

What's Cooking in Xtext^{2.0}

Sebastian Zarnekow, Jan Köhnlein
itemis



A Language IDE Framework

Define Your
Own Language

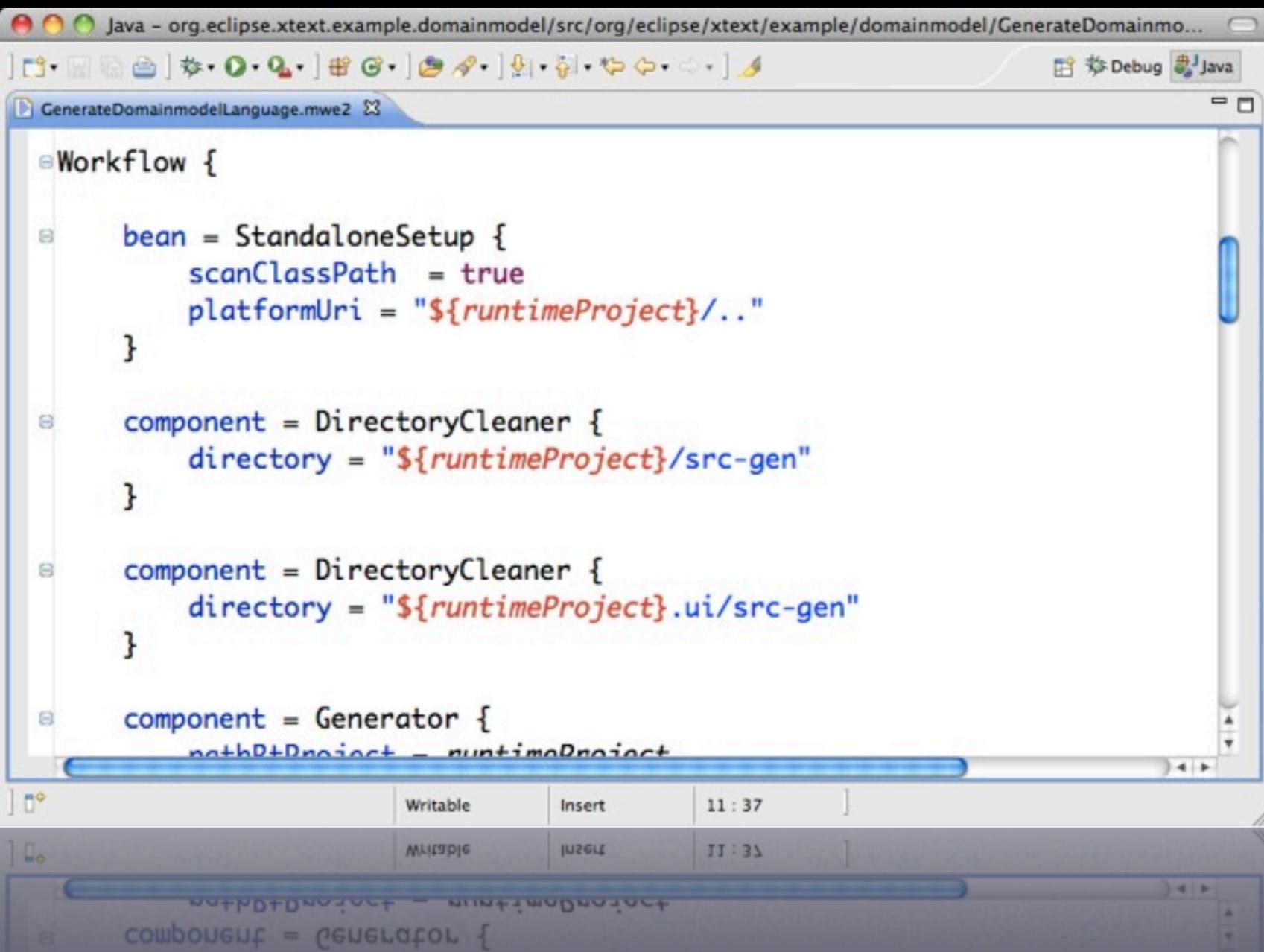


Generate a
Rich IDE



Execute
Your Language

<NoXML/>



The screenshot shows the Eclipse IDE interface with a Java editor open. The title bar reads "Java - org.eclipse.xtext.example.domainmodel/src/org/eclipse/xtext/example/domainmodel/GenerateDomainmo...". The editor displays a mwe2 workflow script:

```
Workflow {  
    bean = StandaloneSetup {  
        scanClassPath = true  
        platformUri = "${runtimeProject}..."  
    }  
  
    component = DirectoryCleaner {  
        directory = "${runtimeProject}/src-gen"  
    }  
  
    component = DirectoryCleaner {  
        directory = "${runtimeProject}.ui/src-gen"  
    }  
  
    component = Generator {  
        pathP+Project = runtimeProject  
    }  
}  
  
componenf = GENERATOR {
```

Create Executable DSLs

The screenshot shows the Eclipse IDE interface with two editors open. On the left, the 'Base.dmodel' editor displays a Domain-Specific Language (DSL) definition for a 'base' package. It contains two entity definitions: 'Person' and 'Address'. The 'Person' entity has attributes: name (String), surname (String), dateOfBirth (Date), and addresses (List<Address>). The 'Address' entity has attributes: street (String), city (String), state (String), and postalCode (String). On the right, the 'Person.java' editor shows the generated Java code for the 'Person' entity. It includes a constructor, getters for name and surname, and setters for name. The Java code is color-coded, with 'base' and 'Person' in purple, and other identifiers in blue.

```
Java - example.domainmodel/src/Base.dmodel - Eclipse SDK
```

```
Base.dmodel
```

```
Address.java Person.java
```

```
package base;
import java.util.*;

entity Person {
    name: String
    surname: String
    dateOfBirth: Date
    addresses: List<Address>
}

entity Address {
    street: String
    city: String
    state: String
    postalCode: String
}
```

```
public class Person {
    private String name;

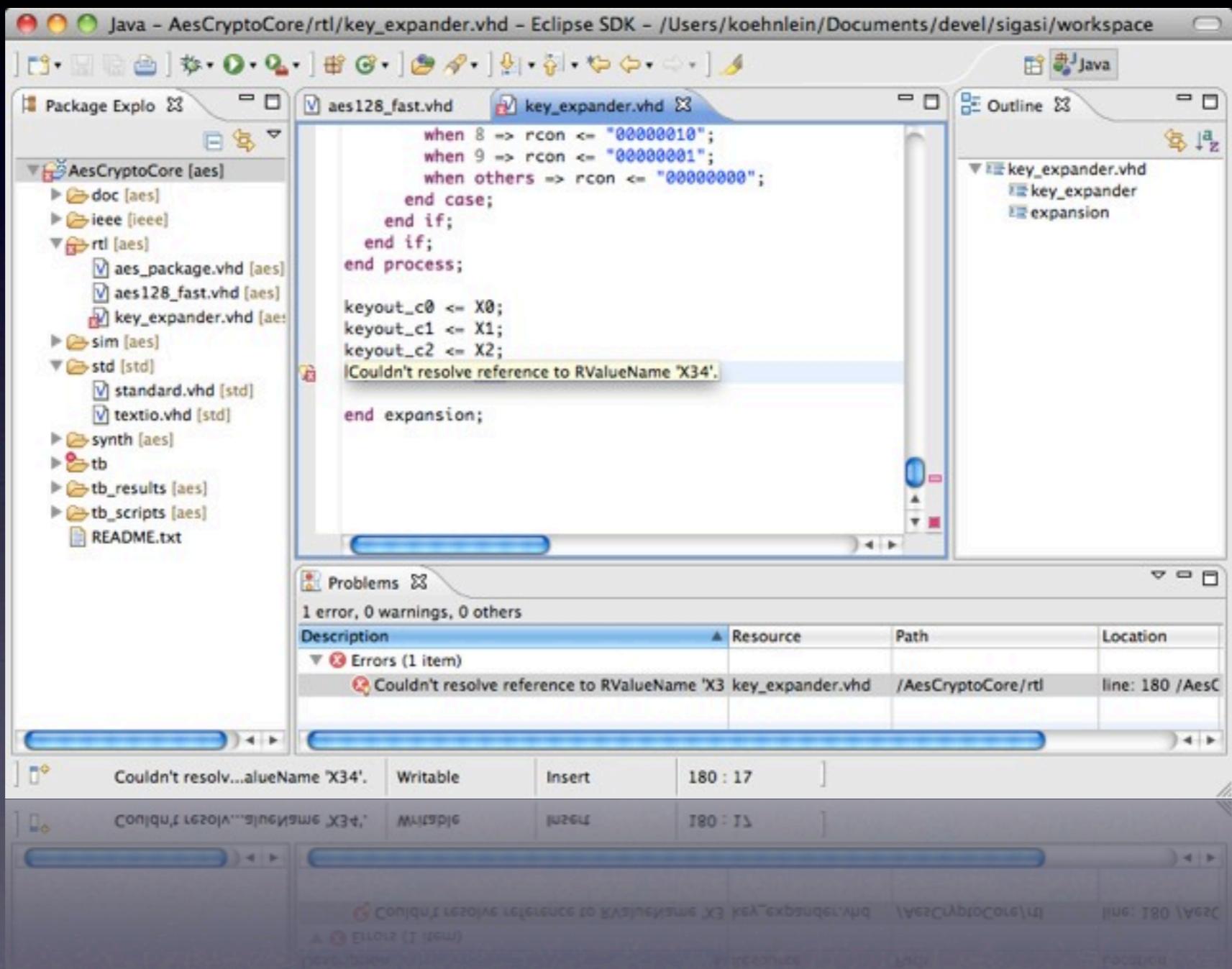
    public String getName() {
        return name;
    }

    public void setName(String na
        this.name = name;
    }

    private String surname;

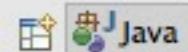
    public String getSurname() {
        return surname;
    }
}
```

Build IDEs for Existing Languages



Xtext^{2.0}

New
Features



Test.xtext

Example.dmodel

Refactoring.dmodel

Reference.dmodel

Base.dmodel

Business.dmodel

```
import java.util.*  
  
/**  
 * <h1>Xtext Hover</h1>  
 *  
 * You can now define <em>hovers</em> for Xtext elements.  
 * The default implementation parses comments in a JavaDoc-like fashion.  
 * Any <em>HTML</em> is allowed, even  
 * <a href="www.eclipsecon.org">hyperlinks</a> and images.  
 * 
      <or>
        <test property="org.eclipse.core.resources.name"
              value="*.java"/>
        <test property="org.eclipse.core.resources.name"
              value="*.JAVA"/>
      </or>
    </adapt>
  </iterate>
</activeWhen>
```

Xtext^{2.0}

D S L s u n l e a s h e d

xbase



Screencast Runtime Instance Domainmodel

Java – sample/src/Sample.dmodel – Eclipse Platform

The screenshot shows the Eclipse Platform interface with the title "Java – sample/src/Sample.dmodel – Eclipse Platform". The central part displays the contents of the file "Sample.dmodel". The code defines a package "org.company" with imports for "java.util.List" and "org.company.types.*". It contains an entity "Person" with attributes "firstName", "lastName", and "gender". The "Outline" view on the right shows the structure of the code, including the package, imports, and the Person entity with its attributes. The "Package Explorer" view on the left shows the project structure with files like "Gender.java" and "build.properties". The bottom navigation bar includes tabs for "Problems", "Javadoc", and "Declaration", and status indicators for "Writable", "Insert", and the current line number "11 : 9".

```
package org.company {  
    import java.util.List  
    import org.company.types.*  
  
    entity Person {  
        firstName: String  
        lastName: String  
        gender: Gender  
    }  
}
```

Outline View:

- org.company
 - java.util.List
 - org.company.types.*
- Person
 - firstName : String
 - lastName : String
 - gender : Gender

Package Explorer View:

- sample
 - JRE System Library [J2SE-1.5]
 - Plug-in Dependencies
 - src
 - org.company.types
 - Gender.java
 - Sample.dmodel
- META-INF
- build.properties

Bottom Status Bar:

- Writable
- Insert
- 11 : 9

Screencast Runtime Instance Domainmodel

Java – sample/src/Sample.dmodel – Eclipse Platform

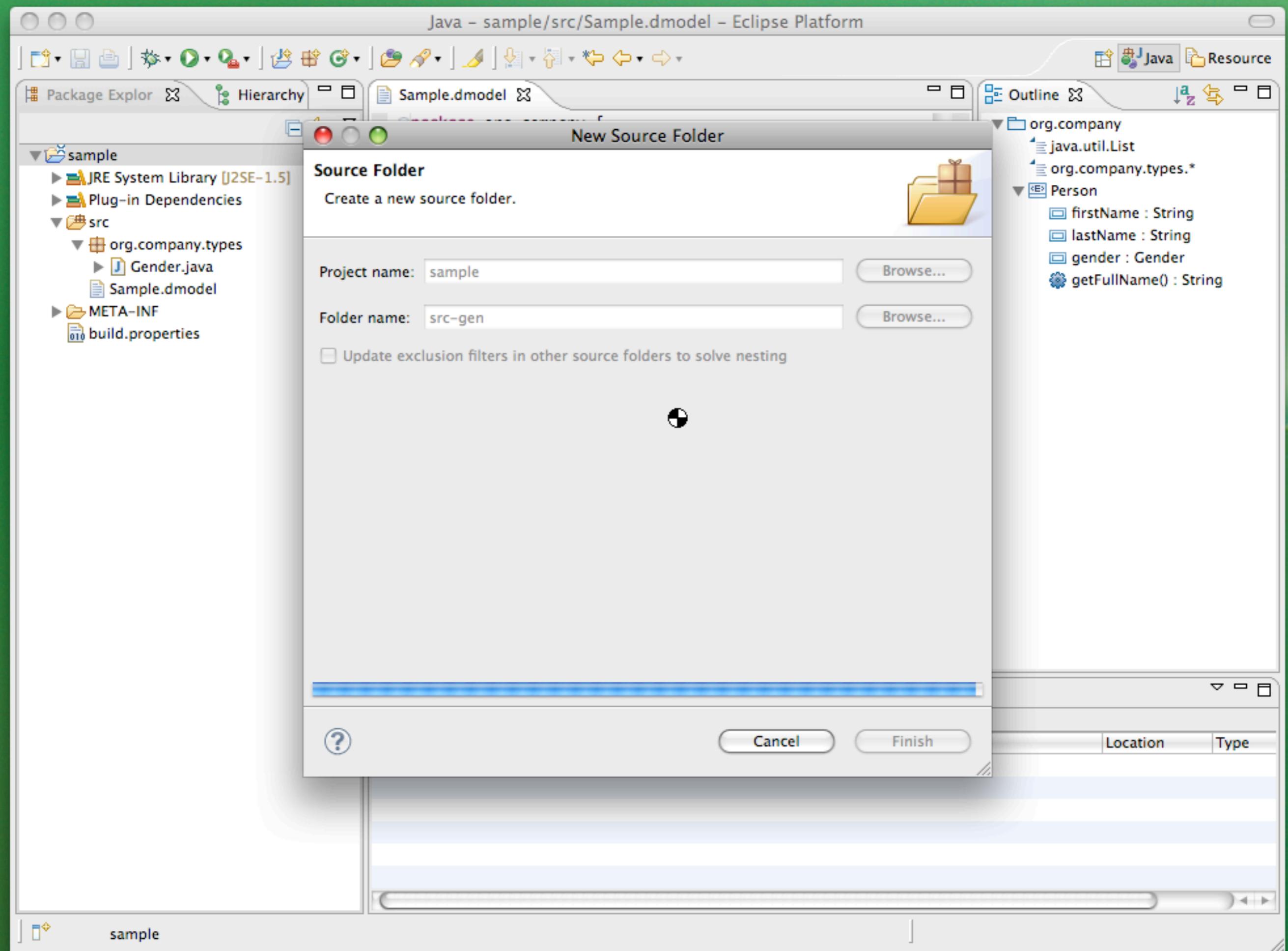
The screenshot shows the Eclipse Platform interface with the following components:

- Package Explorer:** Shows the project structure with a tree view of files and folders.
- Hierarchy:** A view showing the inheritance hierarchy of selected elements.
- Sample.dmodel:** The active editor showing Java code. The code defines a package `org.company` containing an entity `Person`. The `Person` entity has attributes `firstName`, `lastName`, and `gender`, and a method `getFullName()`. A tooltip for the `lastName` attribute is displayed, indicating it is a `String` property.
- Outline:** A view showing the structure of the current file, including the package, imports, entity declarations, and their attributes and methods.
- Problems:** A view showing 0 items.
- Declaration:** A tabbed view showing the declaration of the currently selected element.

Bottom status bar:

- Writable
- Insert
- 8 : 17

Screencast Runtime Instance Domainmodel



Screencast Runtime Instance Domainmodel

The screenshot shows the Eclipse Java IDE interface with the following components:

- Package Explorer:** Shows the project structure with packages `sample`, `JRE System Library [J2SE-1.5]`, `Plug-in Dependencies`, `src` (containing `org.company.types` with files `Gender.java` and `Sample.dmodel`), `src-gen` (containing `org.company` with file `Person.java`), `META-INF`, and `build.properties`.
- Editor:** Displays the `Person.java` source code. The code defines a class `Person` with private fields `firstName`, `lastName`, and `gender`, and public methods `getFirstName`, `setFirstName`, `getLastName`, and `setLastName`.
- Outline:** Shows the class structure of `Person`, including its fields and methods.
- Problems:** Shows 0 errors, 1 warning, and 0 others. The warning is listed under "Warnings (1 item)".

```

package org.company;

import org.company.types.Gender;

public class Person {
    private String firstName;

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    private String lastName;

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    private Gender gender;
}
  
```

Screencast Runtime Instance Domainmodel

Java – sample/src/Sample.dmodel – Eclipse Platform

The screenshot shows the Eclipse Platform interface with the following components:

- Package Explorer:** Shows the project structure with folders like sample, src, and src-gen.
- Hierarchy:** Shows the class hierarchy for Person.
- Sample.dmodel:** The main editor window containing the following DDL-like code:

```
package org.company;

import java.util.List
import org.company.types.*

entity Person {
    firstName: String
    lastName: String
    gender: Gender
    friends : List<Person>
}

op getFullName(): String {
    firstName + ' ' + lastName
}
```
- Outline:** A tree view showing the structure of the Person entity, including its attributes (firstName, lastName, gender, friends) and methods (getFullName).
- Problems:** Shows 0 errors, 1 warning, and 0 others.
- Declaration:** Shows the declaration of the Person entity.
- Table:** A table showing the details of the warning:

Description	Resource	Path	Location	Type
► ! Warnings (1 item)				
- Bottom Bar:** Includes buttons for Writable, Insert, and a timestamp (10 : 31).

Screencast Runtime Instance Domainmodel

Java – sample/src/Sample.dmodel – Eclipse Platform

Package Explor Hierarchy

sample

- JRE System Library [J2SE-1.5]
- Plug-in Dependencies
- src
 - org.company.types
 - Gender.java
 - Sample.dmodel
- src-gen
 - org.company
 - Person.java
- META-INF
- build.properties

*Sample.dmodel

```
package org.company;

import java.util.List
import org.company.types.*

entity Person {
    firstName: String
    lastName: String
    gender: Gender
    friends : List<Person>

    op getFullName(): String {
        firstName + ' ' + lastName
    }
    op getSortedFriends(): List<Person> {
        friends.sort()
    }
}
```

Outline

- org.company
 - java.util.List
 - org.company.types.*
- Person
 - firstName : String
 - lastName : String
 - gender : Gender
 - friends : List<Person>
 - getFullName() : String
 - getSortedFriends() : List<Person>

Problems @ Javadoc Declaration

0 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
► Warnings (1 item)				

Writable Insert 16 : 23

<T> List<T>
org.eclipse.xtext.xbase.lib.ListExtensions.sort<T> list)

Screencast Runtime Instance Domainmodel

Java – sample/src/Sample.dmodel – Eclipse Platform

The screenshot shows the Eclipse Platform interface with the title "Java – sample/src/Sample.dmodel – Eclipse Platform". The central view displays the contents of the file "Sample.dmodel". The code defines a package "org.company" containing an entity "Person" with attributes "firstName", "lastName", "gender", and a list of friends. It also contains two operations: "getFullName()" and "getSortedFriends()". The cursor is positioned at the end of the "getSortedFriends()" implementation, specifically at the line "friends.sortBy(ele.fullName)". A tooltip is open over the "fullName" identifier, showing its declaration: "fullName : String – Person.getFullName()". The left side of the interface features the "Package Explorer" view, which shows the project structure with folders like "sample", "src", and "src-gen", and files like "Gender.java" and "Sample.dmodel". The right side has an "Outline" view showing the class hierarchy and member details. The bottom of the screen includes a "Problems" view showing 0 errors, 1 warning, and 0 others, and a "Declaration" view showing the current declaration of "fullName". The status bar at the bottom indicates the file is "Writable" and shows the time as 16:36.

```
package org.company {  
    import java.util.List  
    import org.company.types.*  
  
    entity Person {  
        firstName: String  
        lastName: String  
        gender: Gender  
        friends : List<Person>  
  
        op getFullName(): String {  
            firstName + ' ' + lastName  
        }  
        op getSortedFriends(): List<Person> {  
            friends.sortBy(ele.fullName)  
        }  
    }  
}
```

Xbase

Operator
Overloading

Closures

Concise
Syntax

Full
Generics

Compiled
To Java

The screenshot shows the Eclipse IDE interface with the following components:

- Package Explorer:** Shows the project structure with files like MANIFEST.MF, about.html, build.properties, plugin.properties, plugin.xml, and plugin.xml_gen.
- Editor View:** Displays the Xtext grammar file `Domainmodel.xtext`. The code defines a grammar for `Domainmodel` with rules for `DomainModel`, `AbstractElement`, `Import`, and `PackageDeclaration`.
- Outline View:** Shows a tree structure of grammar elements under the node `grammar org.eclipse.xtext.example.domainmodel.Domainmodel with org.eclipse.xtext.common.Terminals`. Elements include `DomainModel`, `AbstractElement`, `Import`, `PackageDeclaration`, `Entity`, `Feature`, `Property`, `Visibility`, `QualifiedNameWithWildCard`, and `QualifiedName`.
- Bottom Bar:** Includes tabs for Problems, Console, Progress, Search, Call Hierarchy, Properties, and a status bar showing memory usage (68M of 254M), writability, and current line (3 : 16).

```

/*
 * Copyright (c) 2009 itemis AG (http://www.itemis.eu) and others
 * All rights reserved. This program and the accompanying material
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * http://www.eclipse.org/legal/epl-v10.html
 */
grammar org.eclipse.xtext.example.domainmodel.Domainmodel with org.eclipse.xtext.common.Terminals

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

DomainModel:
    elements+=AbstractElement*;

AbstractElement:
    PackageDeclaration | Entity | Import;

Import:
    'import' importedNamespace=QualifiedNameWithWildCard;

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

```

* Copyright (c) 2009 itemis AG (<http://www.itemis.eu>) and others.
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * <http://www.eclipse.org/legal/epl-v10.html>
 *****/

```

grammar org.eclipse.xtext.example.domainmodel.Domainmodel with org.eclipse.xtext.xbase.Xbase

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

DomainModel:
    elements+=AbstractElement*;

AbstractElement:
    PackageDeclaration | Entity | Import;

Import:
    'import' importedNamespace=QualifiedNameWithWildCard;

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

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

Feature:
    Property;

Property:
    name=ID ':' type=Entity;

enum Visibility:
```

82M of 254M



Updating projects: (69%)



Writable

Insert

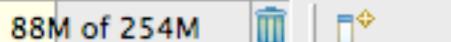
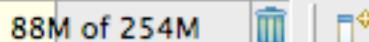
8:



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

- DomainModel:**
elements+=AbstractElement*
- AbstractElement:**
PackageDeclaration | Entity | Import;
- Import:**
'import' importedNamespace=QualifiedNameWithWildCard;
- PackageDeclaration:**
'package' name=QualifiedName '{'
elements+=AbstractElement*
'}';
- Entity:**
'entity' name=ID ('extends' superType=JvmTypeReference)? '{'
features+=Feature*
'}';
- Feature:**
Property ;
- Property:**
name=ID ':' type=JvmTypeReference;
- enum Visibility:**
public | private | protected;
- QualifiedNameWithWildCard :**
QualifiedName ('.' '*')?;
- QualifiedName:**
ID ('.' ID)*
;

88M of 254M



Writable

Insert

32



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

- DomainModel:
 - elements+=AbstractElement*
- AbstractElement:
 - PackageDeclaration | Entity | Import;
- Import:
 - 'import' importedNamespace=QualifiedNameWithWildCard;
- PackageDeclaration:
 - 'package' name=QualifiedName '{'
 - elements+=AbstractElement*
 - '}'
- Entity:
 - 'entity' name=ID ('extends' superType=JvmTypeReference)? '{'
 - features+=Feature*
 - '}'
- Feature:
 - Property | Operation;
- Property:
 - name=ID ':'
- enum Visibility
 - public | pr
- QualifiedNameWi
 - QualifiedNa
- QualifiedName:
 - ID ('.' ID)
 - ;

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

- DomainModel:
 - elements+=AbstractElement*
- AbstractElement:
 - PackageDeclaration | Entity | Import;
- Import:
 - 'import' importedNamespace=QualifiedNameWithWildCard;
- PackageDeclaration:
 - 'package' name=QualifiedName '{'
 - elements+=AbstractElement*
 - '}'
- Entity:
 - 'entity' name=ID ('extends' superType=JvmTypeReference)? '{'
 - features+=Feature*
 - '}'
- Feature:
 - Property | Operation;
- Operation:
 - 'op' name=ID '(' (parameters+=JvmFormalParameter (',' parameters+=JvmFormalParameter)*)? ')'
 - :: returnType=JvmTypeReference
 - body=XBlockExpression
- Property:
 - name=ID
- enum Visibility:
 - public |

ParserRule XExpression

- XAdditiveExpression
- XAndExpression
- XAssignment
- XBlockExpression
- XBooleanLiteral
- XCasePart
- XCastedExpression
- XCatchClause
- XClosure
- XConstructorCall