Installation

- Grab a USB key
- Install Eclipse
- Save the zip files

Xtext for Beginners

Moritz Eysholdt, Jan Köhnlein, Holger Schill

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate models
- Introduce Cross-references
- Outlook

Outline

- **Example Application**
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate models
- Introduce Cross-references
- Outlook

Eclipse Community Survey 2012

Eclipse Community Survey 2012

Eclipse Community Survey 2012 Evaluate

What is the *primary type* of software you are personally involved in developing?

- Desktop client applications
- Embedded software
- Mainframe applications
- Mobile applications
- Plug-ins for Eclipse
- Research/scientific applications

Applications

lications (client/server, CRM, database

are

Don't know

- Other (please specify)

Demo

What is the *primary* server framework you use for deployed applications?

- Apache Struts
- Apache Struts 2
- Apache Tapestry
- Drupal

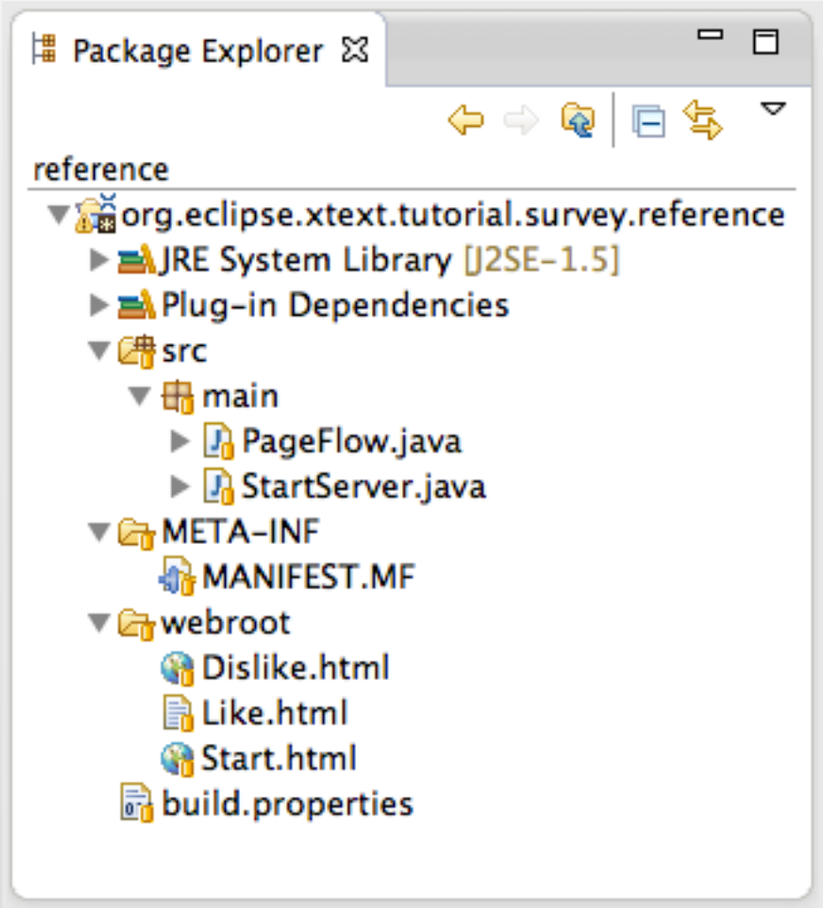
Example: Surveys

- 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 pageflow engine
- In memory persistence

API View


Package Explorer
reference
org.eclipse.xtext.tutorial.survey.reference
JRE System Library [J2SE-1.5]
Plug-in Dependencies
src
main
PageFlow.java
StartServer.java
META-INF
MANIFEST.MF
webroot
Dislike.html
Like.html
Start.html
build.properties

Challenges

- A heterogeneous platform
  - Java, HTML, maybe a Database
- Difficult to extend
  - more questions, multiple surveys
  - other front ends
- Hard to maintain

The Domain

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

DSL Approach

- Create a domain-specific language
  - Describes the data formally
- Generate code from its models

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"
    )
    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'
    )
)
```

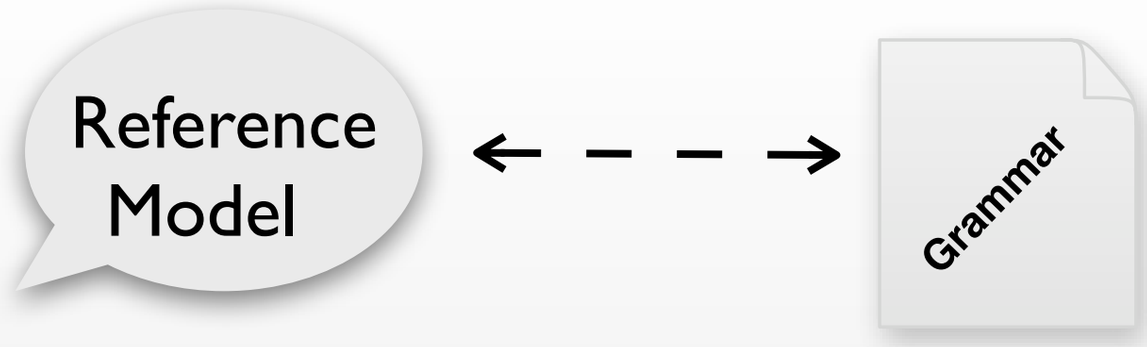
Advantages

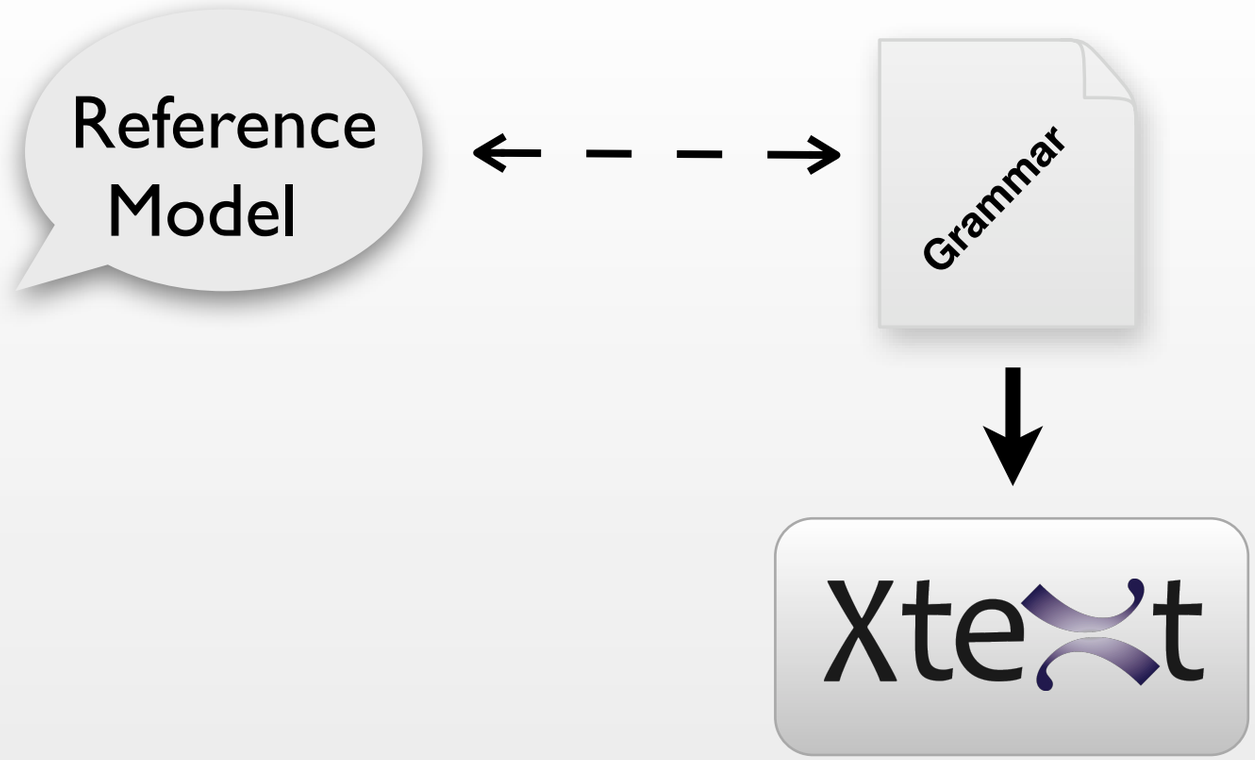
- Addresses heterogeneity
- Easy design of surveys
- Easy to add new front ends
- Separation of roles during development
- Speedup for development
- Improved maintainability

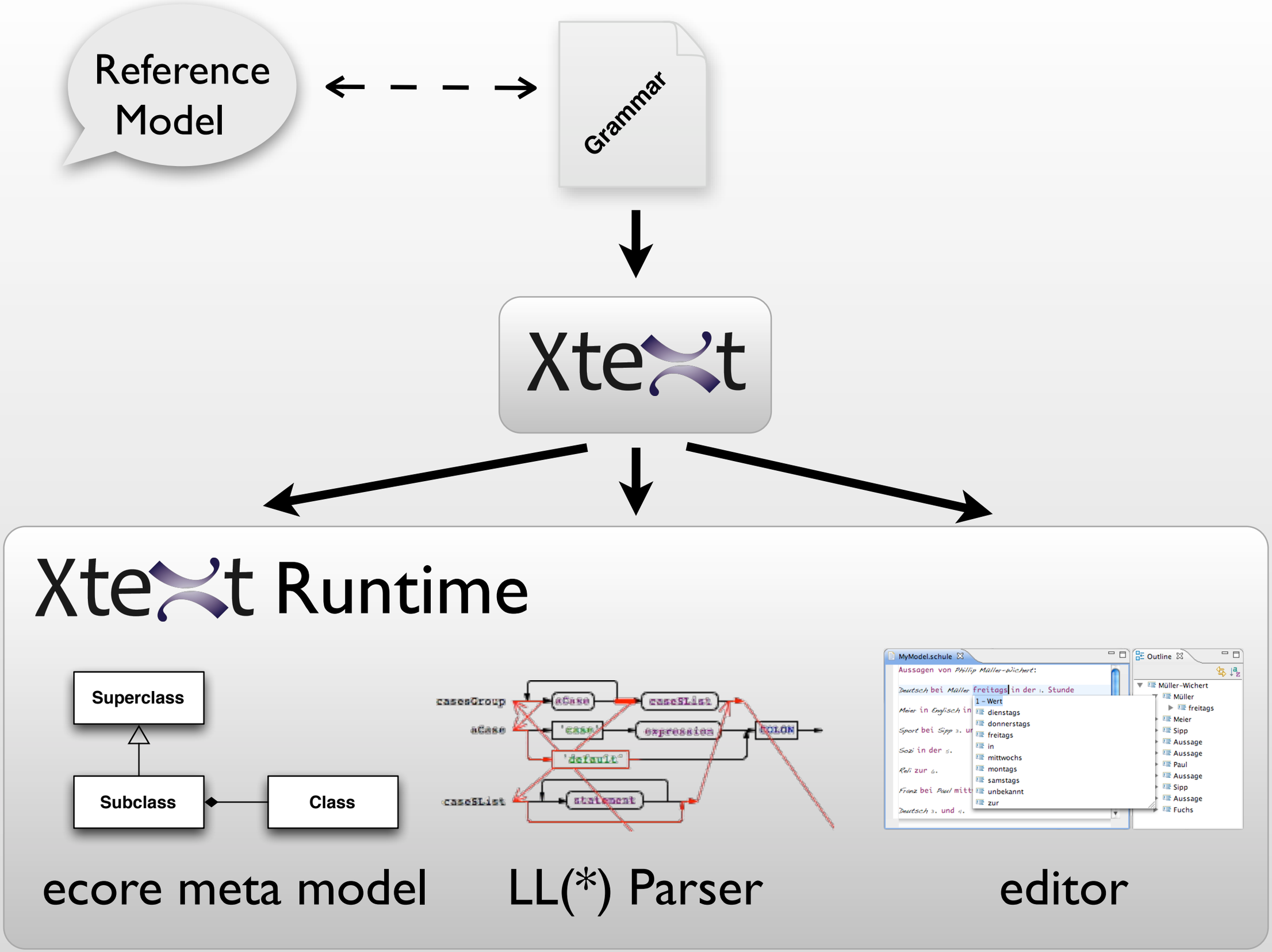
Outline

- Example Application
- **Build your DSL with Xtext**
- Generate Code with Xtend
- Validate models
- Introduce Cross-references
- Outlook


Grammar


Reference Model
Grammar


Reference Model
Grammar
Xtext


Reference Model
Grammar
Xtext
Xtext Runtime
Superclass
Subclass
Class
ecore meta model
LL(*) Parser
editor

```
grammar org.eclipse.xtext.tutorial.Grammar with org.eclipse.xtext.common.Terminals

generate survey "http://www.eclipse.org/xtext/tutorial/Grammar"

Model:
    entities+=Entity*;

Entity:
    abstract?="abstract" "entity" name=ID (tableName=ID)?"{"
        attributes+=Attribute+
    "}";

Attribute:
    StringAttribute | IntAttribute;

StringAttribute:
    "string" value=STRING
;

IntAttribute:
    "int" value=INT
;
```

```
abstract entity Person tab_person {
    int 42
    string "Some String"
}
```

Grammar

Namespace

```
grammar org.eclipse.xtext.tutorial.Grammar with org.eclipse.xtext.common.Terminals

generate survey "http://www.eclipse.org/xtext/tutorial/Grammar"

Model:
    entities+=Entity*;

Entity:
    abstract?="abstract" "entity" name=ID (tableName=ID)?"{"
        attributes+=Attribute+
    "}";

Attribute:
    StringAttribute | IntAttribute;

StringAttribute:
    "string" value=STRING
;

IntAttribute:
    "int" value=INT
;
```

```
abstract entity Person tab_person {
    int 42
    string "Some String"
}
```

Parser Rule

```
grammar org.eclipse.xtext.tutorial.Grammar with org.eclipse.xtext.common.Terminals

generate survey "http://www.eclipse.org/xtext/tutorial/Grammar"

Model:
    entities+=Entity*;

Entity:
    abstract?="abstract" "entity" name=ID (tableName=ID)?"{"
        attributes+=Attribute+
    "}";

Attribute:
    StringAttribute | IntAttribute;

StringAttribute:
    "string" value=STRING
;

IntAttribute:
    "int" value=INT
;
```

Keyword

```
abstract entity Person tab_person {
    int 42
    string "Some String"
}
```

```
grammar org.eclipse.xtext.tutorial.Grammar with org.eclipse.xtext.common.Terminals

generate survey "http://www.eclipse.org/xtext/tutorial/Grammar"

Model:
    entities+=Entity*;

Entity:
    abstract?="abstract" "entity" name=ID (tableName=ID)?"{"
        attributes+=Attribute+
    "}";

Attribute:
    StringAttribute | IntAttribute;

StringAttribute:
    "string" value=STRING
;

IntAttribute:
    "int" value=INT
;
```

Simple assignment

Boolean assignment

Multivalue assignment

```
abstract entity Person tab_person {
    int 42
    string "Some String"
}
```

```
grammar org.eclipse.xtext.tutorial.Grammar with org.eclipse.xtext.common.Terminals

generate survey "http://www.eclipse.org/xtext/tutorial/Grammar"

Model:
    entities+=Entity*;

Entity:
    abstract?="abstract" "entity" name=ID (tableName=ID)?"{"
        attributes+=Attribute+
    "}";

Attribute:
    StringAttribute | IntAttribute;

StringAttribute:
    "string" value=STRING
;

IntAttribute:
    "int" value=INT
;
```

Cardinality 0..*

Cardinality 1..*

```
abstract entity Person tab_person {
    int 42
    string "Some String"
}
```

```
grammar org.eclipse.xtext.tutorial.Grammar with org.eclipse.xtext.common.Terminals

generate survey "http://www.eclipse.org/xtext/tutorial/Grammar"

Model:
    entities+=Entity*;

Entity:
    abstract?="abstract" "entity" name=ID (tableName=ID)?"{"
        attributes+=Attribute+
    "}";

Attribute:
    StringAttribute | IntAttribute;

StringAttribute:
    "string" value=STRING
;

IntAttribute:
    "int" value=INT
;
```

Optional

```
abstract entity Person tab_person {
    int 42
    string "Some String"
}
```

```
grammar org.eclipse.xtext.tutorial.Grammar with org.eclipse.xtext.common.Terminals

generate survey "http://www.eclipse.org/xtext/tutorial/Grammar"

Model:
    entities+=Entity*;

Entity:
    abstract?="abstract" "entity" name=ID (tableName=ID)?"{"
        attributes+=Attribute+
    "}";

Attribute:
    StringAttribute | IntAttribute;

StringAttribute:
    "string" value=STRING
;

IntAttribute:
    "int" value=INT
;
```

Alternative

```
abstract entity Person tab_person {
    int 42
    string "Some String"
}
```

Outline

- Example Application
- Build your DSL with Xtext
- **Generate Code with Xtend**
- Validate models
- Introduce Cross-references
- Outlook

Xtend: 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»
      «element»
   «ENDFOR»
   For decisions there is the IF statement
   «IF elements.isEmpty()»
       no elements.
   «ENDIF»
'''
```

Xtend: Dispatch Methods

```
def dispatch area(Rectangle r) {
   r.width * r.height
}

def dispatch area(Circle c) {
   c.radius * c.radius * Math::PI
}

def someCalculation(Shape s) {
   area(s) // polymorphic call
}
```

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- **Validate models**
- Introduce Cross-references
- Outlook

Validator

```
@Check
def textMustNotBeEmpty(Question question) {
   if(question.getText().isEmpty()) {
      error("Empty question is illegal",
            question,
            QUESTION__TEXT)
   }
}
```

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate models
- **Introduce Cross-references**
- Outlook

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]
   ')';
```

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 '...'
    )
...
```

referable **Choices**	
global scope	custom scope
Start.like.yes	yes
Start.like.no	no
Like.particular.xtext	-
Like.particular.exercises	-
Like.particular.tutors	-
...	-

Outline

- Example Application
- Build your DSL with Xtext
- Generate Code with Xtend
- Validate models
- Introduce Cross-references
- **Outlook**

Outlook

- Enhance generator
- Customize IDE
  - Outline, Formatter, Labels, ...
- Add more expressions

More on Xtext/Xtend at EclipseCon 2013

- DSLs for Java
- Xtext Best Practices
- Xtext in the Web (Orion Symposium)
- Executable Specification for Xtext (Modeling Symposium)
- Xtend Tutorial
- Xtend Internal DSLs
- Xtend and JavaFX

Thank You

... and don’t forget the evaluation!