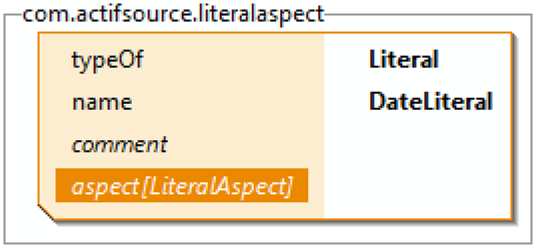
Tutorial

Literal Aspect

Tutorial	Actifsource Tutorial – Literal Aspect
Required Time	• 30 Minutes
Prerequisites	• Actifsource Tutorial – Installing Actifsource • Actifsource Tutorial – Simple Service
Goal	• Writing an aspect for custom literal types • Validate custom literals
Topics covered	• Create a simple demo model • Add a new Literal • Setup the project • Implement the Literal Aspect • Using the Literal Aspect
Notation	↳ To do ⓘ Information • **Bold**: Terms from actifsource or other technologies and tools • **<u>Bold underlined</u>**: actifsource Resources • <u>Underlined</u>: User Resources • *<u>UnderlinedItalics</u>*: Resource Functions • `Monospaced`: User input • *Italics*: Important terms in current situation
Disclaimer	The authors do not accept any liability arising out of the application or use of any information or equipment described herein. The information contained within this document is by its very nature incomplete. Therefore the authors accept no responsibility for the precise accuracy of the documentation contained herein. It should be used rather as a guide and starting point.
Contact	**actifsource GmbH** Täfernstrasse 37 5405 Baden-Dättwil Switzerland <u>www.actifsource.com</u>
Trademark	**actifsource** is a registered trademark of **actifsource GmbH** in Switzerland, the EU, USA, and China. Other names appearing on the site may be trademarks of their respective owners.

Overview

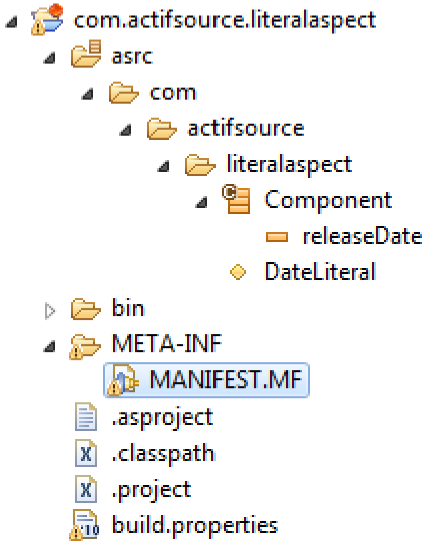
- Create a simple demo model
- Add a new Literal



com.actifsource.literalaspect

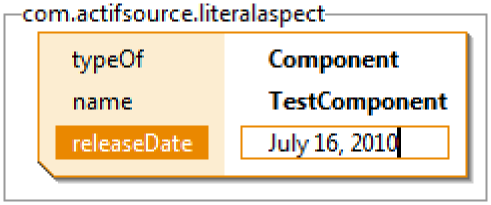
typeOf	**Literal**
name	**DateLiteral**
comment	
aspect [LiteralAspect]	

- Setup the project



```
com.actifsource.literalaspect
  asrc
    com
      actifsource
        literalaspect
          Component
            releaseDate
          DateLiteral
  bin
  META-INF
    MANIFEST.MF
  .asproject
  .classpath
  .project
  build.properties
```

- Implement the Literal Aspect
- Using the Literal Aspect

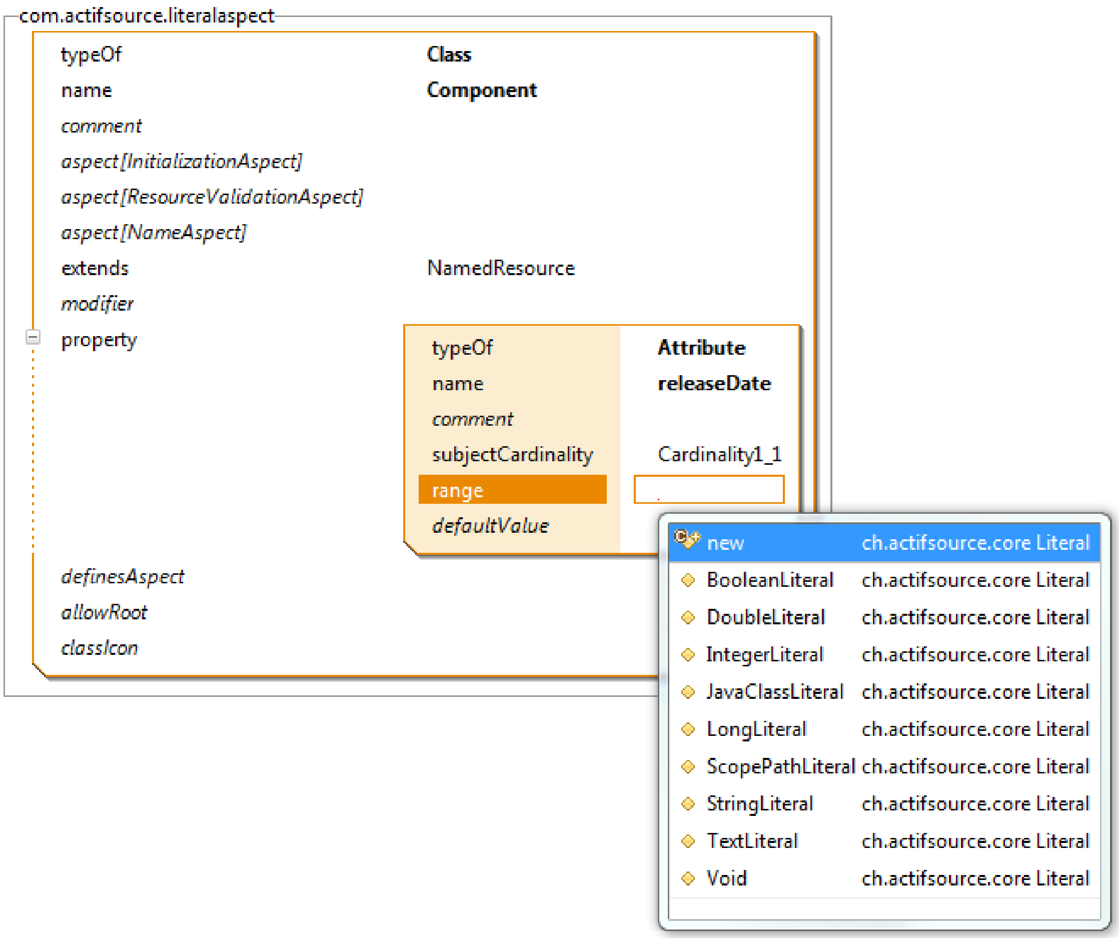


com.actifsource.literalaspect

typeOf	**Component**
name	**TestComponent**
releaseDate	July 16, 2010

Part I: Create a simple demo model

- Prepare a new **actifsource Project** as seen in the *Actifsource Tutorial – Simple Service*
- Create a new class as shown in the picture



com.actifsource.literalaspect

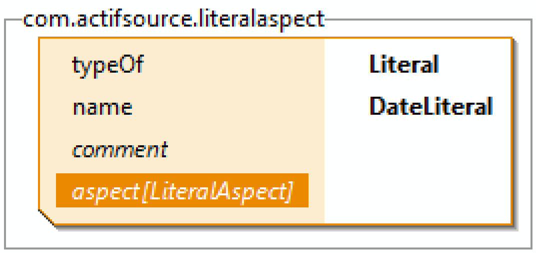
typeOf	**Class**
name	**Component**
comment	
aspect [InitializationAspect]	
aspect [ResourceValidationAspect]	
aspect [NameAspect]	
extends	NamedResource
modifier	
property	
definesAspect	
allowRoot	
classIcon	

typeOf	**Attribute**
name	**releaseDate**
comment	
subjectCardinality	Cardinality1_1
range	
defaultValue	

new	ch.actifsource.core Literal
BooleanLiteral	ch.actifsource.core Literal
DoubleLiteral	ch.actifsource.core Literal
IntegerLiteral	ch.actifsource.core Literal
JavaClassLiteral	ch.actifsource.core Literal
LongLiteral	ch.actifsource.core Literal
ScopePathLiteral	ch.actifsource.core Literal
StringLiteral	ch.actifsource.core Literal
TextLiteral	ch.actifsource.core Literal
Void	ch.actifsource.core Literal

Part II: Add a new LiteralRange

- ⓘ As you can see in the content assist, there is no Literal for representing a date.
- Create a new Literal using the content assist
- Set the name to "DateLiteral"



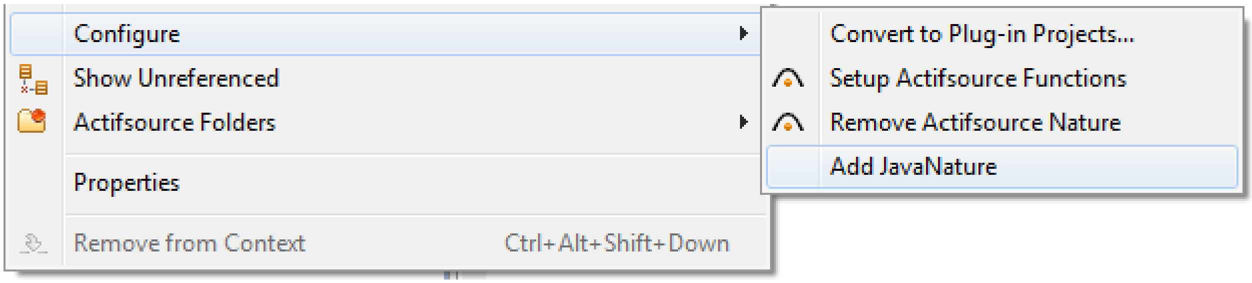
typeOf	Literal
name	DateLiteral
comment	
aspect [LiteralAspect]	

- For the moment there is not much more we can do, since we first need to setup the project.

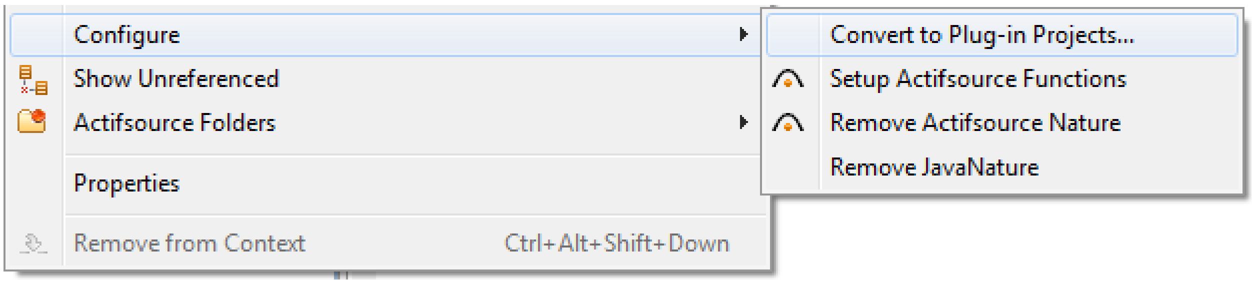
Part III: Setup the project

ⓘ First we need to define out actifsource project as a java project.
ⓘ Note that eclipse allows a project to be an actifsource project and a java project at the same time.

↳ Right click on the project and select “configure” and “Add JavaNature”


Configure
Show Unreferenced
Actifsource Folders
Properties
Remove from Context Ctrl+Alt+Shift+Down
Convert to Plug-in Projects...
Setup Actifsource Functions
Remove Actifsource Nature
Add JavaNature

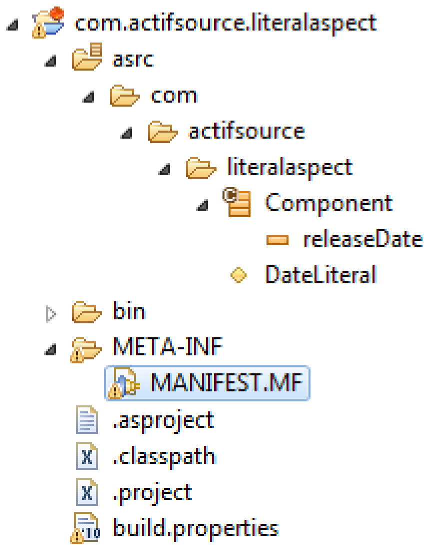
↳ Right click on the project and select “configure” and “Convert to Plug-In Projects...” and select “Finish”


Configure
Show Unreferenced
Actifsource Folders
Properties
Remove from Context Ctrl+Alt+Shift+Down
Convert to Plug-in Projects...
Setup Actifsource Functions
Remove Actifsource Nature
Remove JavaNature

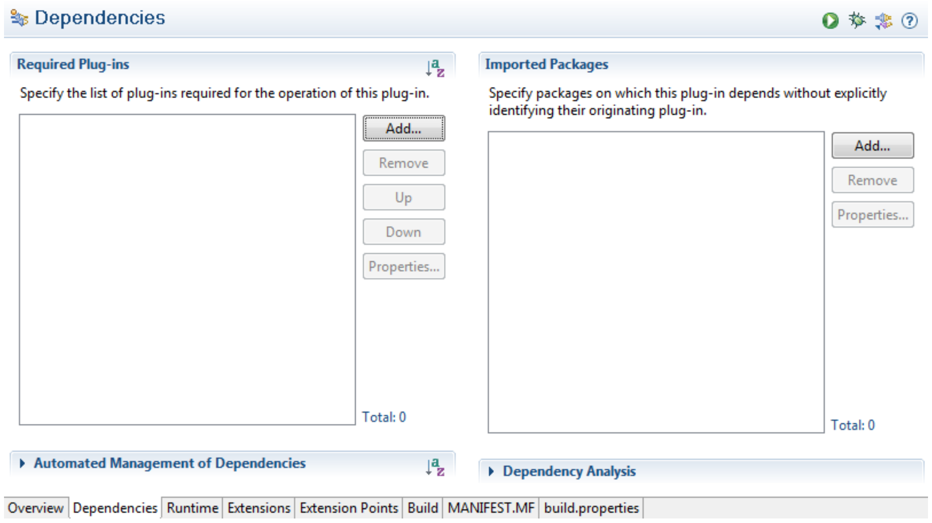
Setup the project

ⓘ Now you have a Plug-In project and need to set the dependencies to actifsource plugin defining the Literal Aspect. In this case it is the “ch.actifsource.core”-Plug-In.

↳ Open the “META-INF” folder and double click on the “MANIFEST.MF” file



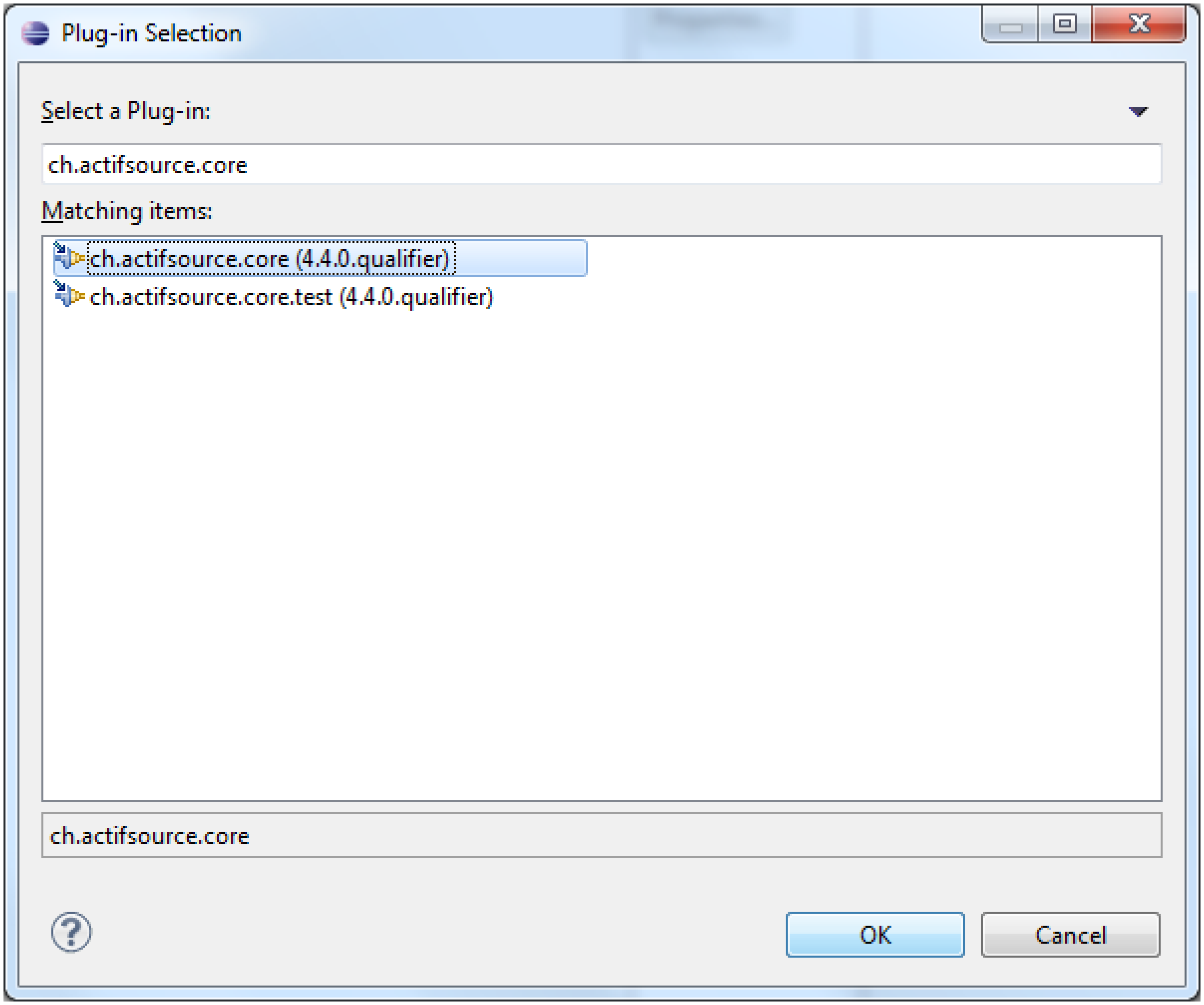
ⓘ Now select the dependencies page



Specify the list of plug-ins required for the operation of this plug-in.

Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.

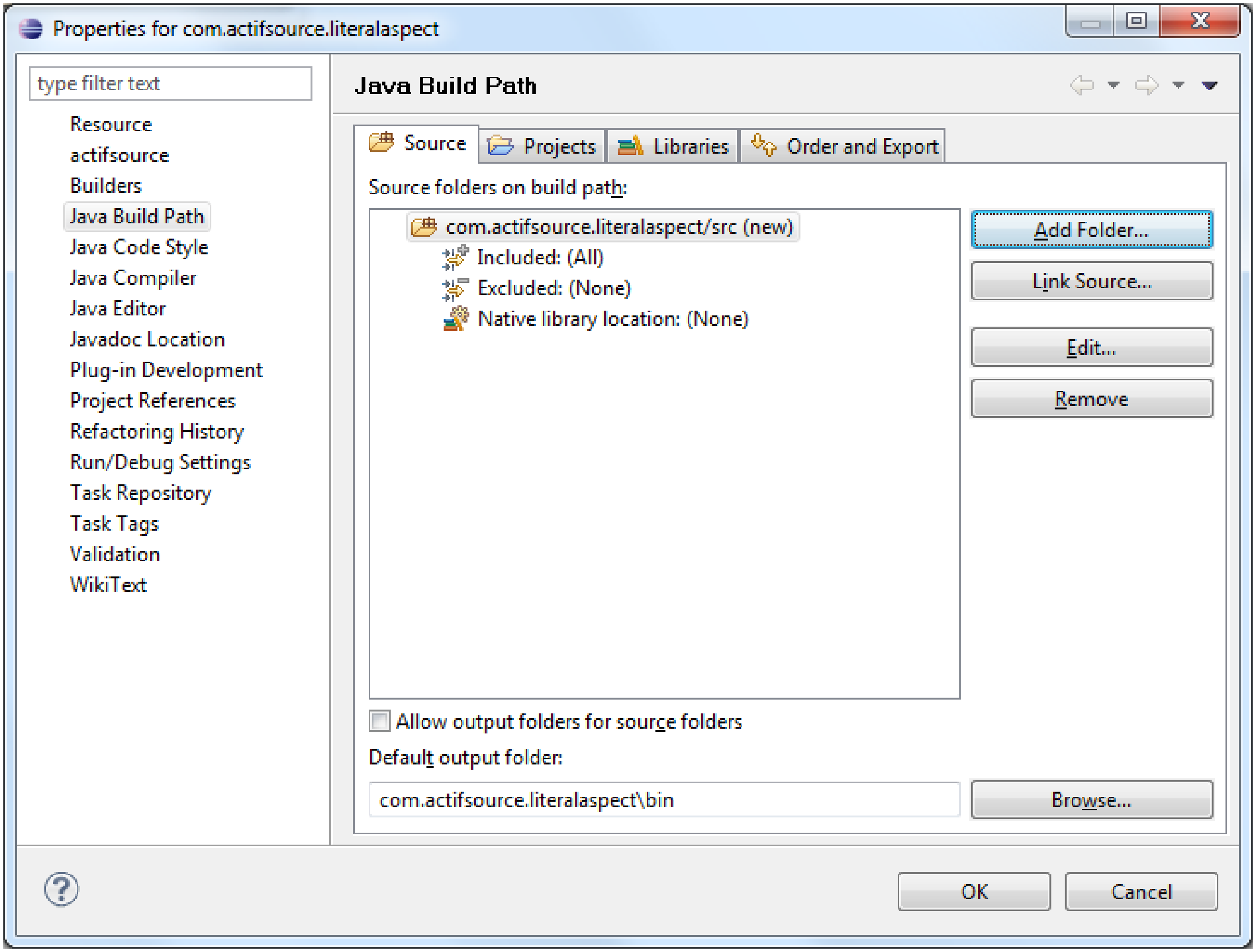
- Click the “Add” button and start typing “ch.actifsource.core” to start filtering the plug-in list.
- Select “ch.actifsource.core”



- Save the manifest-file and close the editor.

• Setup the project

- Switch to the “Java Perspective”
- Right click on the project, select “Properties” and go to the “Java Build Path” preference page
- Remove the project root from the build path and add a new source folder instead



Properties for com.actifsource.literalaspect

Java Build Path

Source folders on build path:

com.actifsource.literalaspect/src (new)
Included: (All)
Excluded: (None)
Native library location: (None)

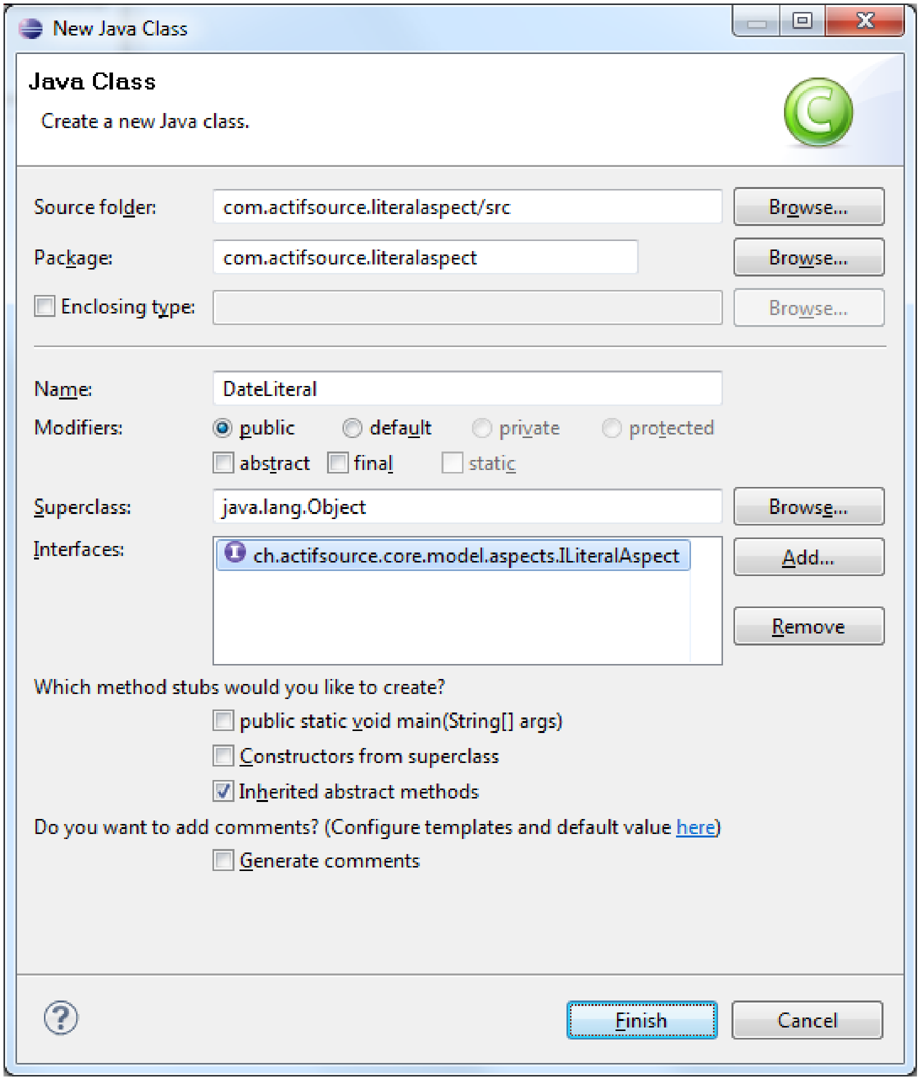
Allow output folders for source folders

Default output folder:

com.actifsource.literalaspect\bin

Part IV: Implement the Literal Aspect

- ⓘ After the project setup, we are ready to create a java class using an actifsource core interface
- ↳ Java Class implementing the “ch.actifsource.core.



New Java Class

Java Class

Create a new Java class.

Source folder:	com.actifsource.literalaspect/src	Browse...
Package:	com.actifsource.literalaspect	Browse...
☐ Enclosing type:		Browse...
Name:	DateLiteral	
Modifiers:	● public ○ default ○ private ○ protected	
	☐ abstract ☐ final ☐ static	
Superclass:	java.lang.Object	Browse...
Interfaces:	ch.actifsource.core.model.aspects.ILiteralAspect	Add... / Remove

Which method stubs would you like to create?

- ☐ public static void main(String[] args)
- ☐ Constructors from superclass
- ☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

- ☐ Generate comments

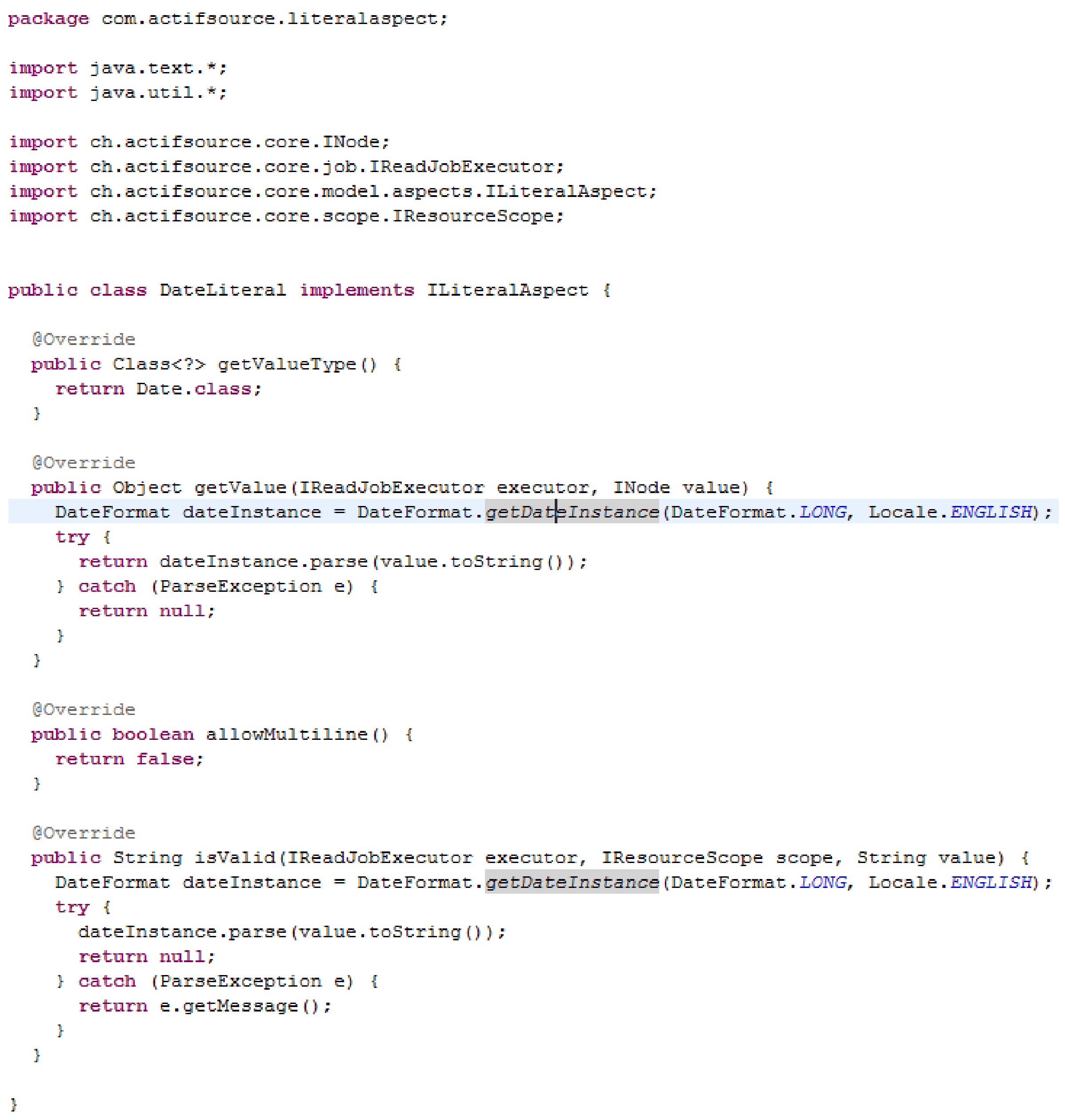
Implement the Literal Aspect

ⓘ There are four methods to implement

getValueType	The java type representing the literal value. In our case the java “Date” class.
getValue	A conversion method to convert the literal value to an instance of the value type, which may return “null”, if the value is invalid
allowMultiLine	True, to allow the use to type in multiple lines using line breaks
isValid	Returning an error string if the value is invalid or “null” if valid

Implement the Literal Aspect

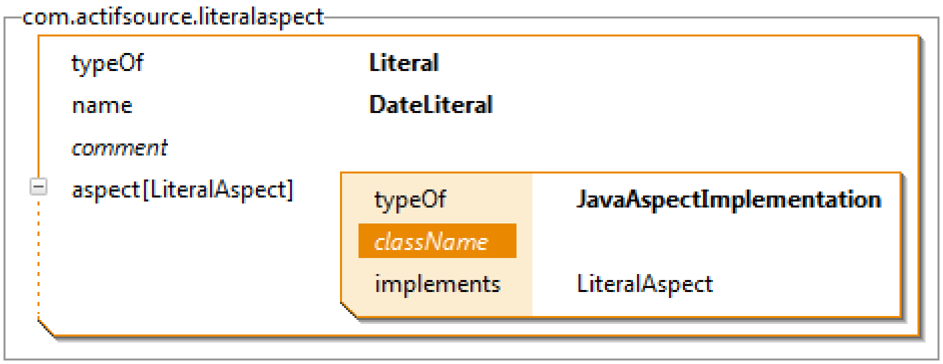
- Implement the LiteralAspect as following



```
package com.actifsource.literalaspect;

import java.text.*;
import java.util.*;

import ch.actifsource.core.INode;
import ch.actifsource.core.job.IReadJobExecutor;
import ch.actifsource.core.model.aspects.ILiteralAspect;
import ch.actifsource.core.scope.IResourceScope;


public class DateLiteral implements ILiteralAspect {

  @Override
  public Class<?> getValueType() {
    return Date.class;
  }

  @Override
  public Object getValue(IReadJobExecutor executor, INode value) {
    DateFormat dateInstance = DateFormat.getDateInstance(DateFormat.LONG, Locale.ENGLISH);
    try {
      return dateInstance.parse(value.toString());
    } catch (ParseException e) {
      return null;
    }
  }

  @Override
  public boolean allowMultiline() {
    return false;
  }

  @Override
  public String isValid(IReadJobExecutor executor, IResourceScope scope, String value) {
    DateFormat dateInstance = DateFormat.getDateInstance(DateFormat.LONG, Locale.ENGLISH);
    try {
      dateInstance.parse(value.toString());
      return null;
    } catch (ParseException e) {
      return e.getMessage();
    }
  }

}
```

Part V: Using the Literal Aspect

- Go back to the “DateLiteral” and add new value for the aspect [LiteralAspect] relation.

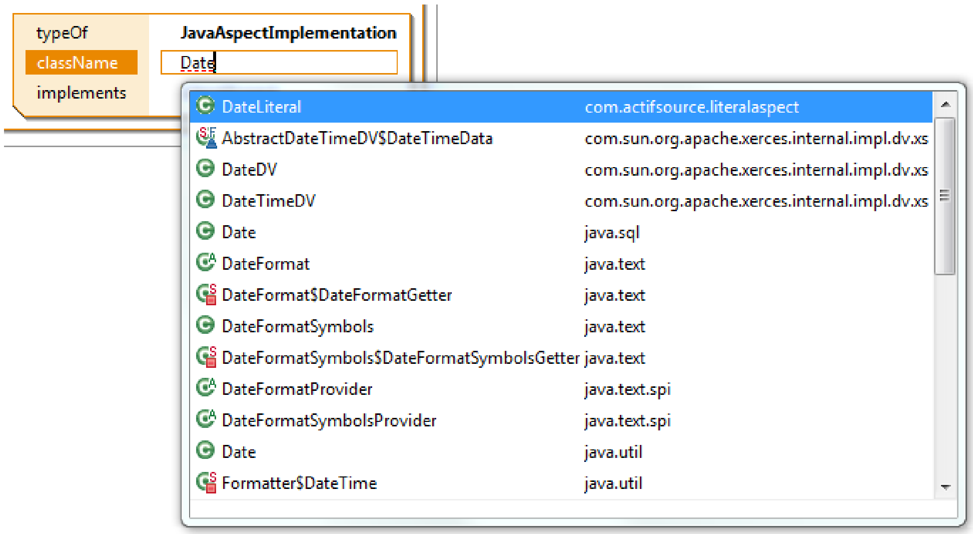


com.actifsource.literalaspect

typeOf	**Literal**
name	**DateLiteral**
comment	
aspect[LiteralAspect]	

typeOf	**JavaAspectImplementation**
className	
implements	LiteralAspect

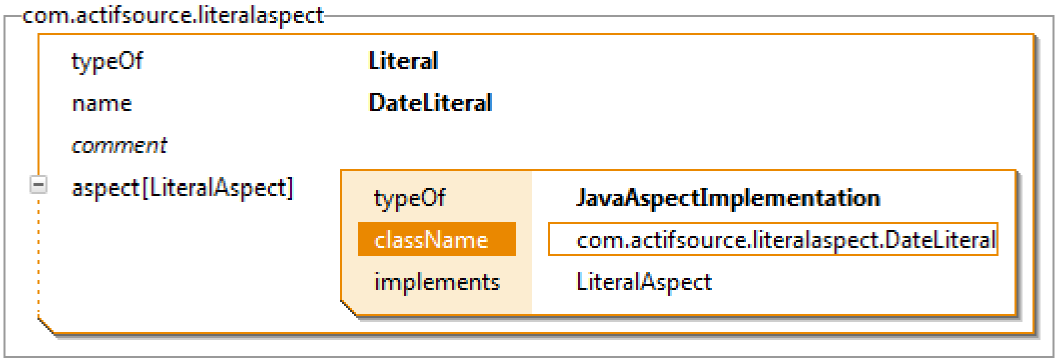
- Type “Date” and use the content assist to select the class.



typeOf	**JavaAspectImplementation**
className	Date
implements	

```
DateLiteral                                  com.actifsource.literalaspect
AbstractDateTimeDV$DateTimeData              com.sun.org.apache.xerces.internal.impl.dv.xs
DateDV                                       com.sun.org.apache.xerces.internal.impl.dv.xs
DateTimeDV                                   com.sun.org.apache.xerces.internal.impl.dv.xs
Date                                         java.sql
DateFormat                                   java.text
DateFormat$DateFormatGetter                  java.text
DateFormatSymbols                            java.text
DateFormatSymbols$DateFormatSymbolsGetter    java.text
DateFormatProvider                           java.text.spi
DateFormatSymbolsProvider                    java.text.spi
Date                                         java.util
Formatter$DateTime                           java.util
```

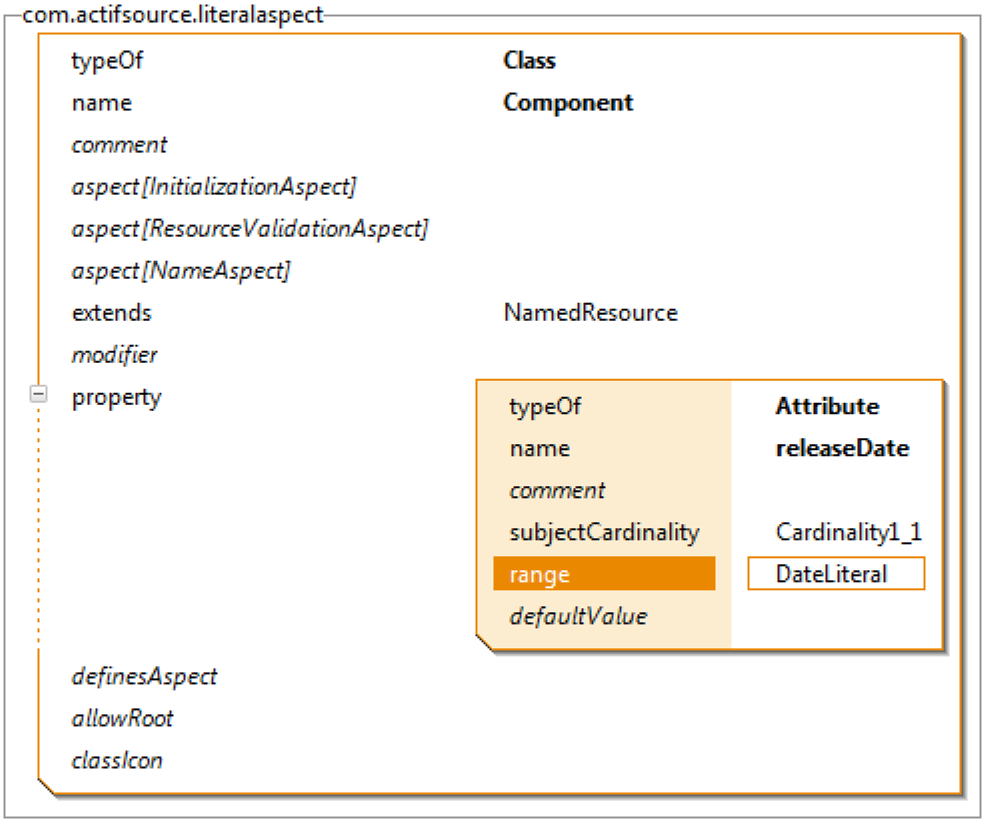
ⓘ The Literal is now ready to use, since we created the Literal early with the content assist, the attribute in the “Component” is already using it



com.actifsource.literalaspect

typeOf	**Literal**
name	**DateLiteral**
comment	
aspect[LiteralAspect]	

typeOf	**JavaAspectImplementation**
className	com.actifsource.literalaspect.DateLiteral
implements	LiteralAspect

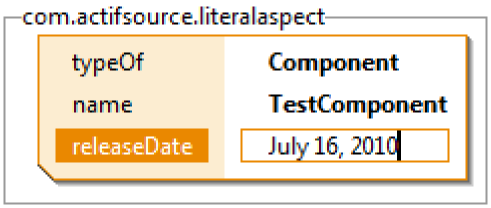


com.actifsource.literalaspect

typeOf	**Class**
name	**Component**
comment	
aspect[InitializationAspect]	
aspect[ResourceValidationAspect]	
aspect[NameAspect]	
extends	NamedResource
modifier	
property	
definesAspect	
allowRoot	
classIcon	

typeOf	**Attribute**
name	**releaseDate**
comment	
subjectCardinality	Cardinality1_1
range	DateLiteral
defaultValue	

↳ To test the literal, just create a new instance of the “Component” and define a “releaseDate”.


com.actifsource.literalaspect

typeOf	**Component**
name	**TestComponent**
releaseDate	July 16, 2010