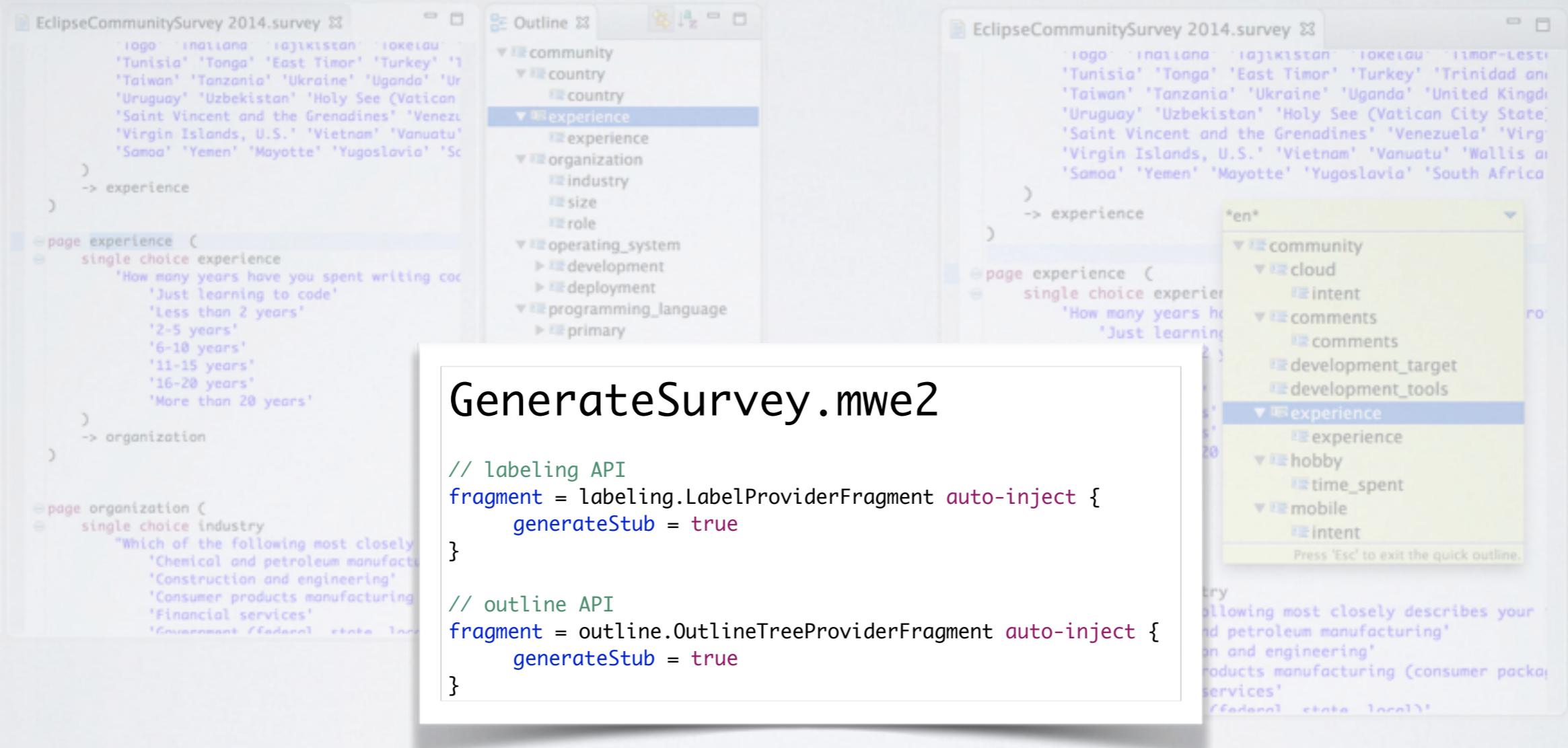
OUTLINE VIEW

The screenshot shows the Eclipse Community Survey 2014 survey editor interface. On the left, the survey structure is displayed in a tree view:

- Logo
- country
 - Tunisia
 - Tonga
 - East Timor
 - Turkey
 - Taiwan
 - Tanzania
 - Ukraine
 - Uganda
 - Uruguay
 - Uzbekistan
 - Holy See (Vatican City State)
 - Saint Vincent and the Grenadines
 - Venezuela
 - Virgin Islands, U.S.
 - Vietnam
 - Vanuatu
 - Samoa
 - Yemen
 - Mayotte
 - Yugoslavia
 - South Africa
- experience
 - experience
 - organization
 - industry
 - size
 - role
 - operating_system
 - development
 - deployment
 - programming_language
 - primary
 - others
 - development_target
 - development_tools
 - cloud
 - cloud_2
 - mobile
 - mobile_2
 - open_source
 - hobby
 - hobby_2
 - eclipse
 - comments

- page experience (single choice experience)
- How many years have you spent writing code?
- Just learning to code
- Less than 2 years
- 2-5 years
- 6-10 years
- 11-15 years
- 16-20 years
- More than 20 years
- organization
- single choice industry
 - Which of the following most closely describes your industry?
 - Chemical and petroleum manufacturing
 - Construction and engineering
 - Consumer products manufacturing (consumer packaged goods)
 - Financial services
 - Government (federal, state, local)

OUTLINEVIEW



org.eclipse.xtext.tutorial.survey.ui.outline.SurveyOutlineTreeProvider
org.eclipse.xtext.tutorial.survey.ui.labeling.SurveyLabelProvider

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate Models
- Cross-References and Scoping
- Advanced Grammar
- EMF / Ecore integration
- Index
- UI customizations
- **Assigned Actions**
- Testing
- Xbase
- Problem Solving
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

15:30 - 17:00

RECAP: AST CONSTRUCTION

Class :

(abstract?='abstract')? 'class' name=ID ('extends' super=[Class])?;

Example I: **abstract class A**

RECAP: AST CONSTRUCTION

Class :

(abstract?='abstract')? 'class' name=ID ('extends' super=[Class])?;

Example I:

abstract class A



First Feature assignment in rule

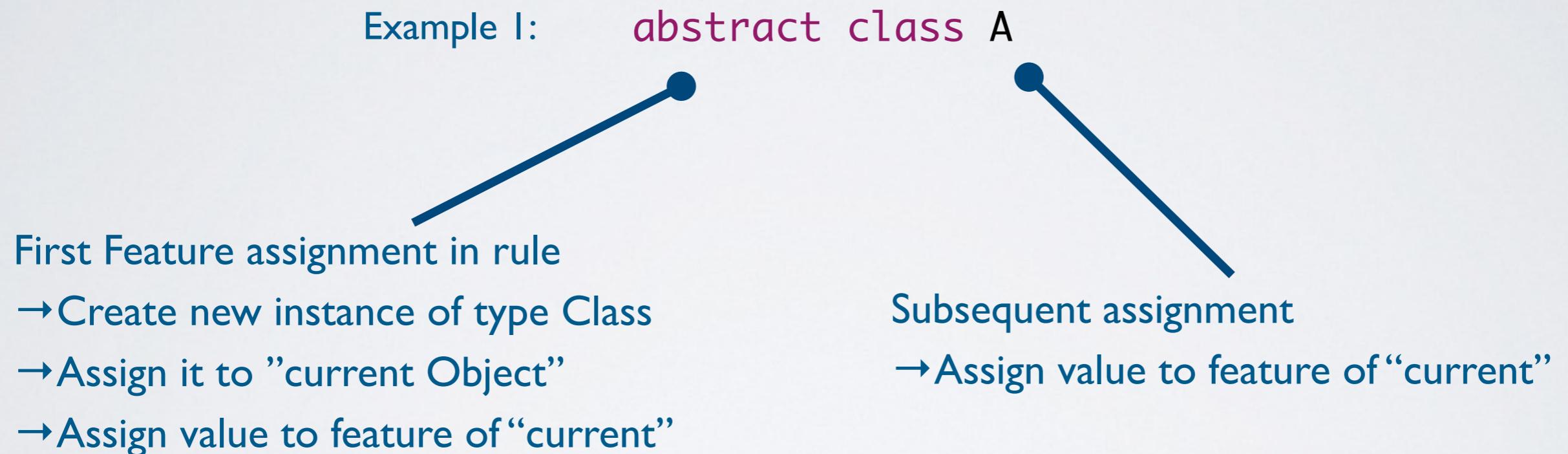
- Create new instance of type Class
- Assign it to "current Object"
- Assign value to feature of "current"

RECAP: AST CONSTRUCTION

Class :

(abstract?='abstract')? 'class' name=ID ('extends' super=[Class])?;

Example I:



RECAP: AST CONSTRUCTION

Class :

(abstract?='abstract')? 'class' name=ID ('extends' super=[Class])?;

Example 1:



First Feature assignment in rule

- Create new instance of type Class
- Assign it to "current Object"
- Assign value to feature of "current"

Subsequent assignment

- Assign value to feature of "current"

Example 2:

class B extends A

RECAP: AST CONSTRUCTION

Class :

(abstract?='abstract')? 'class' name=ID ('extends' super=[Class])?;

Example 1:

abstract class A

First Feature assignment in rule

- Create new instance of type Class
- Assign it to "current Object"
- Assign value to feature of "current"

Subsequent assignment

- Assign value to feature of "current"

Example 2:

class B extends A

RECAP: AST CONSTRUCTION

Class :

(abstract?='abstract')? 'class' name=ID ('extends' super=[Class])?;

Example 1:

abstract class A

First Feature assignment in rule

- Create new instance of type Class
- Assign it to "current Object"
- Assign value to feature of "current"

Subsequent assignment

- Assign value to feature of "current"

Example 2:

class B extends A

RECAP: AST CONSTRUCTION II

Type :

(Class | DataType) 'is' visibility='public' | 'private');

DataType :

'datatype' name=ID;

Class :

'class' name=ID;

RECAP: AST CONSTRUCTION II

Type :

(Class | DataType) 'is' visibility='public' | 'private');

DataType :

'datatype' name=ID;

Class :

'class' name=ID;

Example datatype A is public

RECAP: AST CONSTRUCTION II

Type :

```
(Class | DataType) 'is' visibility='public' | 'private');
```

DataType :

```
'datatype' name=ID;
```

Class :

```
'class' name=ID;
```



Rule “Type” steps into rule “DataType”



Example datatype A is public

RECAP: AST CONSTRUCTION II

Type :

```
(Class | DataType) 'is' visibility='public' | 'private');
```

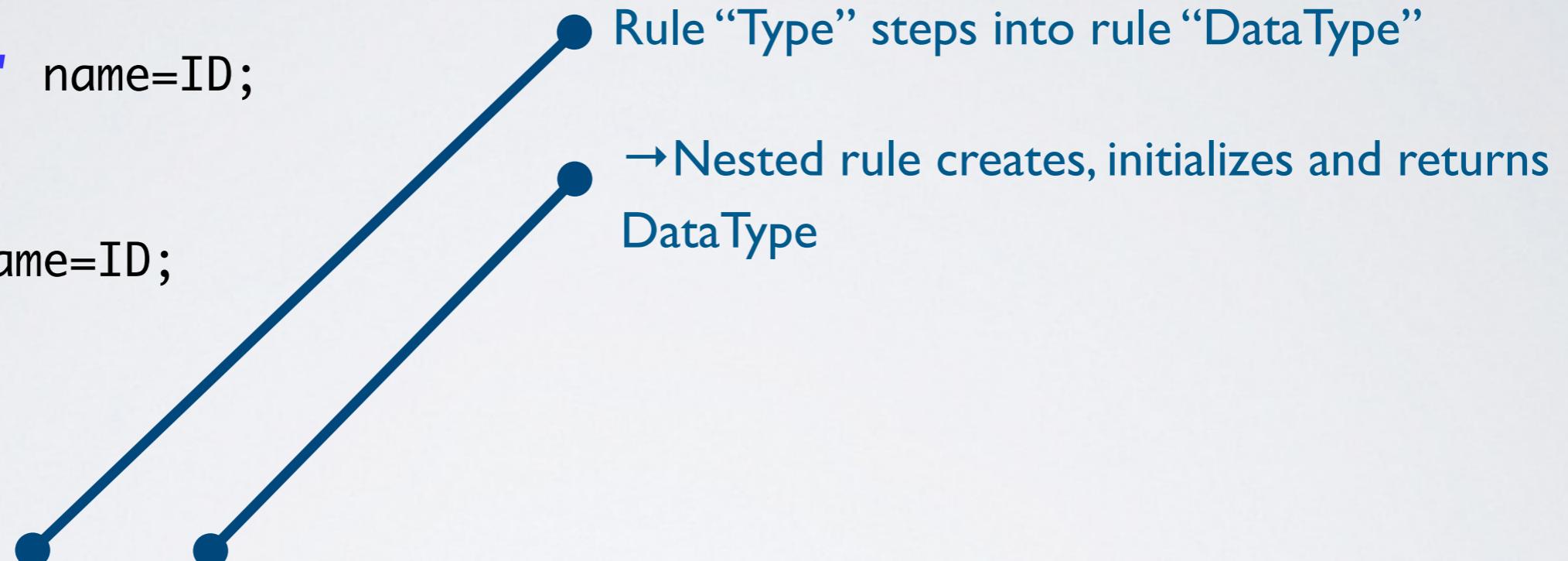
DataType :

```
'datatype' name=ID;
```

Class :

```
'class' name=ID;
```

Example datatype A is public



RECAP: AST CONSTRUCTION II

Type :

```
(Class | DataType) 'is' visibility='public' | 'private');
```

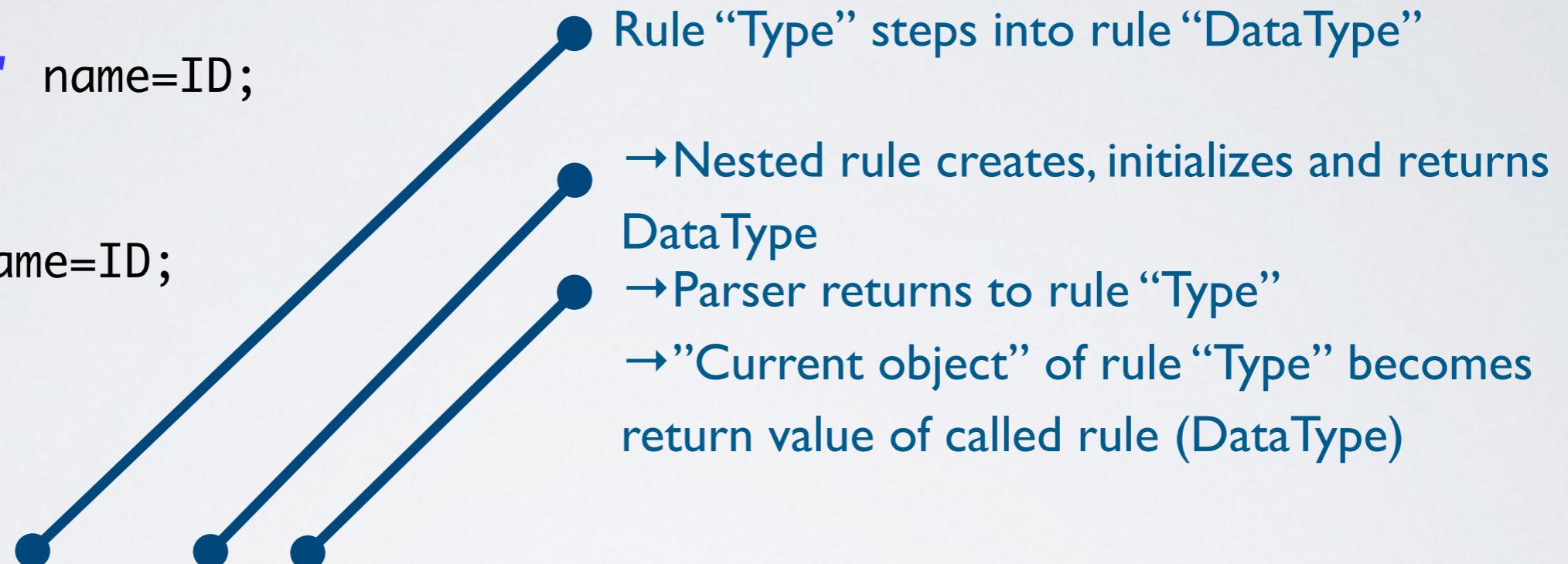
DataType :

```
'datatype' name=ID;
```

Class :

```
'class' name=ID;
```

Example datatype A is public



RECAP: AST CONSTRUCTION II

Type :

```
(Class | DataType) 'is' visibility='public' | 'private');
```

DataType :

```
'datatype' name=ID;
```

Class :

```
'class' name=ID;
```

Example

datatype A is public



Rule “Type” steps into rule “DataType”

→ Nested rule creates, initializes and returns
DataType

→ Parser returns to rule “Type”

→ “Current object” of rule “Type” becomes
return value of called rule (DataType)

→ Subsequent assignments set
features of “current object”

RECAP: AST CONSTRUCTION II

Type :

```
(Class | DataType) 'is' visibility='public' | 'private');
```

DataType :

```
'datatype' name=ID;
```

Class :

```
'class' name=ID;
```

Example

datatype A is public



Rule "Type" steps into rule "DataType"

→ Nested rule creates, initializes and returns
DataType

→ Parser returns to rule "Type"

→ "Current object" of rule "Type" becomes
return value of called rule (DataType)

→ Subsequent assignments set
features of "current object"

- Rule "Type" returns either a Class or a DataType

- In this example only one instance has been created

UNASSIGNED ACTIONS

Questionnaire:

(answers+=Answer)*;

Answer:

{Answer} 'no' | yes?= 'yes';

UNASSIGNED ACTIONS

Questionnaire:

(answers+=Answer)*;

Answer:

{Answer} 'no' | yes?= 'yes';

Example: yes no

UNASSIGNED ACTIONS

Questionnaire:

```
(answers+=Answer)*;
```

Answer:

```
{Answer} 'no' | yes?= 'yes';
```

Example: yes no

Assignment triggers object creation

UNASSIGNED ACTIONS

Questionnaire:

(answers+=Answer)*;

Answer:

{Answer} 'no' | yes?= 'yes';



UNASSIGNED ACTIONS

Questionnaire:

(answers+=Answer)*;

Answer:

{Answer} 'no' | yes?= 'yes';



Example:

- Without the action a no would never create an AST element

EXPRESSIONS

Expression :
 Expression '+' Expression |
 '(' Expression ')' |
 Number;

Number :
 value = INT;

EXPRESSIONS

► Expression :
 Expression '+' Expression |
 '(' Expression ')' |
 Number;

Number :
 value = INT;

EXPRESSIONS

► Expression :
 Expression '+' Expression |
 '(' Expression ')' |
 Number;

Number :
 value = INT;

LEFT-RECURSION !

EXPRESSIONS

Sum :

```
left=Number ('+' right=Sum)?;
```

Number :

```
value = INT;
```

EXPRESSIONS

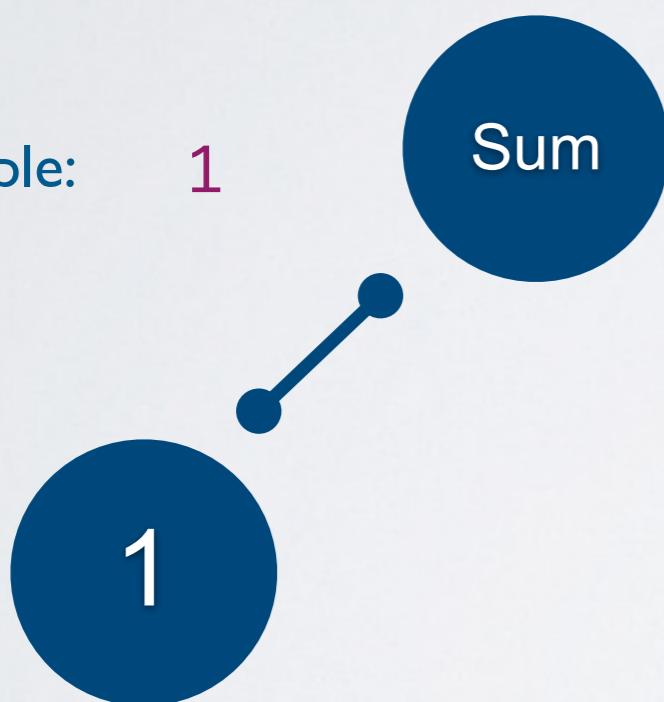
Sum :

```
left=Number ('+' right=Sum)?;
```

Number :

```
value = INT;
```

Example: 1



EXPRESSIONS

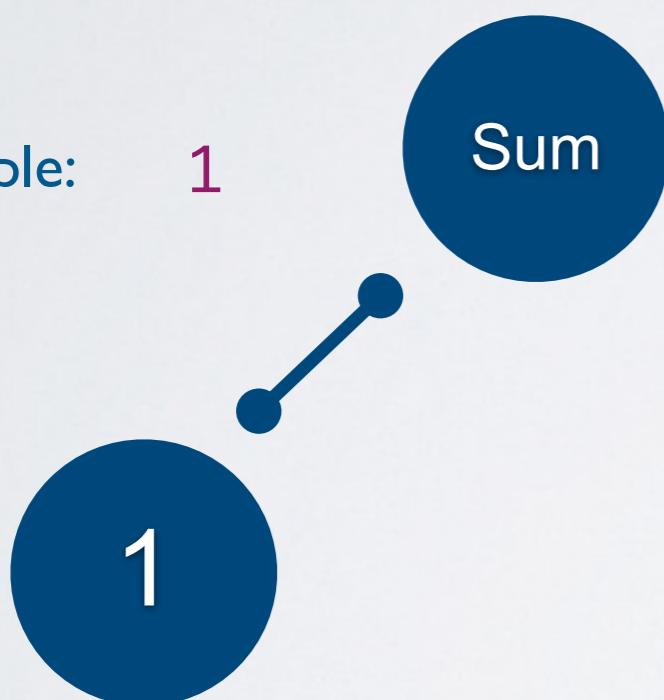
Sum :

```
left=Number ('+' right=Sum)?;
```

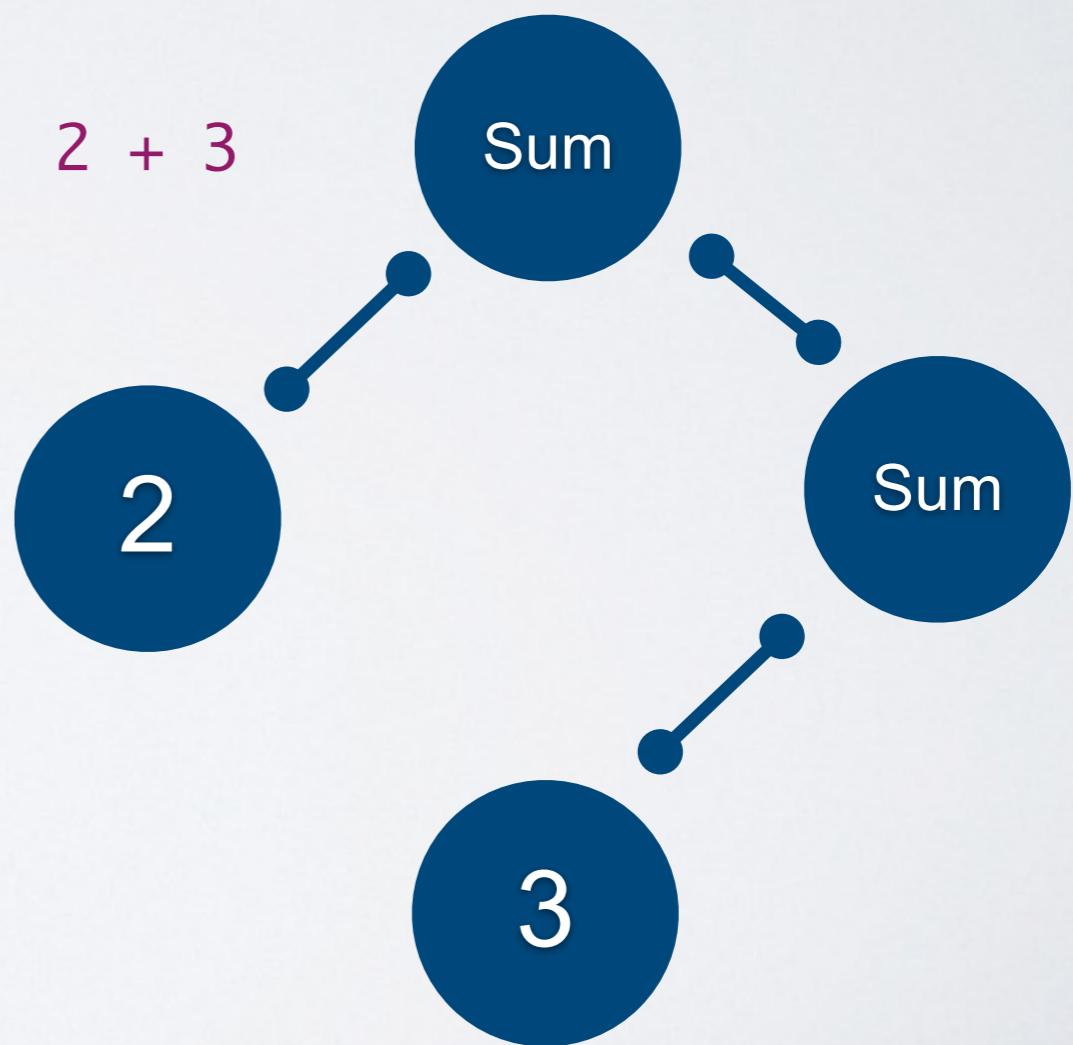
Number :

```
value = INT;
```

Example: 1



Example: 2 + 3



EXPRESSIONS

D-

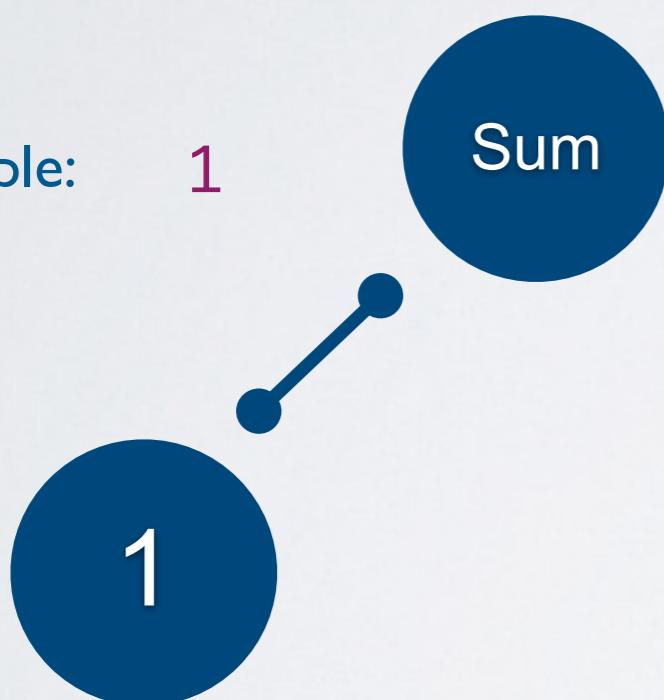
Sum :

```
left=Number ('+' right=Sum)?;
```

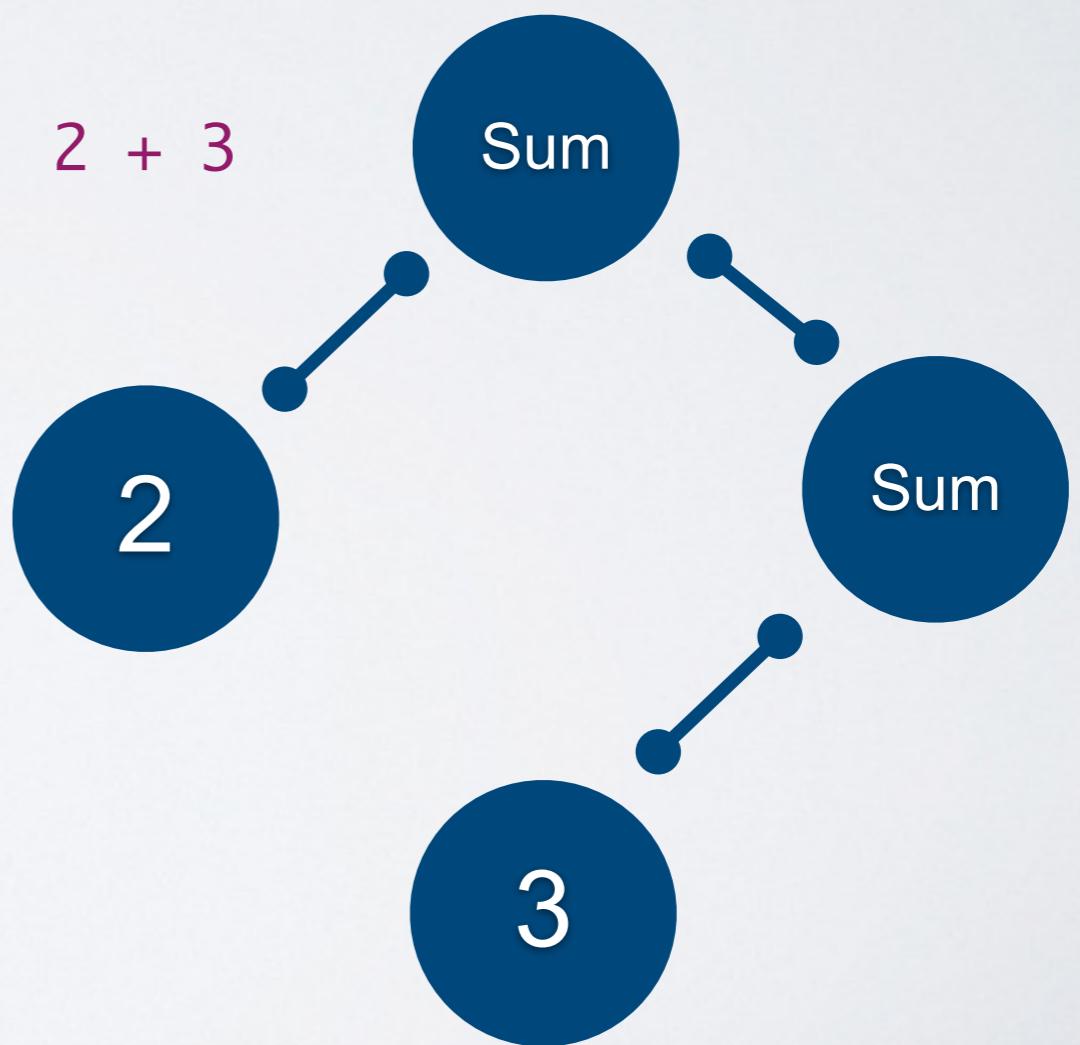
Number :

```
value = INT;
```

Example: 1



Example: 2 + 3



EXPRESSIONS

Addition **returns** Expression:

```
Number ({Sum.left = current} '+' right=Number)*;
```

Number :

```
value = INT;
```

EXPRESSIONS

Addition **returns** Expression:

```
Number ({Sum.left = current} '+' right=Number)*;
```

Number :

```
value = INT;
```

Example: 1



1

EXPRESSIONS

Addition **returns** Expression:

```
Number ({Sum.left = current} '+' right=Number)*;
```

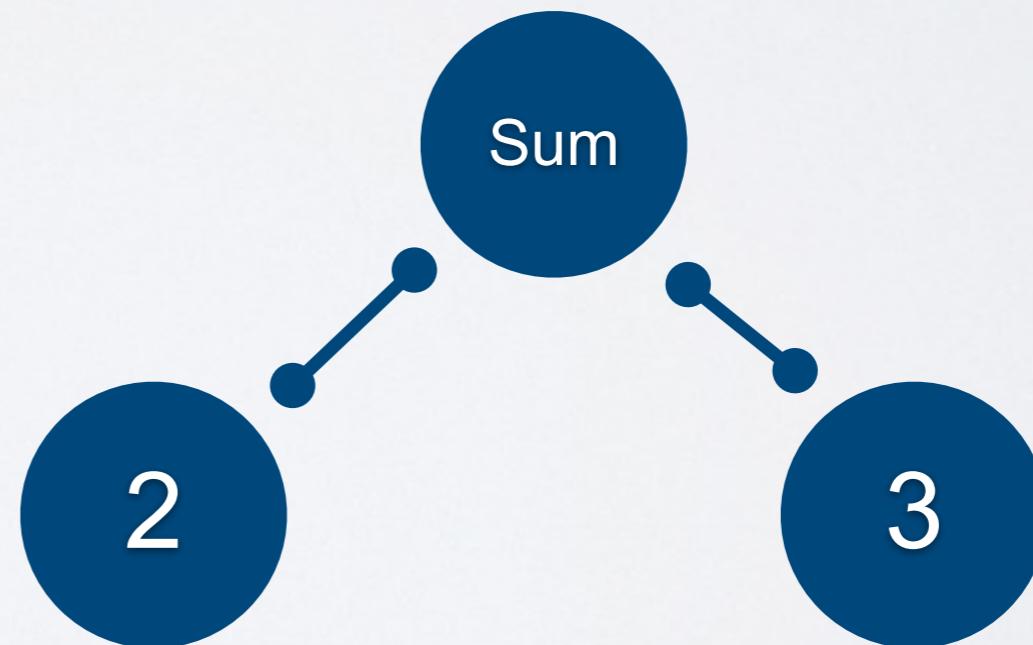
Number :

```
value = INT;
```

Example: 1



Example: 2 + 3



EXPRESSIONS

Addition **returns** Expression:

```
Number ({Sum.left = current} '+' right=Number)*;
```

Number :

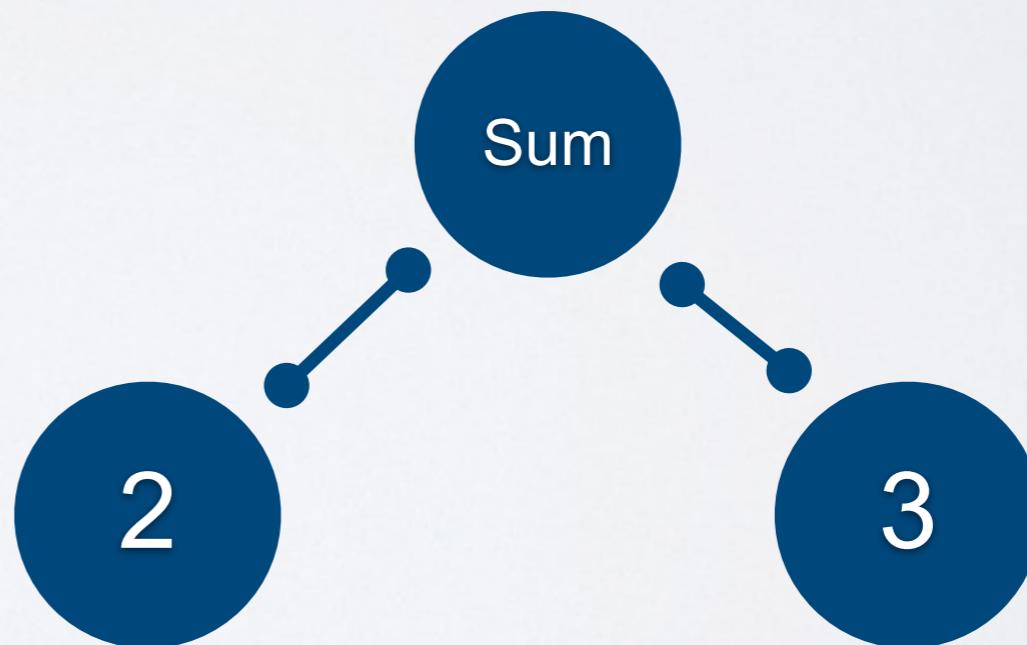
```
value = INT;
```



Example: 1



Example: 2 + 3



Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate Models
- Cross-References and Scoping
- Advanced Grammar
- EMF / Ecore integration
- Index
- UI customizations
- Assigned Actions
- **Testing**
- Xbase
- Problem Solving
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

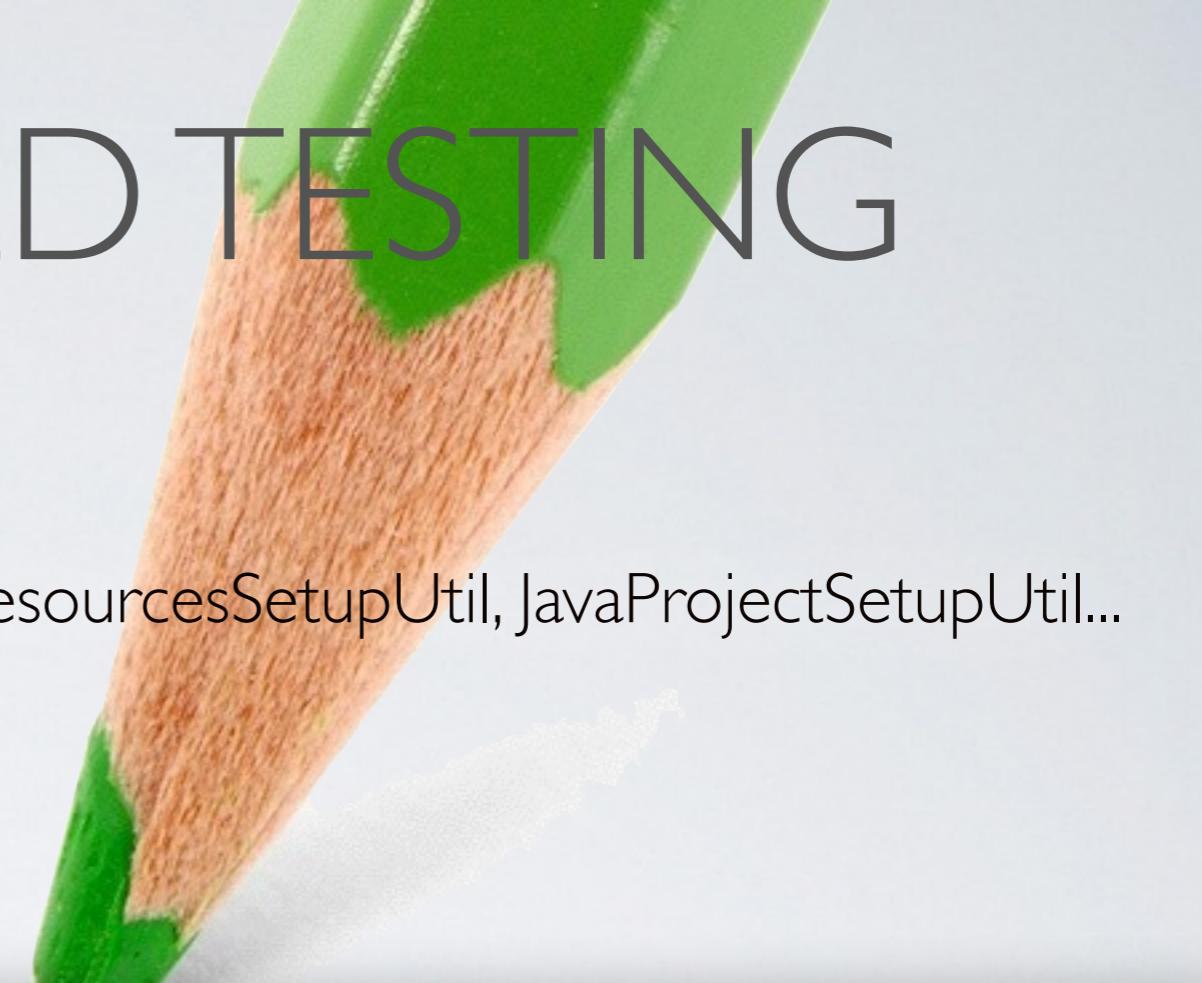
15:30 - 17:00

AUTOMATED TESTING

Xtext: JUnit utils:

XtextRunner, ParseHelper, ValidatorTester, IResourcesSetupUtil, JavaProjectSetupUtil...

GitHub: Xpect:



The screenshot shows the Eclipse IDE interface with the title bar "Java - org.domainmodel.xpect.tests/src/org/domainmodel/tests/validation/test1.dmodel xt - Eclipse Platform". The left side features the Package Explorer and JUnit perspectives, showing a successful run with 2/2 tests passed. The right side displays the content editor with the file "test1.dmodel xt". The code contains Xpect validation annotations and syntax errors. A tooltip is visible over the word "Property1", indicating a warning about capitalization. The status bar at the bottom shows "Failure Trace".

```
/* XPECT_SETUP org.domainmodel.tests.validation.DMValidationTest END_SETUP */

package pkg1 {
    entity MyEntity {

        // capitalized property names are discouraged
        // XPECT warnings --> "Name should start with a lowercase" at "Property1"
        Property1 : String

        // the character % is not valid syntax
        // XPECT errors --> "extraneous input '%' expecting '}'" at "%"
        %
    }
}
```

Xpect: reusable Tests as Library for...

- Parser and AST (demo only, no library).
- Code generators
- Validation
- Linking
- Scoping
- ResourceDescriptions
- JvmModellInferrer



DEMO!

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate Models
- Cross-References and Scoping
- Advanced Grammar
- EMF / Ecore integration
- Index
- UI customizations
- Assigned Actions
- Testing
- **Xbase**
- Problem Solving
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

15:30 - 17:00

XBASE

Enable expressions in your language

XBASE

Enable expressions in your language

Translate your DSL concepts to Java concepts

XBASE

Enable expressions in your language

Translate your DSL concepts to Java concepts

Bring your expressions into context

XBASE

Enable expressions in your language

Translate your DSL concepts to Java concepts

Bring your expressions into context

Get seamless IDE integration

XBASE
DEMO

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate Models
- Cross-References and Scoping
- Advanced Grammar
- EMF / Ecore integration
- Index
- UI customizations
- Assigned Actions
- Testing
- Xbase
- **Problem Solving**
- Outlook

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

15:30 - 17:00

HOWTO FIND A GUICE BINDING CANDIDATE I

- Open the (Ui)Module of your language
- Find a method containing the interface's name in its name
 - 2x CTRL-O to include superclasses
- If there's nothing, open the interface and look for a @ImplementedBy annotation

HOWTO FIND A GUICE BINDING CANDIDATE 2

- Eclipse Navigation Features are the Swiss Army Knife
 - Open Type Dialog * .xtext.*.*<Aspect>
 - Type Hierarchy to Find Implementors
 - Find Usages
 - Call Hierarchy

HOW TO TWEAK SOME FEATURE I

- Is it documented in
 - Eclipse help,
 - the Xtext forum / newsgroup,
 - one of our blogs?
- If so:
 - Read these docs!

HOW TO TWEAK SOME FEATURE 2

- Is it available in one of the examples?
- If so: Monkey see - Monkey do. Look for
 - unusual generator fragments
 - custom bindings
 - extensions in plugin.xml

HOW TO TWEAK SOME FEATURE 3

- Is there an implementation of `IGeneratorFragment` ?
- If so:
 - Use the fragment in your workflow
 - Investigate its properties using code assist or looking at the code
 - Watch out for new stub classes in the `src` folders

HOW TO TWEAK SOME FEATURE 4

- Look for Xtext classes/packages whose names suggest to deal with the feature.
- If there is one:
 - Try to find the hook (interface binding), e.g. by investigating code, debugging, etc.
 - Rather inherit from existing code than write from scratch
 - The JavaDocs should guide you

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate Models
- Cross-References and Scoping
- Advanced Grammar
- EMF / Ecore integration
- Index
- UI customizations
- Assigned Actions
- Testing
- Xbase
- Problem Solving
- **Outlook**

Coffee

9:00 - 10:30

Lunch

11:00 - 12:30

Coffee

13:30 - 15:00

15:30 - 17:00

FURTHER STEPS

What others have built with Xtext:

<http://www.eclipse.org/Xtext/community.html>

Web Forum:

<https://www.eclipse.org/forums/index.php/f/27/>

Professional Support:

itemis

Book: Implementing Domain-Specific Languages with Xtext and Xtend by Lorenzo Bettini





CONFERENCE

PROGRAM

LOCATION

SPEAKERS

only 61 days, 23 hours, 54 minutes and 15 seconds left

XtextCON

May 18-20 2015
Kiel, Germany

REGISTER

www.xtextcon.org

Thank You