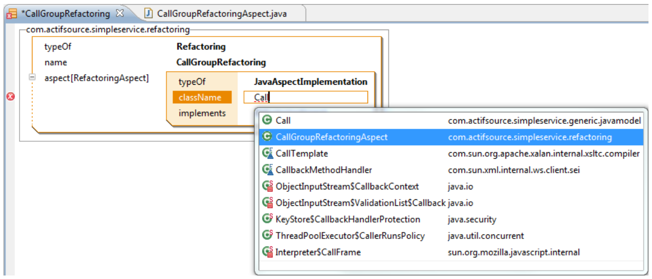
# Tutorial

Refactoring

| Tutorial | Actifsource Tutorial – Refactoring |
|---|---|
| **Required Time** | • 45 Minutes |
| **Prerequisites** | • Actifsource Tutorial – Installing Actifsource<br>• Actifsource Tutorial – Simple Service |
| **Goal** | • Writing an aspect for refactoring instances<br>• Refactor instances to match the new specification<br>• Update templates |
| **Topics covered** | • Create and register a refactoring aspect<br>• Update the simple service model<br>• Write a simple refactoring<br>• Execute a refactoring<br>• Update the templates |
| **Notation** | ↳ To do<br>ⓘ Information<br>• **Bold**: Terms from actifsource or other technologies and tools<br>• **<u>Bold underlined</u>**: actifsource Resources<br>• <u>Underlined</u>: User Resources<br>• *<u>UnderlinedItalics</u>*: Resource Functions<br>• `Monospaced`: User input<br>• *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**<br>Täfernstrasse 37<br>5405 Baden-Dättwil<br>Switzerland<br><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 and register a refactoring aspect



*CallGroupRefactoring | CallGroupRefactoringAspect.java

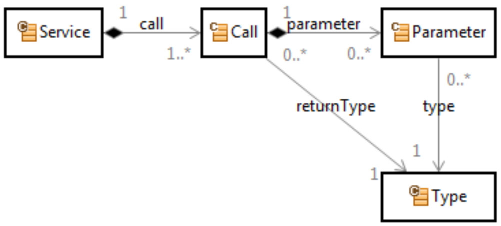
com.actifsource.simpleservice.refactoring

| | |
|---|---|
| typeOf | **Refactoring** |
| name | **CallGroupRefactoring** |
| aspect[RefactoringAspect] | |

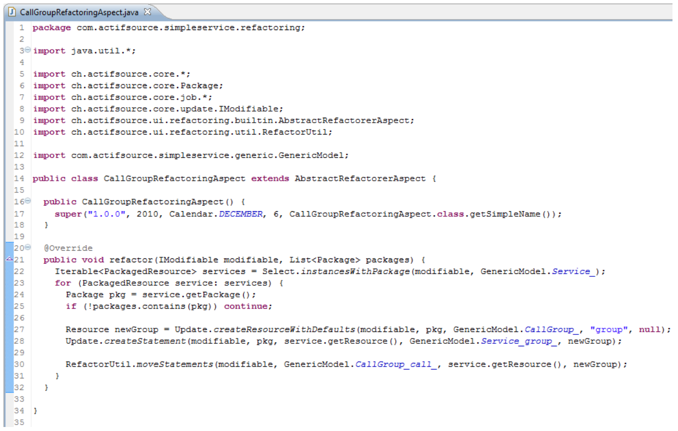
| | |
|---|---|
| typeOf | **JavaAspectImplementation** |
| className | Call |
| implements | |

| | |
|---|---|
| Call | com.actifsource.simpleservice.generic.javamodel |
| CallGroupRefactoringAspect | com.actifsource.simpleservice.refactoring |
| CallTemplate | com.sun.org.apache.xalan.internal.xsltc.compiler |
| CallbackMethodHandler | com.sun.xml.internal.ws.client.sei |
| ObjectInputStream$CallbackContext | java.io |
| ObjectInputStream$ValidationList$Callback | java.io |
| KeyStore$CallbackHandlerProtection | java.security |
| ThreadPoolExecutor$CallerRunsPolicy | java.util.concurrent |
| Interpreter$CallFrame | sun.org.mozilla.javascript.internal |

- Update the simple service model



- Write a simple refactoring

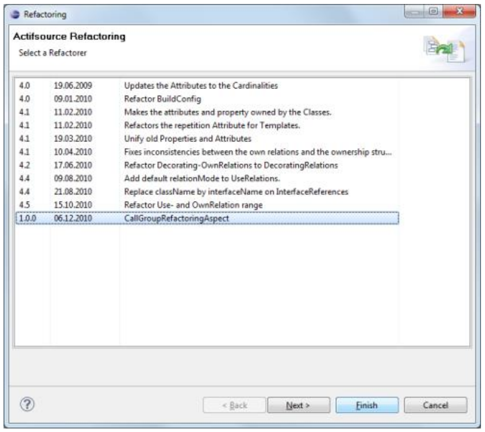


CallGroupRefactoringAspect.java

```java
package com.actifsource.simpleservice.refactoring;

import java.util.*;

import ch.actifsource.core.*;
import ch.actifsource.core.Package;
import ch.actifsource.core.job.*;
import ch.actifsource.core.update.IModifiable;
import ch.actifsource.ui.refactoring.builtin.AbstractRefactorerAspect;
import ch.actifsource.ui.refactoring.util.RefactorUtil;

import com.actifsource.simpleservice.generic.GenericModel;

public class CallGroupRefactoringAspect extends AbstractRefactorerAspect {

  public CallGroupRefactoringAspect() {
    super("1.0.0", 2010, Calendar.DECEMBER, 6, CallGroupRefactoringAspect.class.getSimpleName());
  }

  @Override
  public void refactor(IModifiable modifiable, List<Package> packages) {
    Iterable<PackagedResource> services = Select.instancesWithPackage(modifiable, GenericModel.Service_);
    for (PackagedResource service: services) {
      Package pkg = service.getPackage();
      if (!packages.contains(pkg)) continue;

      Resource newGroup = Update.createResourceWithDefaults(modifiable, pkg, GenericModel.CallGroup_, "group", null);
      Update.createStatement(modifiable, pkg, service.getResource(), GenericModel.Service_group_, newGroup);

      RefactorUtil.moveStatements(modifiable, GenericModel.CallGroup_call_, service.getResource(), newGroup);
    }
  }

}
```

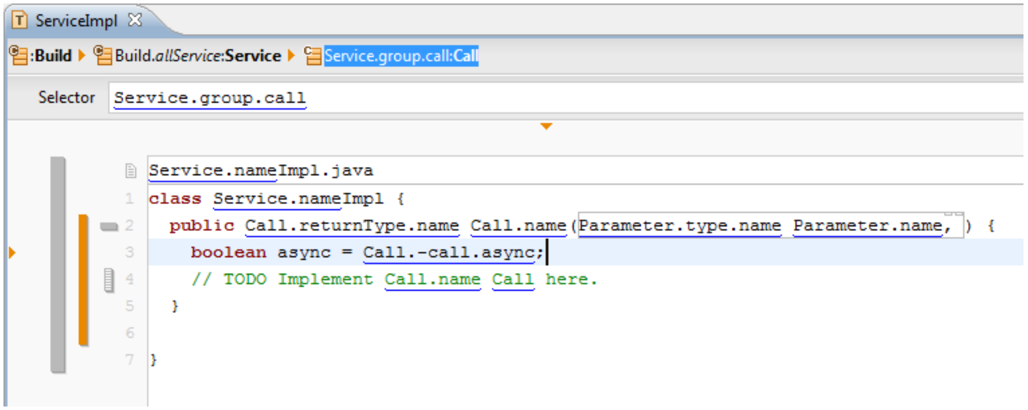
## Overview

- Execute the refactoring



| | | |
|---|---|---|
| 4.0 | 19.06.2009 | Updates the Attributes to the Cardinalities |
| 4.0 | 09.01.2010 | Refactor BuildConfig |
| 4.1 | 11.02.2010 | Makes the attributes and property owned by the Classes. |
| 4.1 | 11.02.2010 | Refactors the repetition Attribute for Templates. |
| 4.1 | 19.03.2010 | Unify old Properties and Attributes |
| 4.1 | 10.04.2010 | Fixes inconsistencies between the own relations and the ownership stru... |
| 4.2 | 17.06.2010 | Refactor Decorating-OwnRelations to DecoratingRelations |
| 4.4 | 09.08.2010 | Add default relationMode to UseRelations. |
| 4.4 | 21.08.2010 | Replace className by interfaceName on InterfaceReferences |
| 4.5 | 15.10.2010 | Refactor Use- and OwnRelation range |
| 1.0.0 | 06.12.2010 | CallGroupRefactoringAspect |

- Update the templates

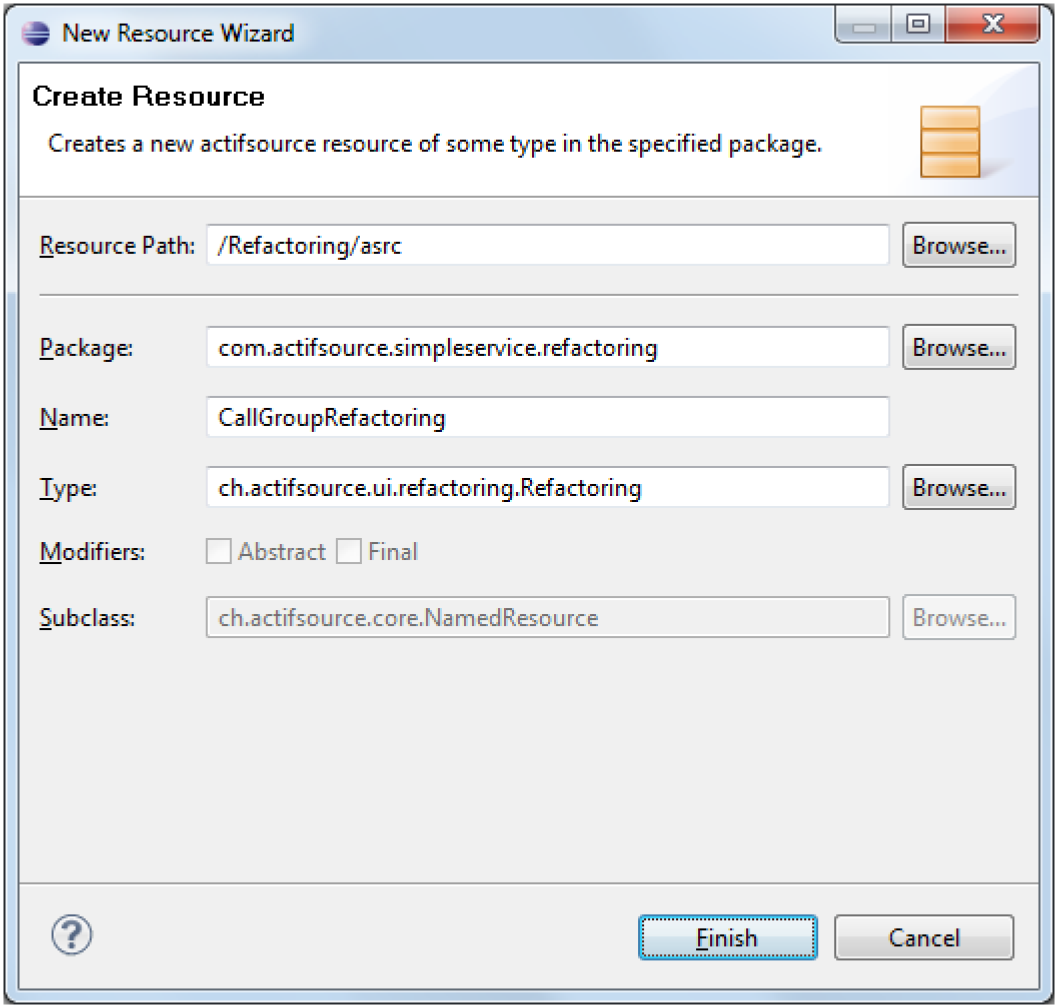


Selector `Service.group.call`

```
Service.nameImpl.java
class Service.nameImpl {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {
    boolean async = Call.-call.async;
    // TODO Implement Call.name Call here.
  }

}
```

# Part I:
# Create and register a refactoring aspect

- Open the project created during the *Actifsource Tutorial – Simple Service*
- Create a **Refactoring** instance


New Resource Wizard

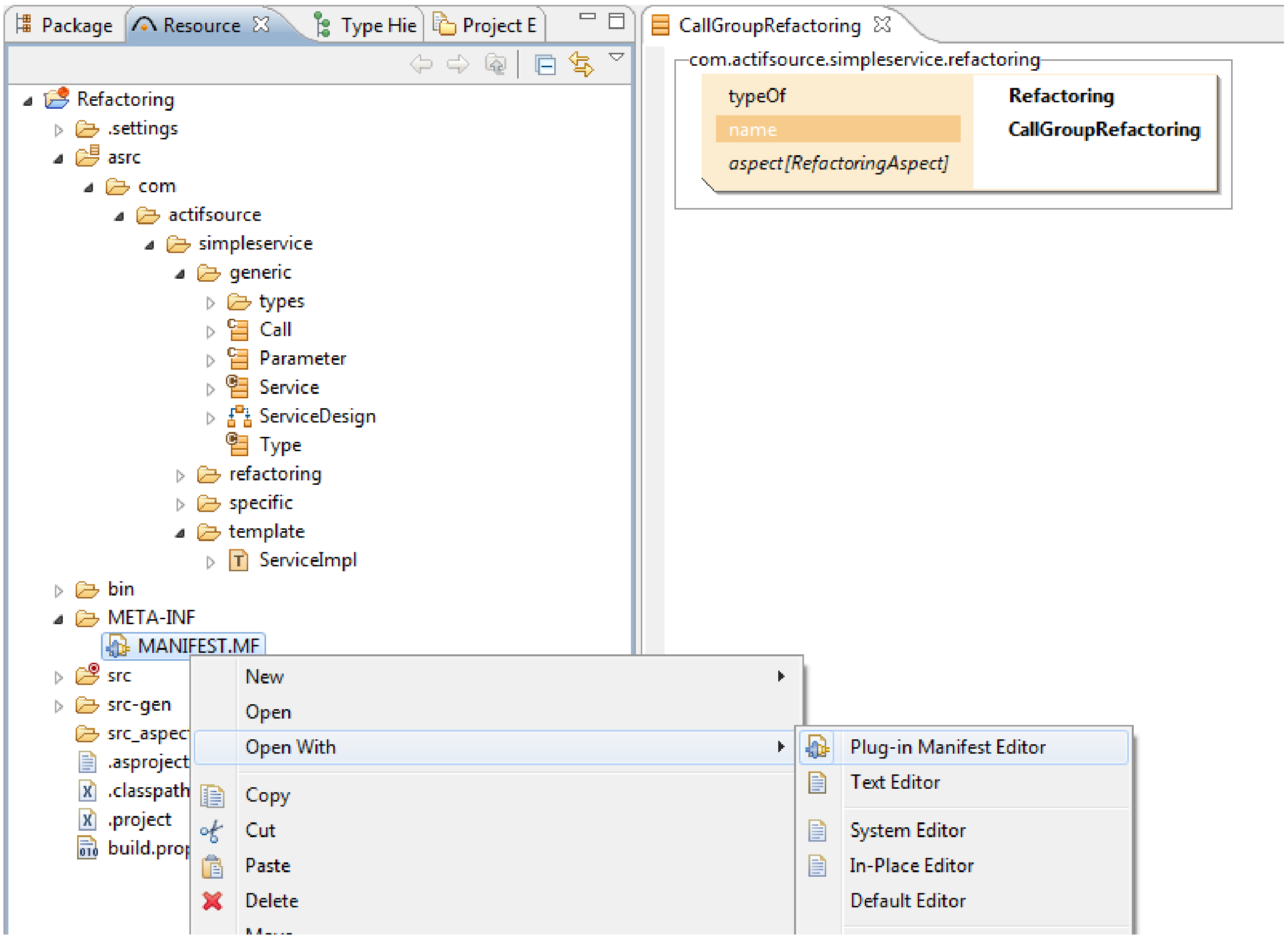
**Create Resource**

Creates a new actifsource resource of some type in the specified package.

| | | |
|---|---|---|
| Resource Path: | /Refactoring/asrc | Browse... |
| Package: | com.actifsource.simpleservice.refactoring | Browse... |
| Name: | CallGroupRefactoring | |
| Type: | ch.actifsource.ui.refactoring.Refactoring | Browse... |
| Modifiers: | ☐ Abstract ☐ Final | |
| Subclass: | ch.actifsource.core.NamedResource | Browse... |

## Create and register a refactoring aspect

- Open the *MANIFEST.MF*

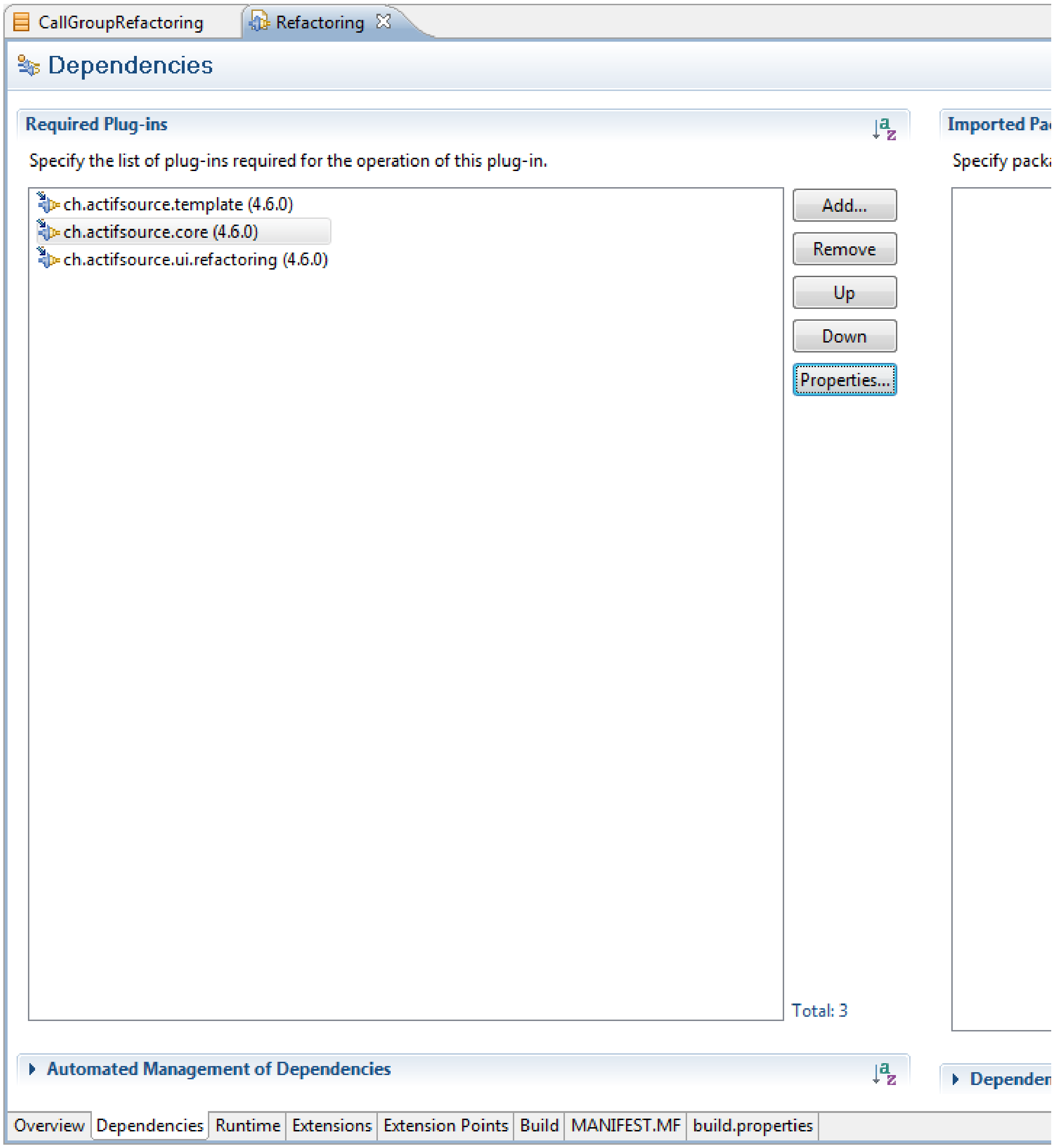


com.actifsource.simpleservice.refactoring

| | |
|---|---|
| typeOf | Refactoring |
| name | CallGroupRefactoring |
| *aspect [RefactoringAspect]* | |

## Create and register a refactoring aspect

- Add a dependency to the **ch.actifsource.ui.refactoring**-Plugin



CallGroupRefactoring | Refactoring

Dependencies

Required Plug-ins

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

ch.actifsource.template (4.6.0)
ch.actifsource.core (4.6.0)
ch.actifsource.ui.refactoring (4.6.0)

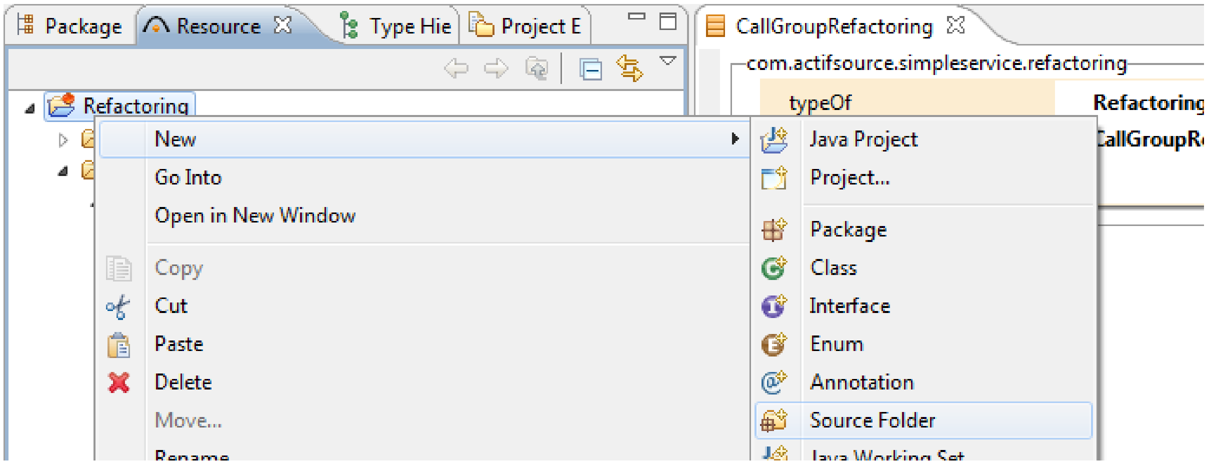
Total: 3

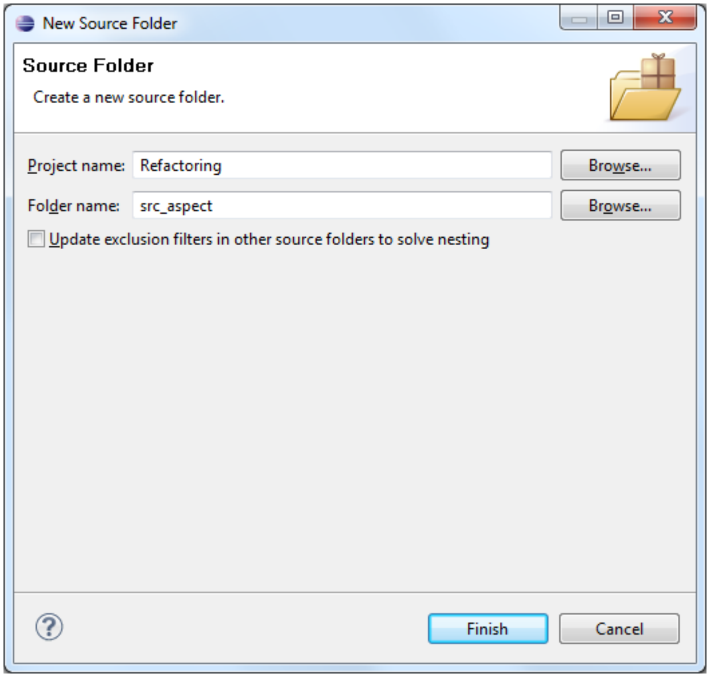
Imported Pa

Specify pack

## Create and register a refactoring aspect

- Create a new *java source folder* for the aspect implementation





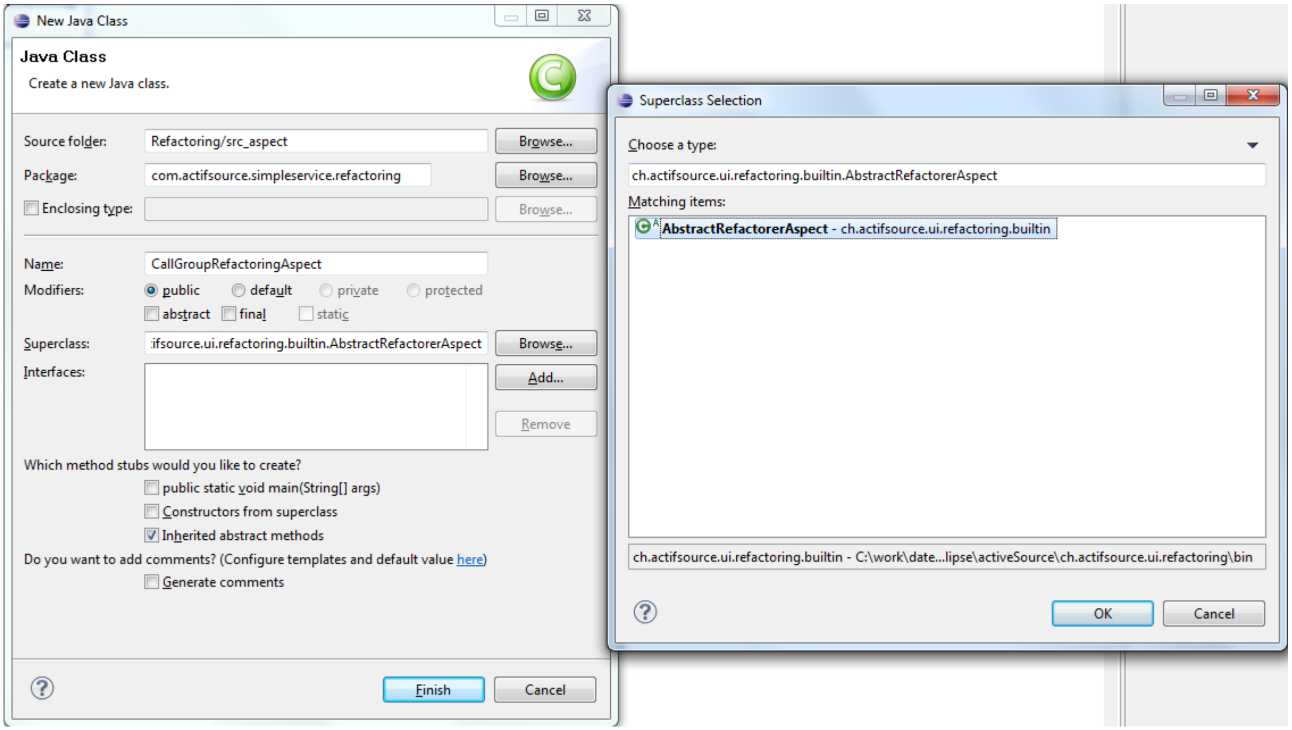
Project name: Refactoring

Folder name: src_aspect

☐ Update exclusion filters in other source folders to solve nesting

## Create and register a refactoring aspect

↳ Create a new *java class* extending the **AbstractRefactorerAspect** class


New Java Class

**Java Class**

Create a new Java class.

Source folder: Refactoring/src_aspect

Package: com.actifsource.simpleservice.refactoring

Enclosing type:

Name: CallGroupRefactoringAspect

Modifiers: [✓ public] default private protected

abstract final static

Superclass: ifsource.ui.refactoring.builtin.AbstractRefactorerAspect

Interfaces:

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

Superclass Selection

Choose a type: ch.actifsource.ui.refactoring.builtin.AbstractRefactorerAspect

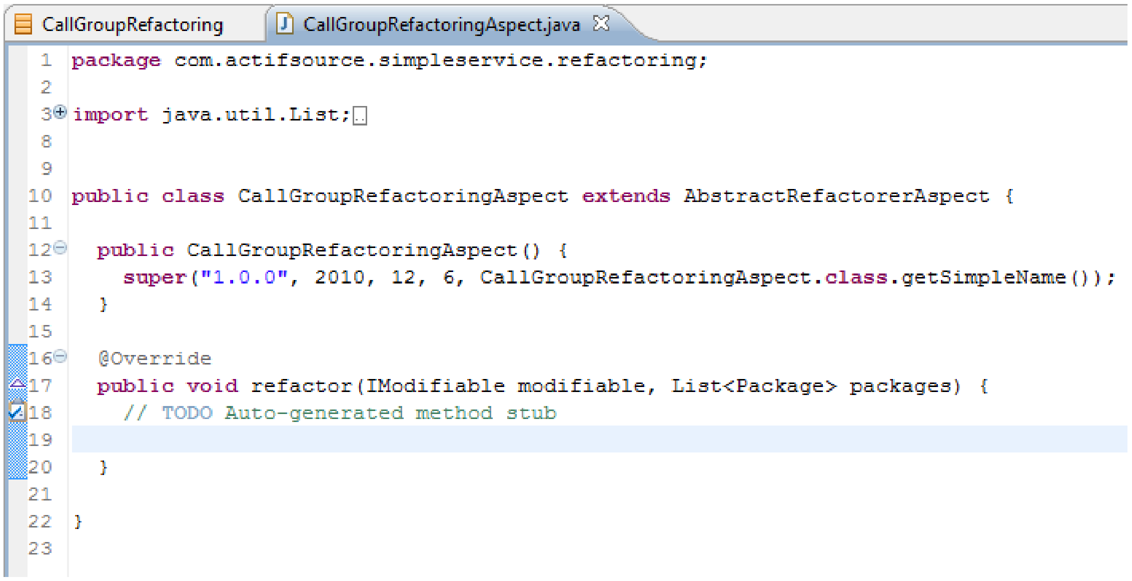
Matching items:

AbstractRefactorerAspect - ch.actifsource.ui.refactoring.builtin

ch.actifsource.ui.refactoring.builtin - C:\work\date...lipse\activeSource\ch.actifsource.ui.refactoring\bin

## Create and register a refactoring aspect

Add a default constructor calling the base constructor

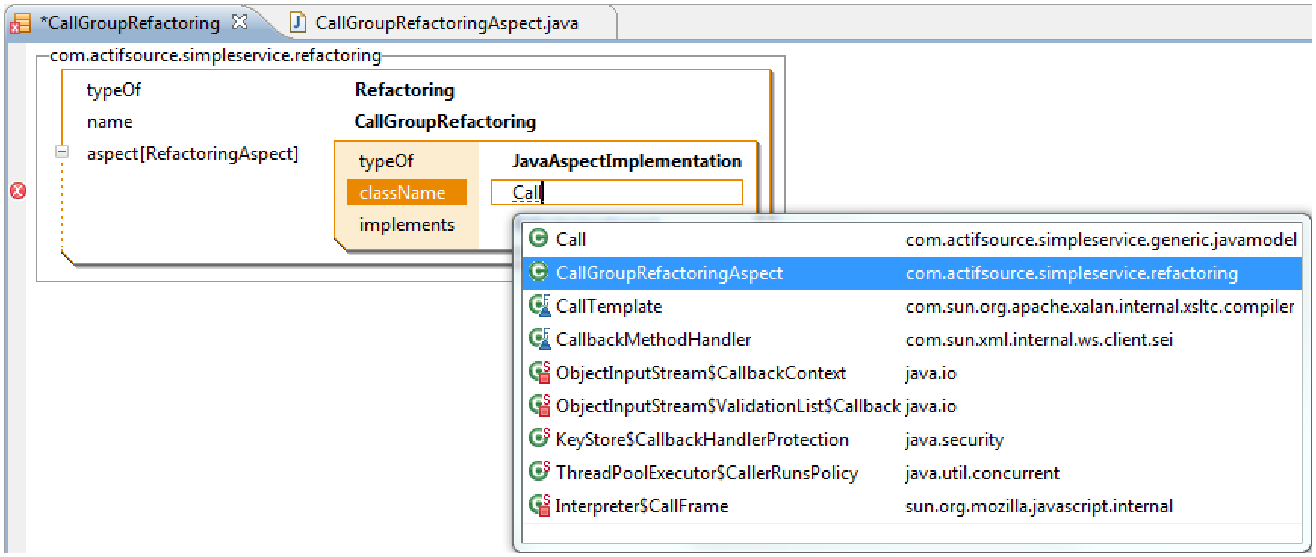


```java
package com.actifsource.simpleservice.refactoring;

import java.util.List;


public class CallGroupRefactoringAspect extends AbstractRefactorerAspect {

  public CallGroupRefactoringAspect() {
    super("1.0.0", 2010, 12, 6, CallGroupRefactoringAspect.class.getSimpleName());
  }

  @Override
  public void refactor(IModifiable modifiable, List<Package> packages) {
    // TODO Auto-generated method stub

  }

}
```

The first parameter defines a version string for the aspect
The following three parameters are used to specify the date when the refactoring was written.
The last parameter is the name.
The arguments are shown in the dialog and used to sort the refactoring aspects.

## Create and register a refactoring aspect

- Register the java class in the **Refactoring** instance



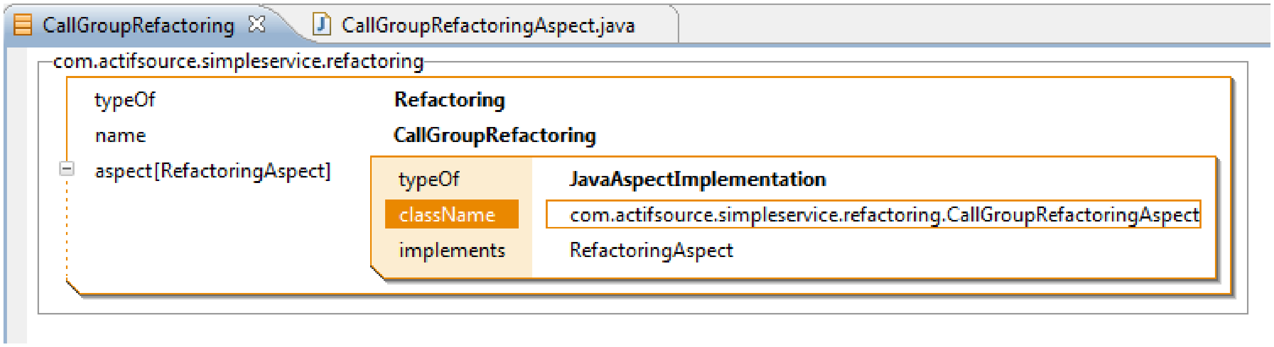
*CallGroupRefactoring | CallGroupRefactoringAspect.java

com.actifsource.simpleservice.refactoring

| | |
|---|---|
| typeOf | **Refactoring** |
| name | **CallGroupRefactoring** |
| aspect[RefactoringAspect] | |

| | |
|---|---|
| typeOf | **JavaAspectImplementation** |
| className | Call |
| implements | |

| | |
|---|---|
| Call | com.actifsource.simpleservice.generic.javamodel |
| CallGroupRefactoringAspect | com.actifsource.simpleservice.refactoring |
| CallTemplate | com.sun.org.apache.xalan.internal.xsltc.compiler |
| CallbackMethodHandler | com.sun.xml.internal.ws.client.sei |
| ObjectInputStream$CallbackContext | java.io |
| ObjectInputStream$ValidationList$Callback | java.io |
| KeyStore$CallbackHandlerProtection | java.security |
| ThreadPoolExecutor$CallerRunsPolicy | java.util.concurrent |
| Interpreter$CallFrame | sun.org.mozilla.javascript.internal |



CallGroupRefactoring | CallGroupRefactoringAspect.java

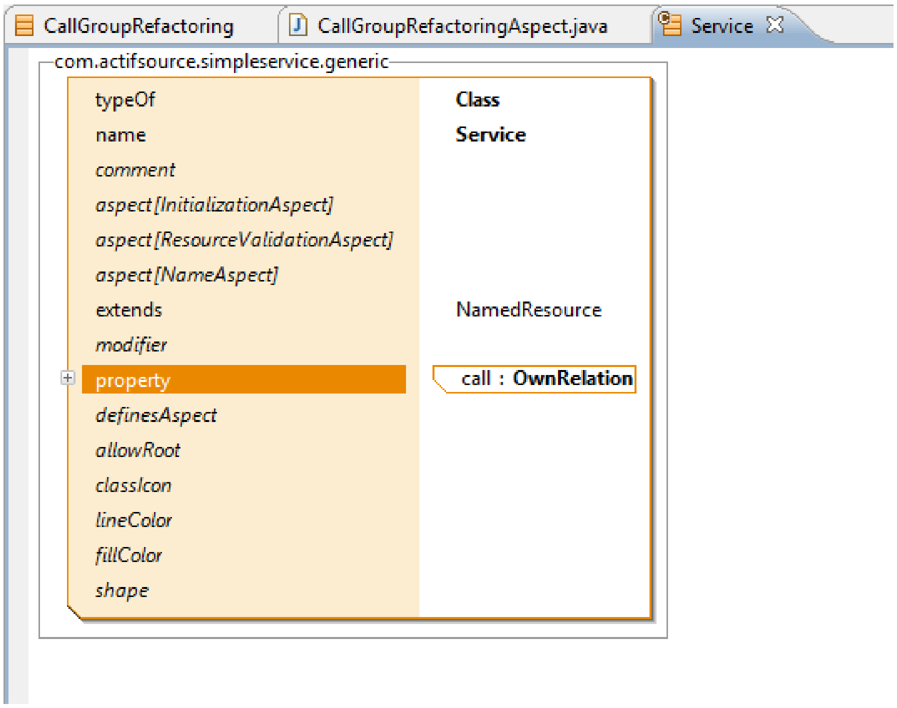
com.actifsource.simpleservice.refactoring

| | |
|---|---|
| typeOf | **Refactoring** |
| name | **CallGroupRefactoring** |
| aspect[RefactoringAspect] | |

| | |
|---|---|
| typeOf | **JavaAspectImplementation** |
| className | com.actifsource.simpleservice.refactoring.CallGroupRefactoringAspect |
| implements | RefactoringAspect |

# Part II: Update the simple service model

- Open the Service class
- Select the call relation and cut the resource by using the *clipboard* (Ctrl+X)

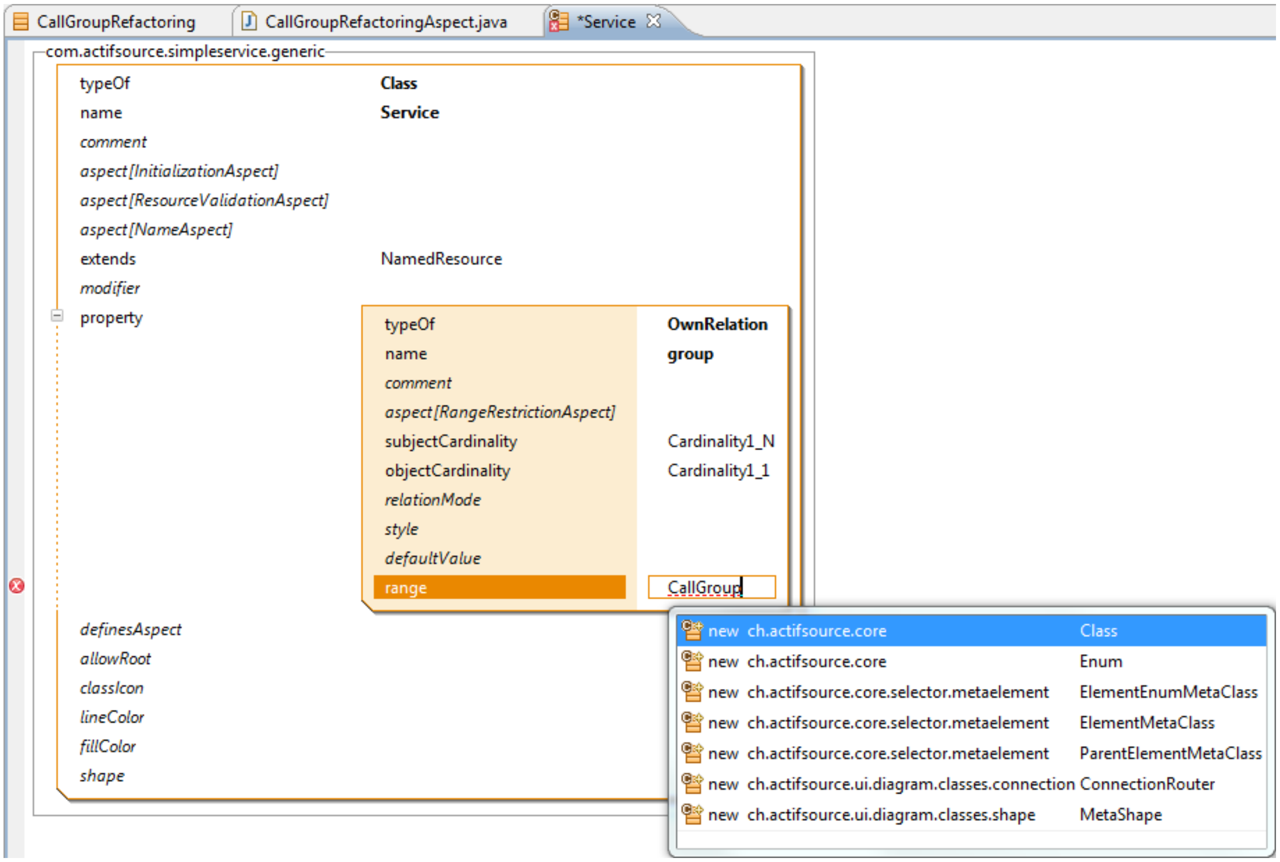


com.actifsource.simpleservice.generic

| | |
|---|---|
| typeOf | **Class** |
| name | **Service** |
| *comment* | |
| *aspect [InitializationAspect]* | |
| *aspect [ResourceValidationAspect]* | |
| *aspect [NameAspect]* | |
| extends | NamedResource |
| *modifier* | |
| property | call : **OwnRelation** |
| *definesAspect* | |
| *allowRoot* | |
| *classIcon* | |
| *lineColor* | |
| *fillColor* | |
| *shape* | |

## Update the simple service model

- Create a new **Relation** named “group” with a new **Class** named “CallGroup” used as the range



CallGroupRefactoring | CallGroupRefactoringAspect.java | *Service

com.actifsource.simpleservice.generic

| | |
|---|---|
| typeOf | **Class** |
| name | **Service** |
| *comment* | |
| *aspect[InitializationAspect]* | |
| *aspect[ResourceValidationAspect]* | |
| *aspect[NameAspect]* | |
| extends | NamedResource |
| *modifier* | |
| property | |
| *definesAspect* | |
| *allowRoot* | |
| *classIcon* | |
| *lineColor* | |
| *fillColor* | |
| *shape* | |

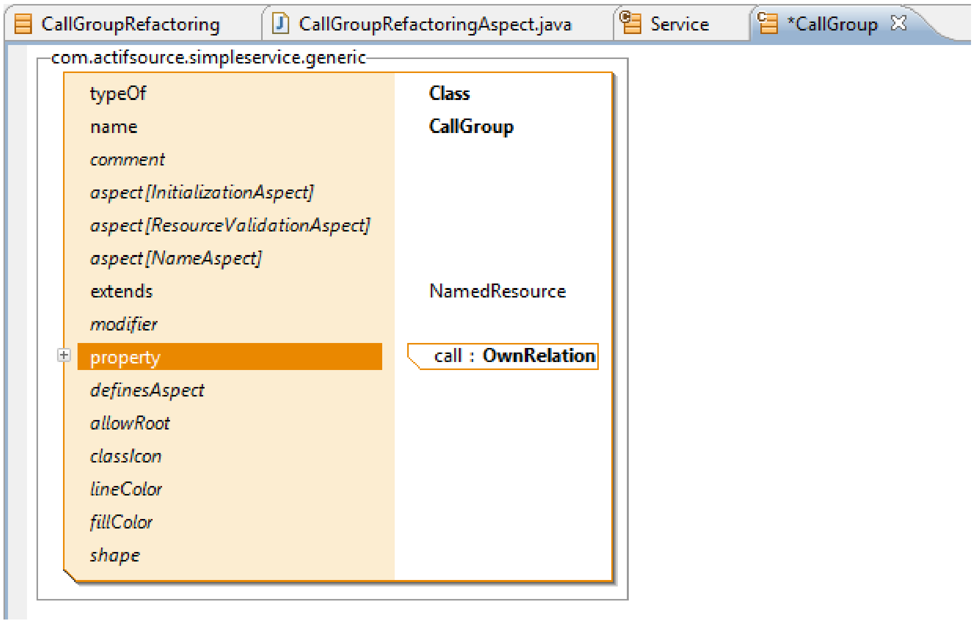
property:

| | |
|---|---|
| typeOf | **OwnRelation** |
| name | **group** |
| *comment* | |
| *aspect[RangeRestrictionAspect]* | |
| subjectCardinality | Cardinality1_N |
| objectCardinality | Cardinality1_1 |
| *relationMode* | |
| *style* | |
| *defaultValue* | |
| range | CallGroup |

```
new  ch.actifsource.core                                   Class
new  ch.actifsource.core                                   Enum
new  ch.actifsource.core.selector.metaelement              ElementEnumMetaClass
new  ch.actifsource.core.selector.metaelement              ElementMetaClass
new  ch.actifsource.core.selector.metaelement              ParentElementMetaClass
new  ch.actifsource.ui.diagram.classes.connection ConnectionRouter
new  ch.actifsource.ui.diagram.classes.shape               MetaShape
```

## Update the simple service model

- Paste the call relation from the *clipboard* into the new Callgroup

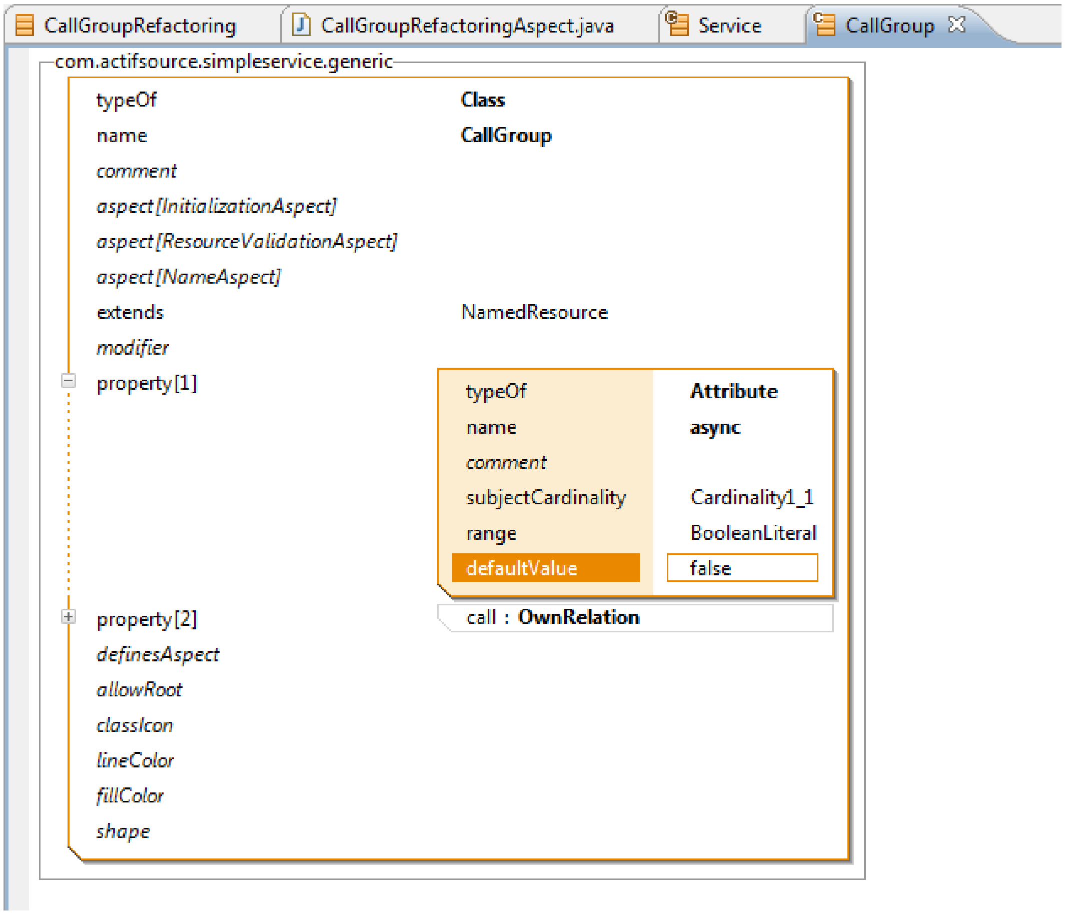


com.actifsource.simpleservice.generic

|  |  |
|---|---|
| typeOf | **Class** |
| name | **CallGroup** |
| *comment* | |
| *aspect [InitializationAspect]* | |
| *aspect [ResourceValidationAspect]* | |
| *aspect [NameAspect]* | |
| extends | NamedResource |
| *modifier* | |
| property | call : **OwnRelation** |
| *definesAspect* | |
| *allowRoot* | |
| *classIcon* | |
| *lineColor* | |
| *fillColor* | |
| *shape* | |

## Update the simple service model

Add an additional **Attribute** named “async”



com.actifsource.simpleservice.generic

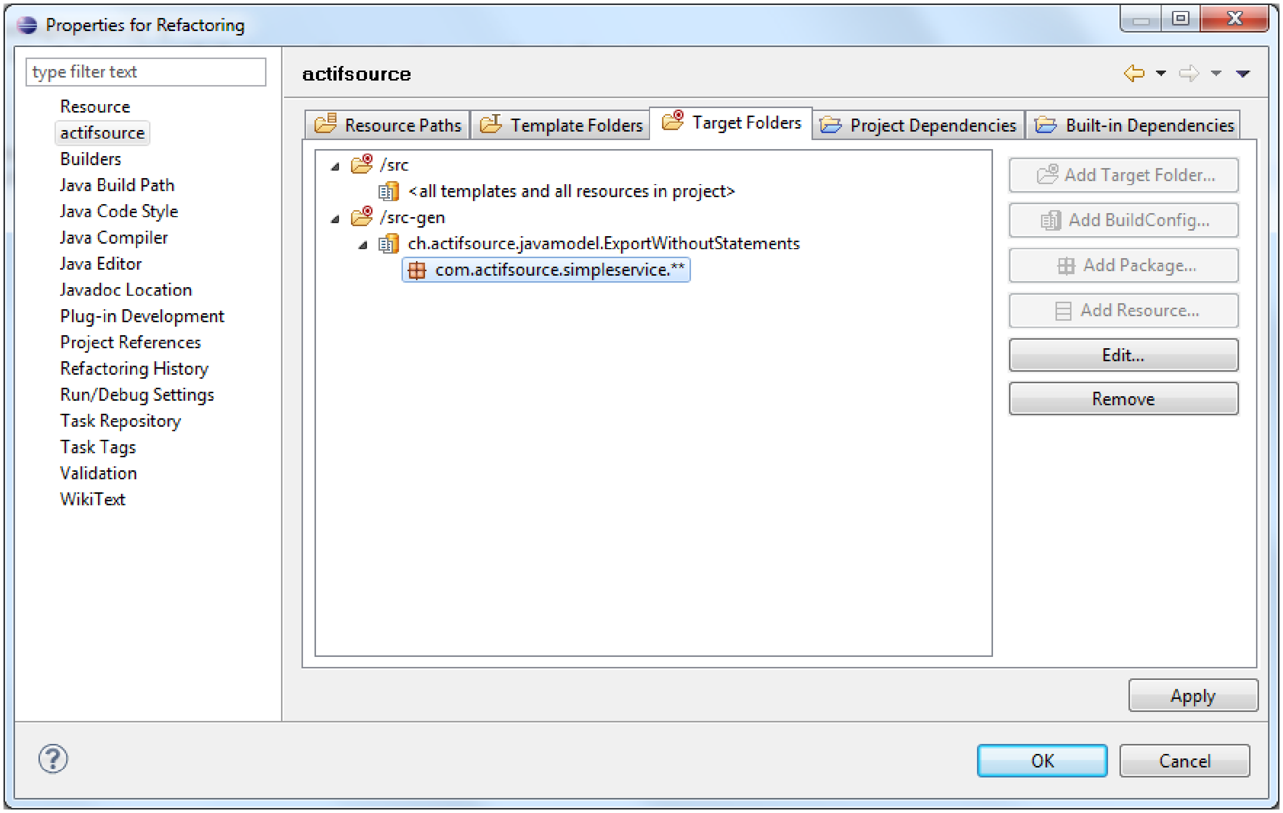
| | |
|---|---|
| typeOf | **Class** |
| name | **CallGroup** |
| *comment* | |
| *aspect [InitializationAspect]* | |
| *aspect [ResourceValidationAspect]* | |
| *aspect [NameAspect]* | |
| extends | NamedResource |
| *modifier* | |
| property[1] | typeOf **Attribute**; name **async**; *comment*; subjectCardinality Cardinality1_1; range BooleanLiteral; defaultValue false |
| property[2] | call : **OwnRelation** |
| *definesAspect* | |
| *allowRoot* | |
| *classIcon* | |
| *lineColor* | |
| *fillColor* | |
| *shape* | |

We only use an attribute to keep the tutorial simple. An alternative would be to create a subclass of CallGroup instead.

# Part III: Write a simple refactoring

ⓘ First we want to have more convenient access to the resource guids.

↳ Add a the **ExportWithoutStatements** buildconfig to the *src-gen* **target folder** for exporting java classes for the *SimpleService* packages


Properties for Refactoring

type filter text

Resource
actifsource
Builders
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Plug-in Development
Project References
Refactoring History
Run/Debug Settings
Task Repository
Task Tags
Validation
WikiText

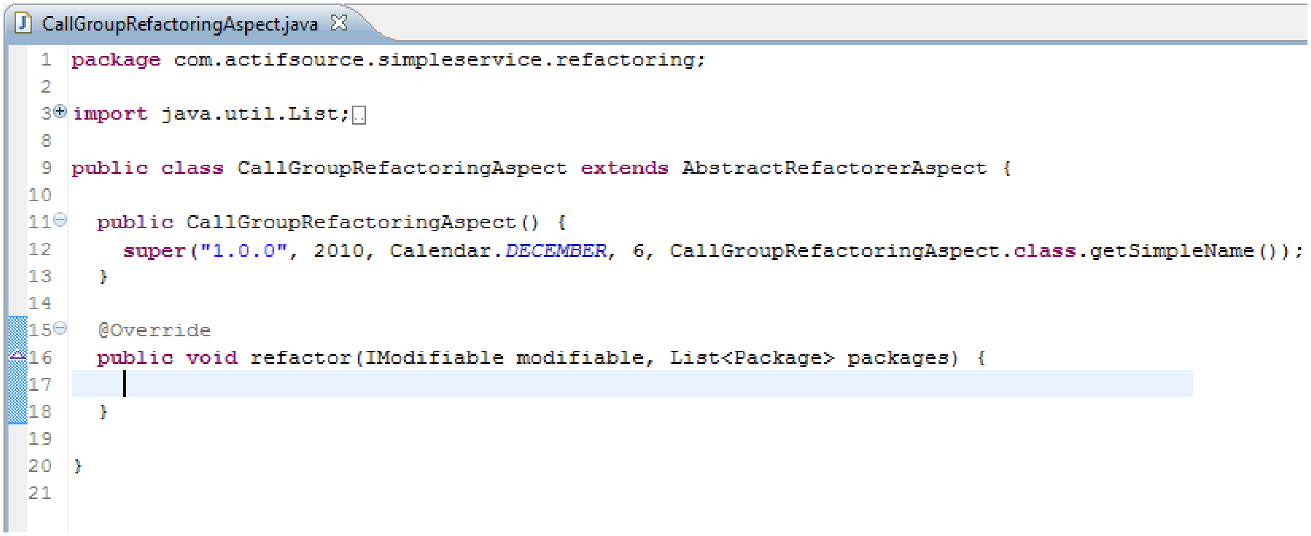
actifsource

Resource Paths | Template Folders | Target Folders | Project Dependencies | Built-in Dependencies

```
/src
    <all templates and all resources in project>
/src-gen
    ch.actifsource.javamodel.ExportWithoutStatements
        com.actifsource.simpleservice.**
```

ⓘ This will generate a class for each package containing a constant for each **resource**.

Implement the *refactor* method



CallGroupRefactoringAspect.java

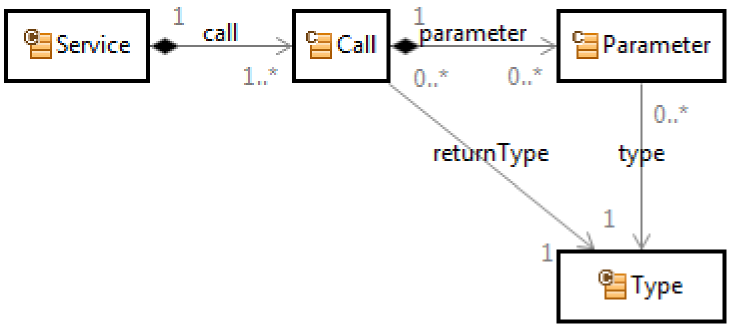
```
package com.actifsource.simpleservice.refactoring;

import java.util.List;

public class CallGroupRefactoringAspect extends AbstractRefactorerAspect {

  public CallGroupRefactoringAspect() {
    super("1.0.0", 2010, Calendar.DECEMBER, 6, CallGroupRefactoringAspect.class.getSimpleName());
  }

  @Override
  public void refactor(IModifiable modifiable, List<Package> packages) {
    
  }

}
```

The *refactor* method has only two parameters an **IModifiable** and a list of packages. The modifiable provides the context to access the actifsource resources. The package list contains the actifsource packages selected by the user when starting the refactoring. How the packages selection is interpreted is up to the implementer.

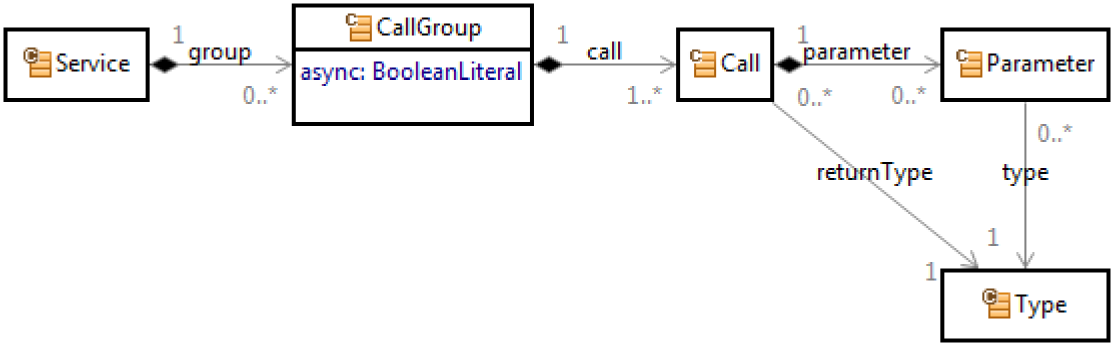
In general you need to use the two classes **Select** and **Update**. These are facades providing a convenient way to select and update resource information in a context. To get information about the available methods, open the class and a have a look at the javadoc comments on how to use them.

ⓘ The actifsource API works with statements and resources. A **Statement** is a triple connecting two resources (*subject*, *object*) through a property (*predicate*). The subject is the resource whose type (class) defines the predicate (property). The object is an instance of properties range. This is almost the same you see in the resource editor. In our example Patient is an instance of Service. Person refers the Calls throw the call-Relation defined in the Service class. For example you will get a statement Person (*subject*), call (*predicate*), Create (*object*) because Person refers to Create through the call relation. This is different from the diagram editor where subject and object are represented by their type.

ⓘ The old model looked like this



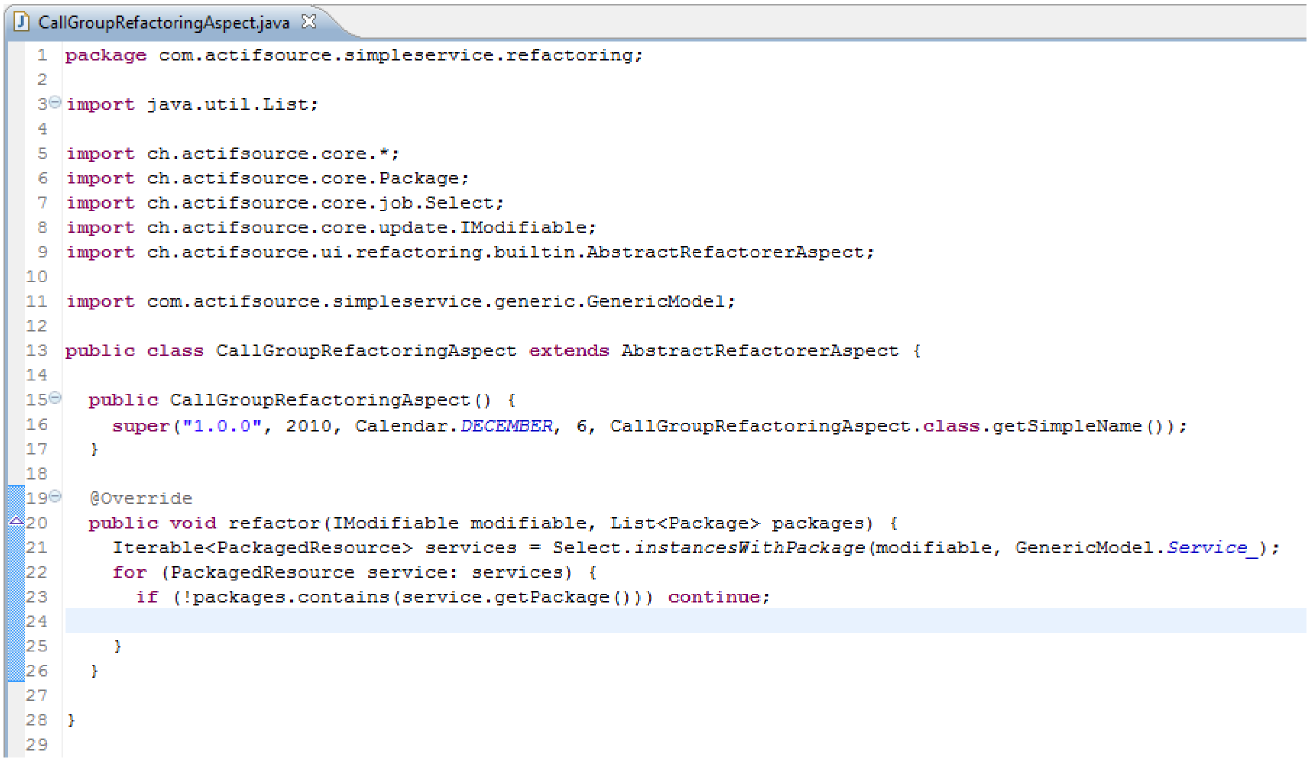
and the new model looks like this now



The only thing the refactoring has to do is adding a CallGroup into each Service and move the Calls into the group.

## Write a simple refactoring

↳ Select all Service instances in packages posted to the refactorer

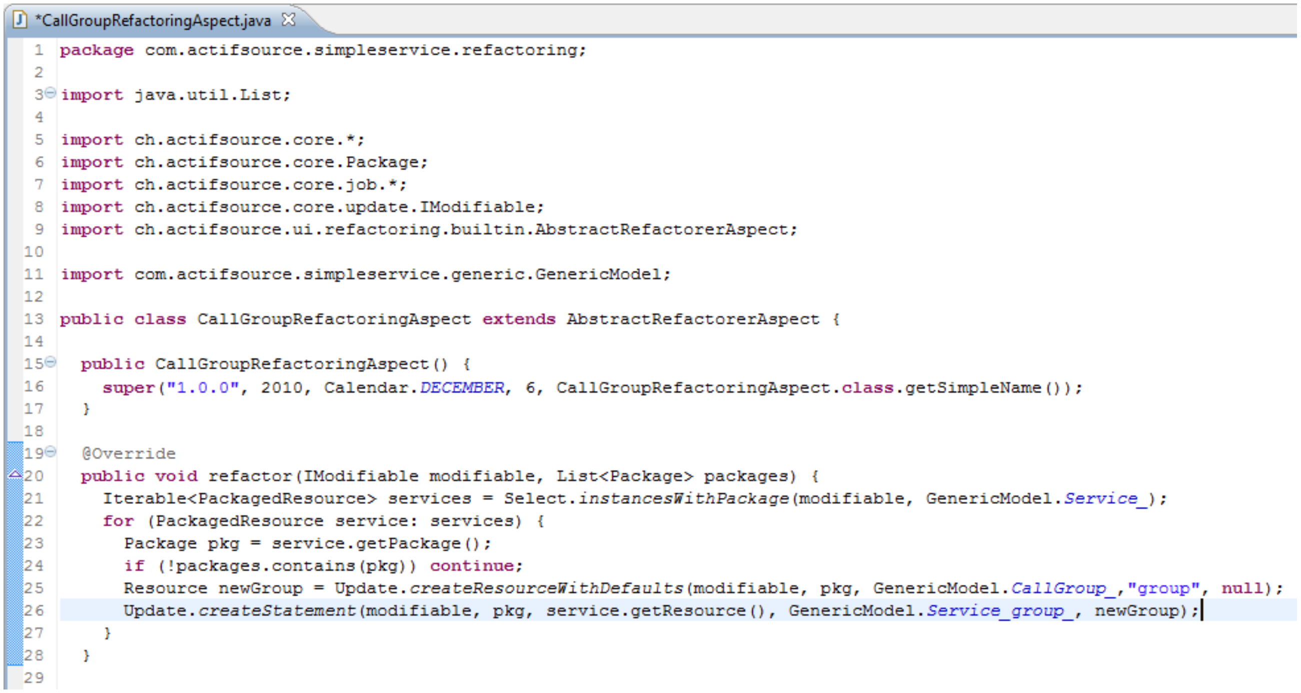

CallGroupRefactoringAspect.java

```java
package com.actifsource.simpleservice.refactoring;

import java.util.List;

import ch.actifsource.core.*;
import ch.actifsource.core.Package;
import ch.actifsource.core.job.Select;
import ch.actifsource.core.update.IModifiable;
import ch.actifsource.ui.refactoring.builtin.AbstractRefactorerAspect;

import com.actifsource.simpleservice.generic.GenericModel;

public class CallGroupRefactoringAspect extends AbstractRefactorerAspect {

  public CallGroupRefactoringAspect() {
    super("1.0.0", 2010, Calendar.DECEMBER, 6, CallGroupRefactoringAspect.class.getSimpleName());
  }

  @Override
  public void refactor(IModifiable modifiable, List<Package> packages) {
    Iterable<PackagedResource> services = Select.instancesWithPackage(modifiable, GenericModel.Service_);
    for (PackagedResource service: services) {
      if (!packages.contains(service.getPackage())) continue;

    }
  }

}
```

ⓘ The *instanceWithPackage* method takes an **ISelectable** and the **GUID** of a class. The result is an Iterable providing all instances reachable through the selectable. Since each **IModifiable** is also an **ISelectable**, you can use the **IModifiable** passed to the refactorer.

## Write a simple refactoring

Create a new CallGroup using the Update-Facade and add it to the Service-Instance

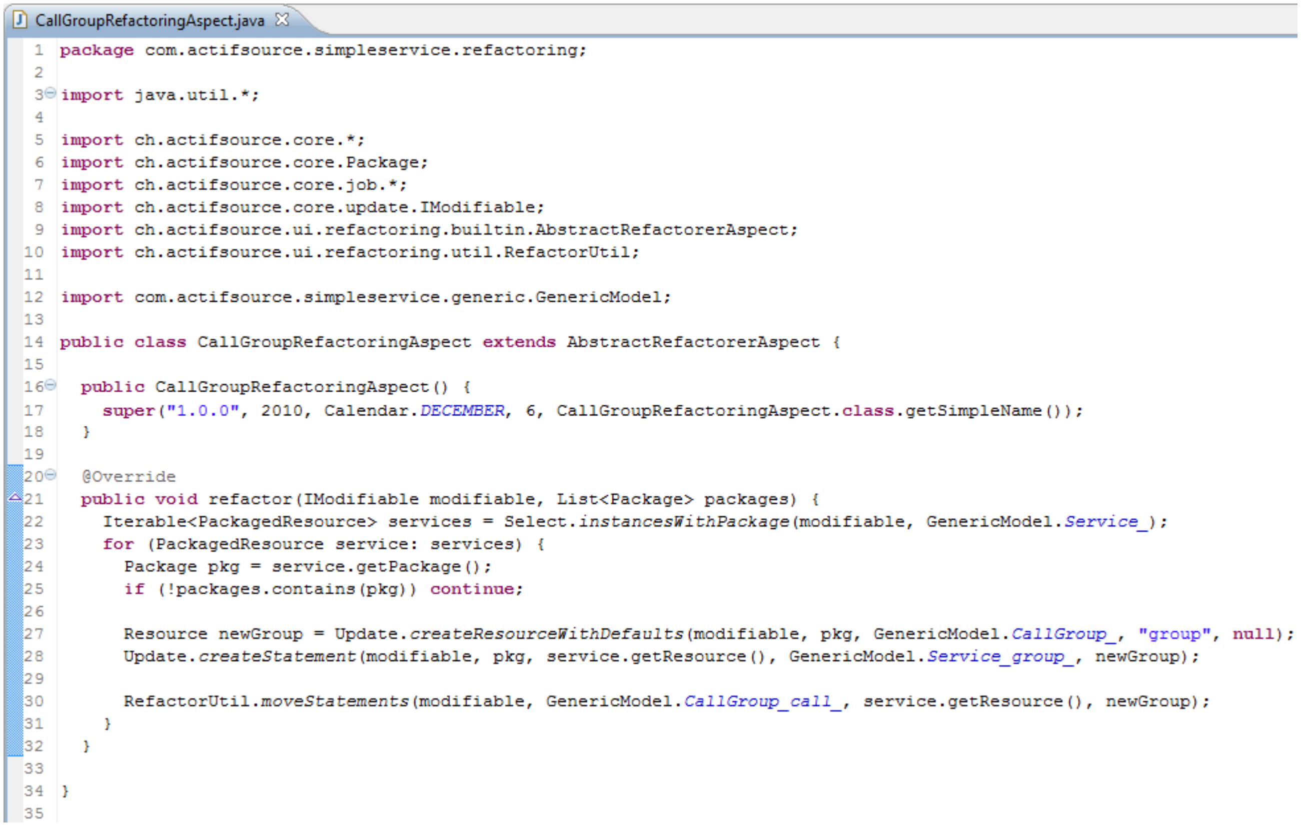

*CallGroupRefactoringAspect.java

```java
package com.actifsource.simpleservice.refactoring;

import java.util.List;

import ch.actifsource.core.*;
import ch.actifsource.core.Package;
import ch.actifsource.core.job.*;
import ch.actifsource.core.update.IModifiable;
import ch.actifsource.ui.refactoring.builtin.AbstractRefactorerAspect;

import com.actifsource.simpleservice.generic.GenericModel;

public class CallGroupRefactoringAspect extends AbstractRefactorerAspect {

  public CallGroupRefactoringAspect() {
    super("1.0.0", 2010, Calendar.DECEMBER, 6, CallGroupRefactoringAspect.class.getSimpleName());
  }

  @Override
  public void refactor(IModifiable modifiable, List<Package> packages) {
    Iterable<PackagedResource> services = Select.instancesWithPackage(modifiable, GenericModel.Service_);
    for (PackagedResource service: services) {
      Package pkg = service.getPackage();
      if (!packages.contains(pkg)) continue;
      Resource newGroup = Update.createResourceWithDefaults(modifiable, pkg, GenericModel.CallGroup_,"group", null);
      Update.createStatement(modifiable, pkg, service.getResource(), GenericModel.Service_group_, newGroup);
    }
  }

```

The Update-Facade always requires an **IModifiable** to work with. Using the *createResourceWithDefaults* method you can create a new instance of a type in the specific package. If you have a named resource, it is recommended to use the overload taking a name. The last parameter defines the default values for the attributes and relations when creating the resource and can be left out. For each property not found in the defaultValue map the default defined in the model will be used.

The *createStatement* method creates a statement. In this case the newly created group is assigned to the service using the group relation.

Move the Calls to the group



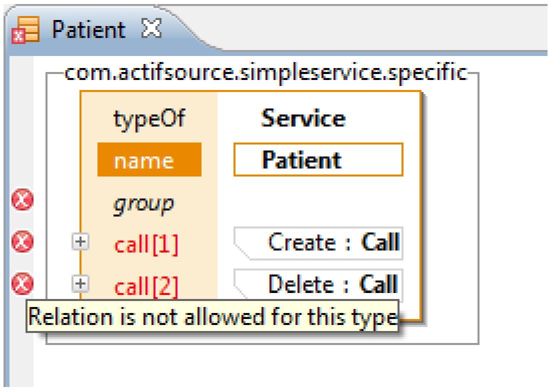
```java
package com.actifsource.simpleservice.refactoring;

import java.util.*;

import ch.actifsource.core.*;
import ch.actifsource.core.Package;
import ch.actifsource.core.job.*;
import ch.actifsource.core.update.IModifiable;
import ch.actifsource.ui.refactoring.builtin.AbstractRefactorerAspect;
import ch.actifsource.ui.refactoring.util.RefactorUtil;

import com.actifsource.simpleservice.generic.GenericModel;

public class CallGroupRefactoringAspect extends AbstractRefactorerAspect {

  public CallGroupRefactoringAspect() {
    super("1.0.0", 2010, Calendar.DECEMBER, 6, CallGroupRefactoringAspect.class.getSimpleName());
  }

  @Override
  public void refactor(IModifiable modifiable, List<Package> packages) {
    Iterable<PackagedResource> services = Select.instancesWithPackage(modifiable, GenericModel.Service_);
    for (PackagedResource service: services) {
      Package pkg = service.getPackage();
      if (!packages.contains(pkg)) continue;

      Resource newGroup = Update.createResourceWithDefaults(modifiable, pkg, GenericModel.CallGroup_, "group", null);
      Update.createStatement(modifiable, pkg, service.getResource(), GenericModel.Service_group_, newGroup);

      RefactorUtil.moveStatements(modifiable, GenericModel.CallGroup_call_, service.getResource(), newGroup);
    }
  }

}
```

ⓘ This time the method is located on the RefactorUtil. The reason for this is that the Update-Facade only provides simple modification methods and no selects. The *moveStatements* method uses methods from both the Select and Update method and is more complex. The *moveStatements* method take the **IModifiable**, a **Property**, the source and the target. It is simply often used when moving a **Property** from one class to another.

ⓘ By using the cut and paste in ResourceEditor, actifsource automatically detected that you have moved the **Property** call.

# Part IV: Execute a refactoring

↳ Open the Patient-Service to see that it is actually invalid.

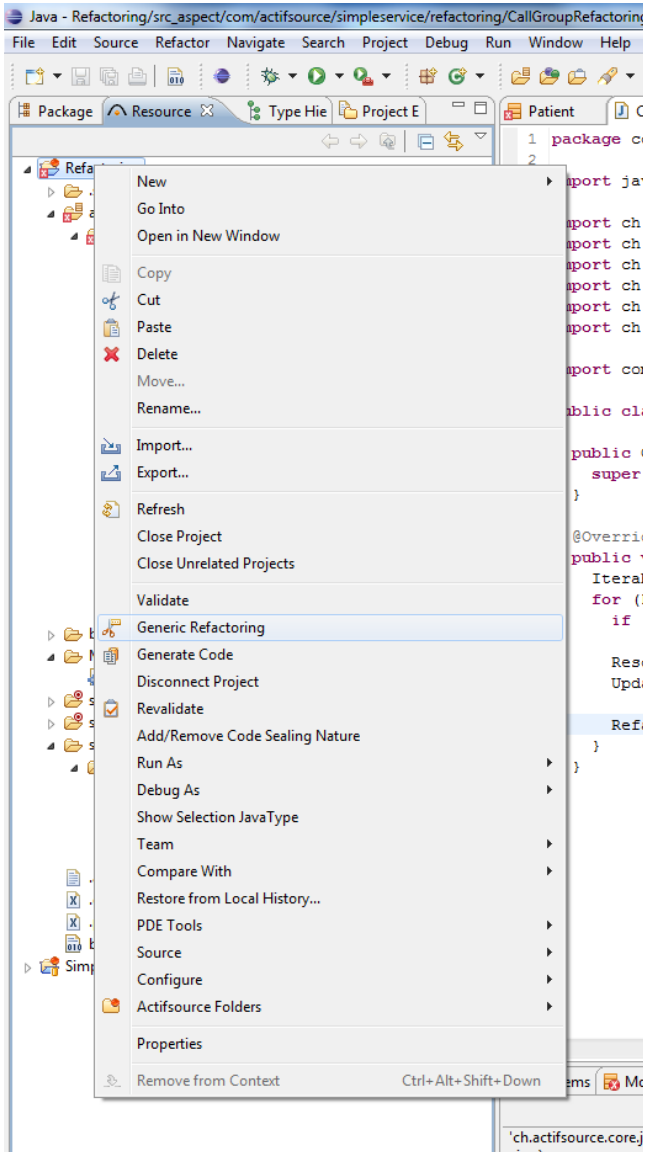


Relation is not allowed for this type

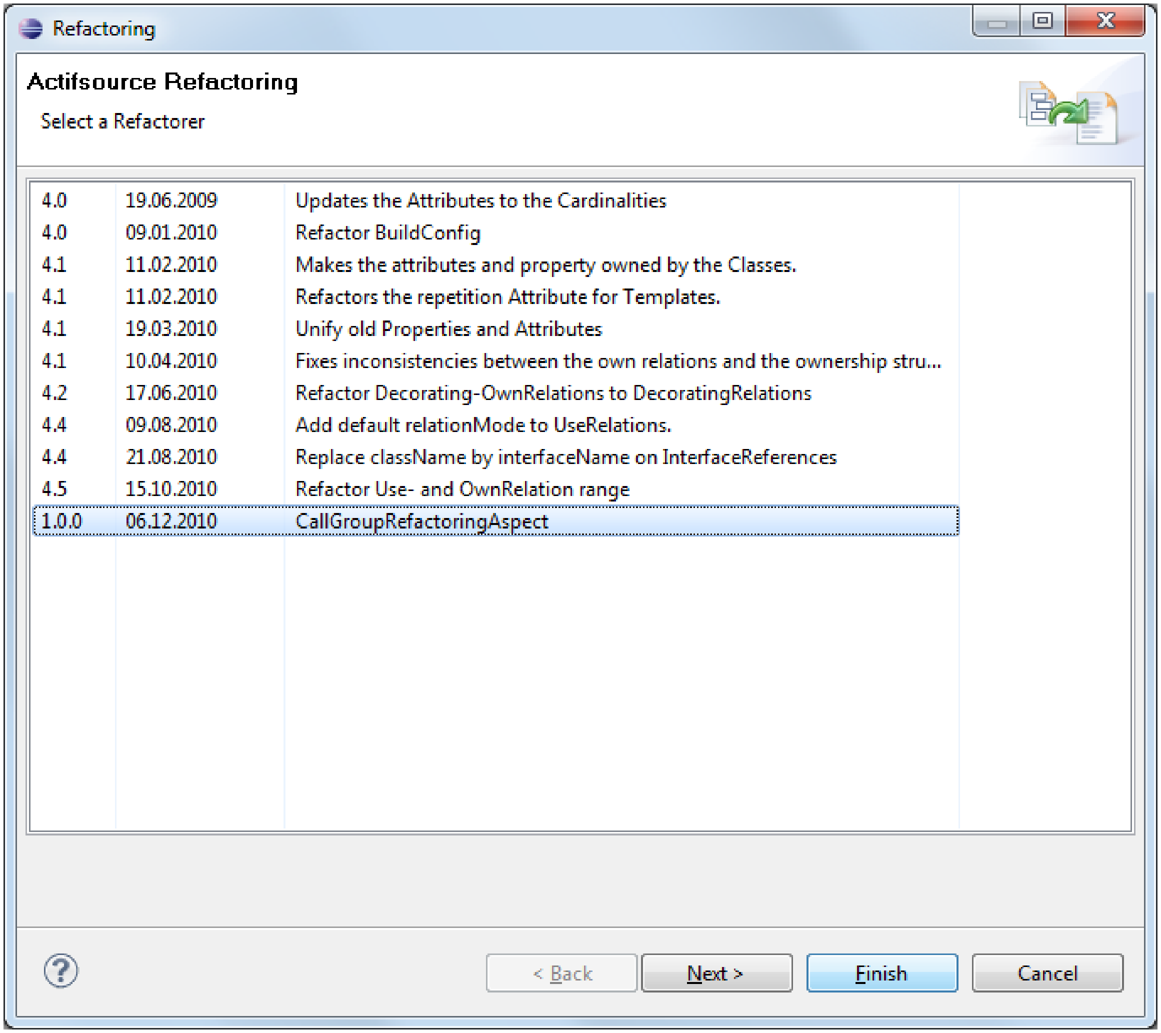
ⓘ As you can see the group relation that is already there and it would be possible to do change the model by hand.

## Execute a refactoring

- Open the context menu on the **project** or the **package** containing the Patient service



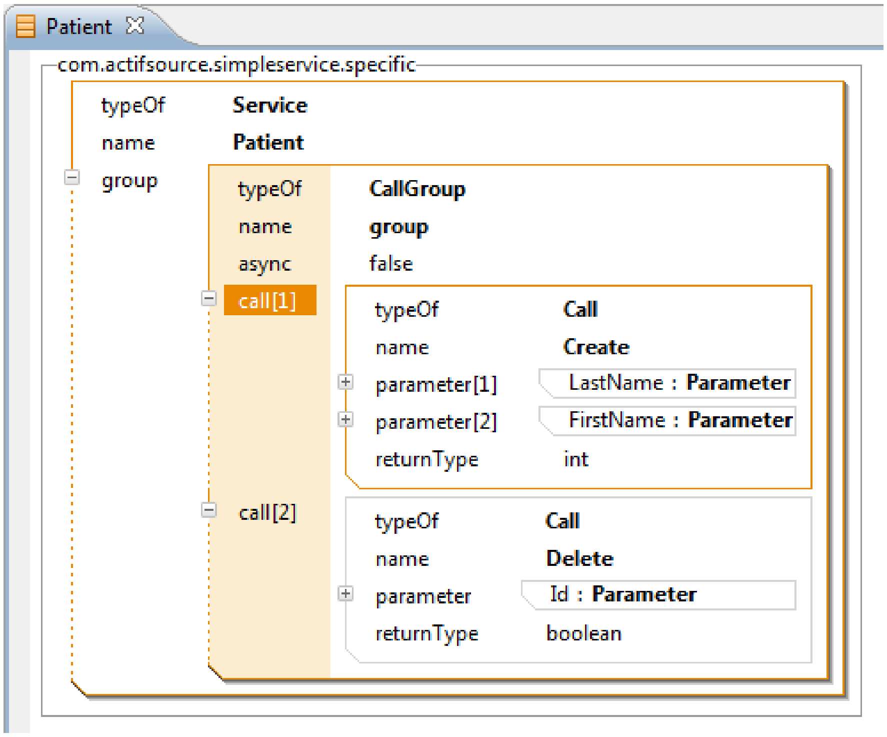
- Select the *CallGroupRefactoringAspect*.
- Press *Finish*.



| | | |
|---|---|---|
| 4.0 | 19.06.2009 | Updates the Attributes to the Cardinalities |
| 4.0 | 09.01.2010 | Refactor BuildConfig |
| 4.1 | 11.02.2010 | Makes the attributes and property owned by the Classes. |
| 4.1 | 11.02.2010 | Refactors the repetition Attribute for Templates. |
| 4.1 | 19.03.2010 | Unify old Properties and Attributes |
| 4.1 | 10.04.2010 | Fixes inconsistencies between the own relations and the ownership stru... |
| 4.2 | 17.06.2010 | Refactor Decorating-OwnRelations to DecoratingRelations |
| 4.4 | 09.08.2010 | Add default relationMode to UseRelations. |
| 4.4 | 21.08.2010 | Replace className by interfaceName on InterfaceReferences |
| 4.5 | 15.10.2010 | Refactor Use- and OwnRelation range |
| 1.0.0 | 06.12.2010 | CallGroupRefactoringAspect |

ⓘ As you can see, the values passed to the constructor are shown in the refactoring dialog. In actifsource we use the aspects to provide refactorings whenever we change the metamodel.

ⓘ The Patient service is now refactored

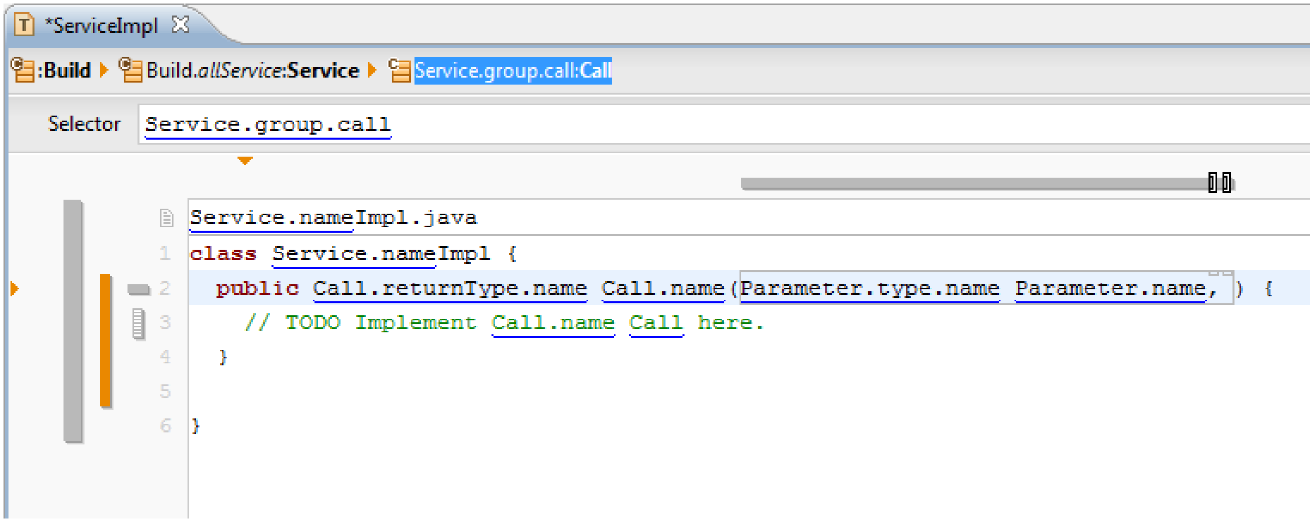

Patient

com.actifsource.simpleservice.specific

typeOf Service
name Patient
group
typeOf CallGroup
name group
async false
call[1]
typeOf Call
name Create
parameter[1] LastName : Parameter
parameter[2] FirstName : Parameter
returnType int
call[2]
typeOf Call
name Delete
parameter Id : Parameter
returnType boolean

# Part IV: Update the templates

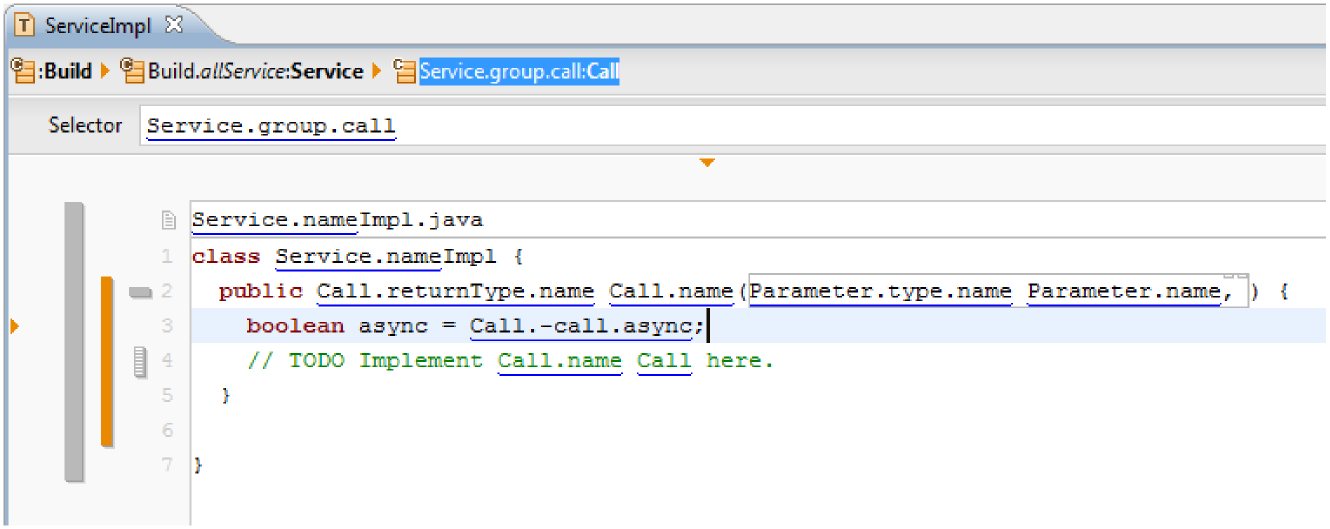
- The refactoring you have written only updates the instances. This is common practice, writing a refactoring for the service model and the templates makes no sense, since updating the classes is different for each step. The same applies for the templates.
- Open the ServiceImpl template
- Go to the Service.call selector by clicking on the first errormarker
- Change it to Service.group.call



```
Service.group.call
```

```
Service.nameImpl.java
class Service.nameImpl {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {
    // TODO Implement Call.name Call here.
  }

}
```

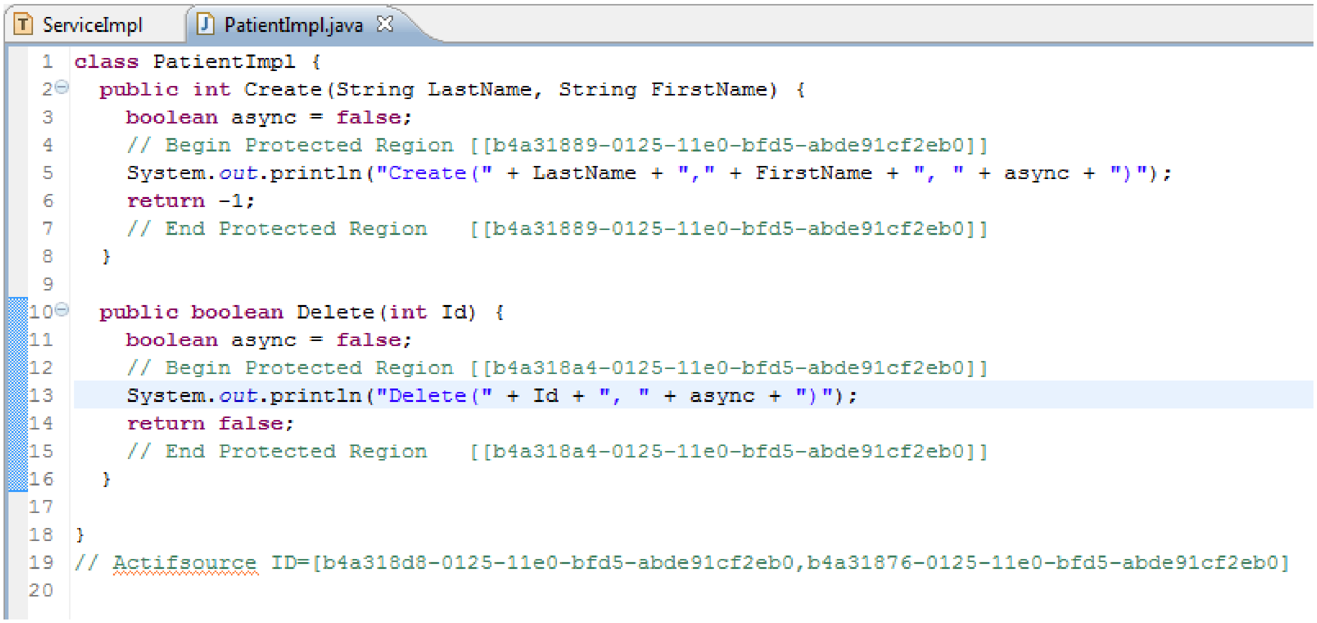
- Add a new line using the async attribute



```
Service.group.call
```

```
Service.nameImpl.java
class Service.nameImpl {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {
    boolean async = Call.-call.async;
    // TODO Implement Call.name Call here.
  }

}
```

Open the generate *PatientImpl.java* and update the protected regions.



```
class PatientImpl {
  public int Create(String LastName, String FirstName) {
    boolean async = false;
    // Begin Protected Region [[b4a31889-0125-11e0-bfd5-abde91cf2eb0]]
    System.out.println("Create(" + LastName + "," + FirstName + ", " + async + ")");
    return -1;
    // End Protected Region   [[b4a31889-0125-11e0-bfd5-abde91cf2eb0]]
  }

  public boolean Delete(int Id) {
    boolean async = false;
    // Begin Protected Region [[b4a318a4-0125-11e0-bfd5-abde91cf2eb0]]
    System.out.println("Delete(" + Id + ", " + async + ")");
    return false;
    // End Protected Region   [[b4a318a4-0125-11e0-bfd5-abde91cf2eb0]]
  }

}
// Actifsource ID=[b4a318d8-0125-11e0-bfd5-abde91cf2eb0,b4a31876-0125-11e0-bfd5-abde91cf2eb0]
```

You may play around a little bit with the model by adding an additional CallGroup to the Patient-service with the async attribute set to **true**.