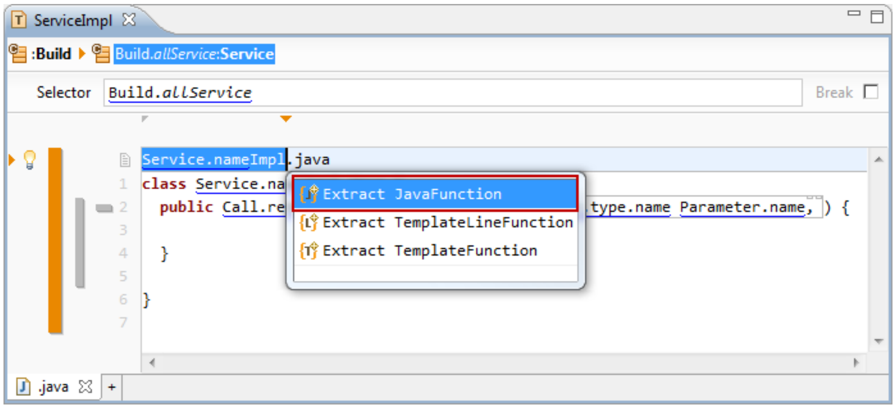
Tutorial

Complex Service

Tutorial	Actifsource Tutorial – Complex Service
Required Time	• 60 Minutes
Prerequisites	• Actifsource Tutorial – Installing Actifsource • Actifsource Tutorial – Simple Service
Goal	• Use Java Functions to reuse text fragments in your templates and capture complex expressions to keep your templates clean and easy to read • Use Function Spaces to keep Java Functions organized
Topics covered	• Extracting Java Functions from template code • Editing Java Functions • Advanced Template Editor Context Operations • Functions Spaces and Template Functions • Built-in Java Functions • Place generated code in specific folders • Copy with Context
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 www.actifsource.com
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.
Compatibility	Created with **actifsource** Version 5.8.5

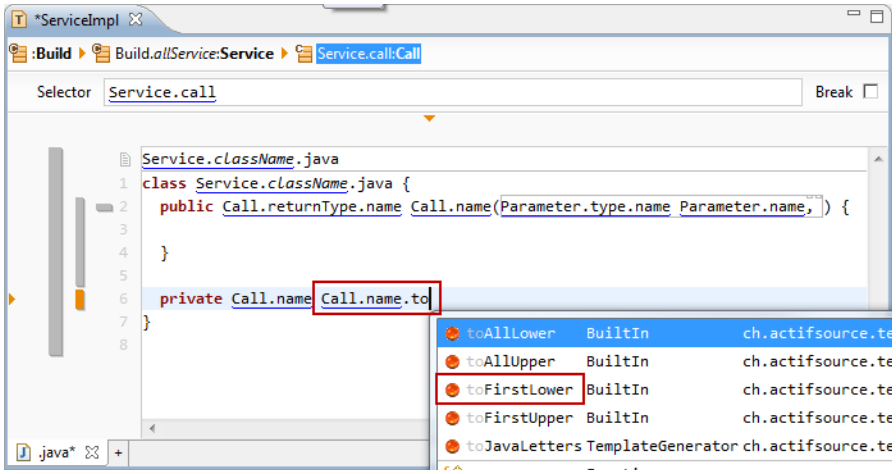
Overview

- Prepare a new actifsource Project as seen in the Actifsource Tutorial – Simple Service
- Learn how to extract Java Functions from template code to cope with complex situations



```
Build.allService

Service.nameImpl.java
class Service.na
  public Call.re                                   type.name Parameter.name, ) {

  }

}
```

- Edit Java Functions
- Learn about advanced Context Operations in the Template Editor
- Learn about Function Spaces and how to place functions
- Use built-in functions



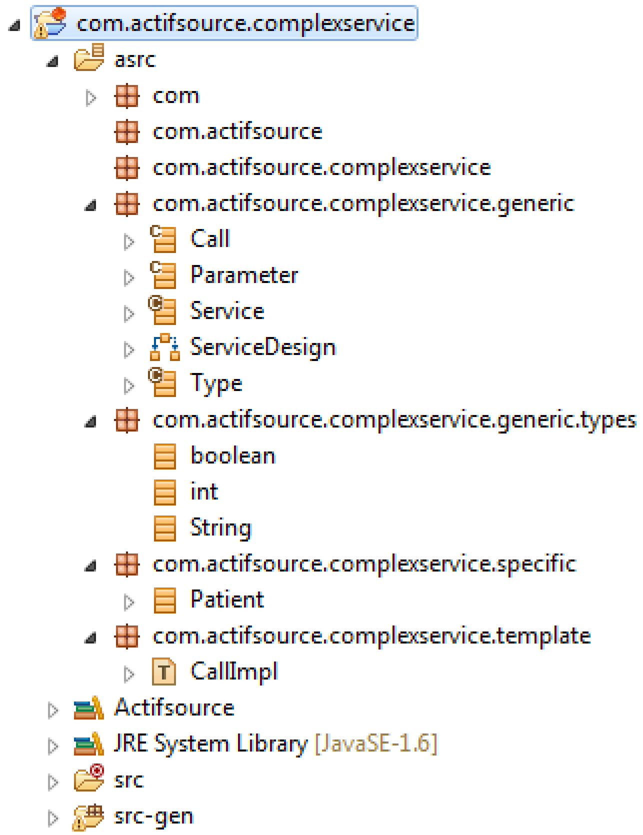
```
Service.call

Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  private Call.name Call.name.to
}
```

```
toAllLower     BuiltIn           ch.actifsource.te
toAllUpper     BuiltIn           ch.actifsource.te
toFirstLower   BuiltIn           ch.actifsource.te
toFirstUpper   BuiltIn           ch.actifsource.te
toJavaLetters  TemplateGenerator ch.actifsource.te
```

- Generate code for specific folders
- Copy template code with its Context

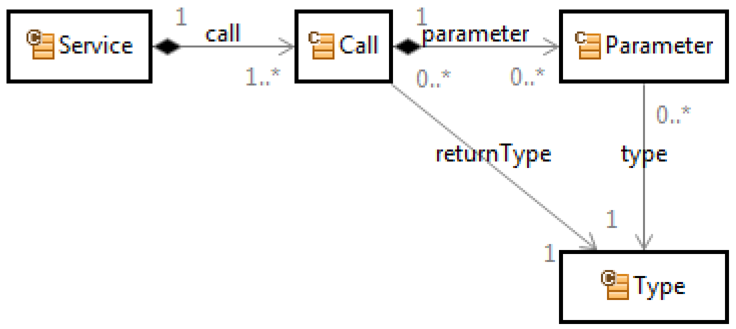
Part I: Preparation

- Prepare a new **actifsource Project** as seen in the *Actifsource Tutorial – Simple Service*
  - Setup the Target Folder *src*
  - Create a **Generic Domain Model**
  - Create a **Specific Domain Model**
  - Create a **Code Template**
- Use the following package structure



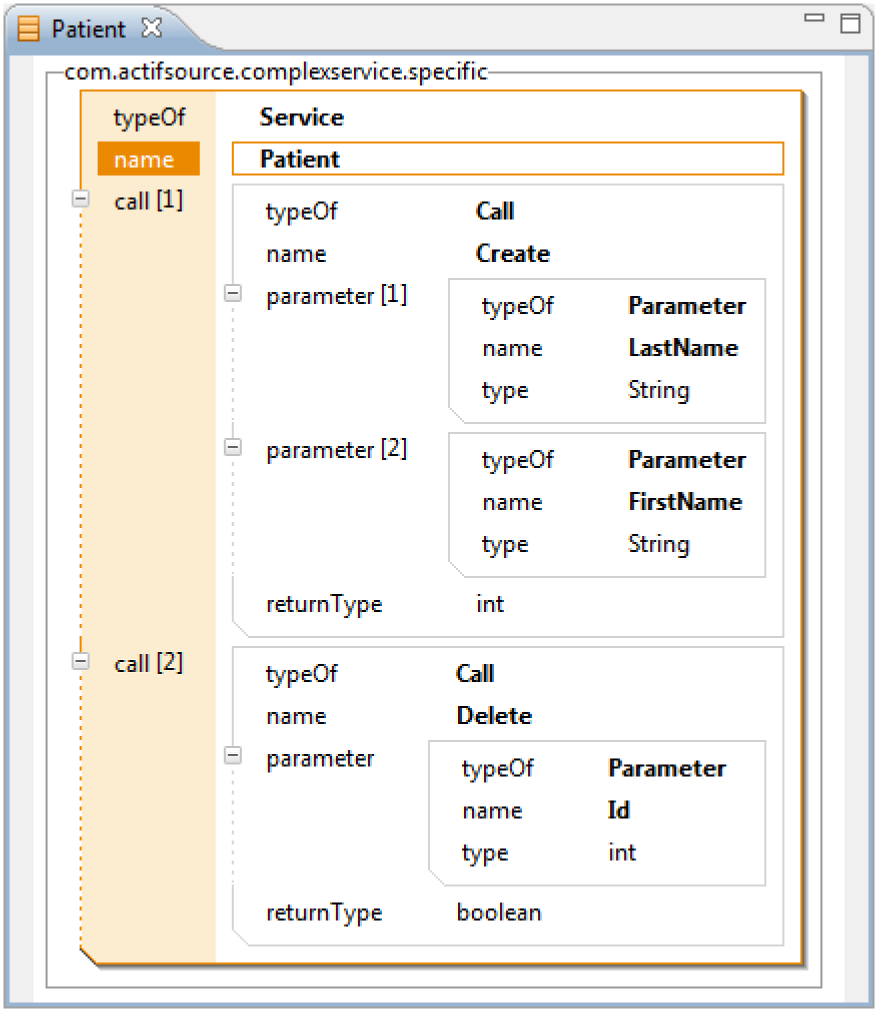
```
com.actifsource.complexservice
  asrc
    com
    com.actifsource
    com.actifsource.complexservice
    com.actifsource.complexservice.generic
      Call
      Parameter
      Service
      ServiceDesign
      Type
    com.actifsource.complexservice.generic.types
      boolean
      int
      String
    com.actifsource.complexservice.specific
      Patient
    com.actifsource.complexservice.template
      CallImpl
  Actifsource
  JRE System Library [JavaSE-1.6]
  src
  src-gen
```

Create a Generic Domain Model



- Create a **Generic Domain Model** in the **DiagramEditor** named *ServiceDesign* in the **Package** *generic*
- The Design shall contain the following **Domain Classes**
  - Service, Call, Parameter, Type
- Insert a **OwnRelations** between
  - Service and Call
  - Call and Parameter
- Insert a **UseRelations** between
  - Call and Type
  - Parameter and Type
- Adjust the **Cardinalities** as shown above

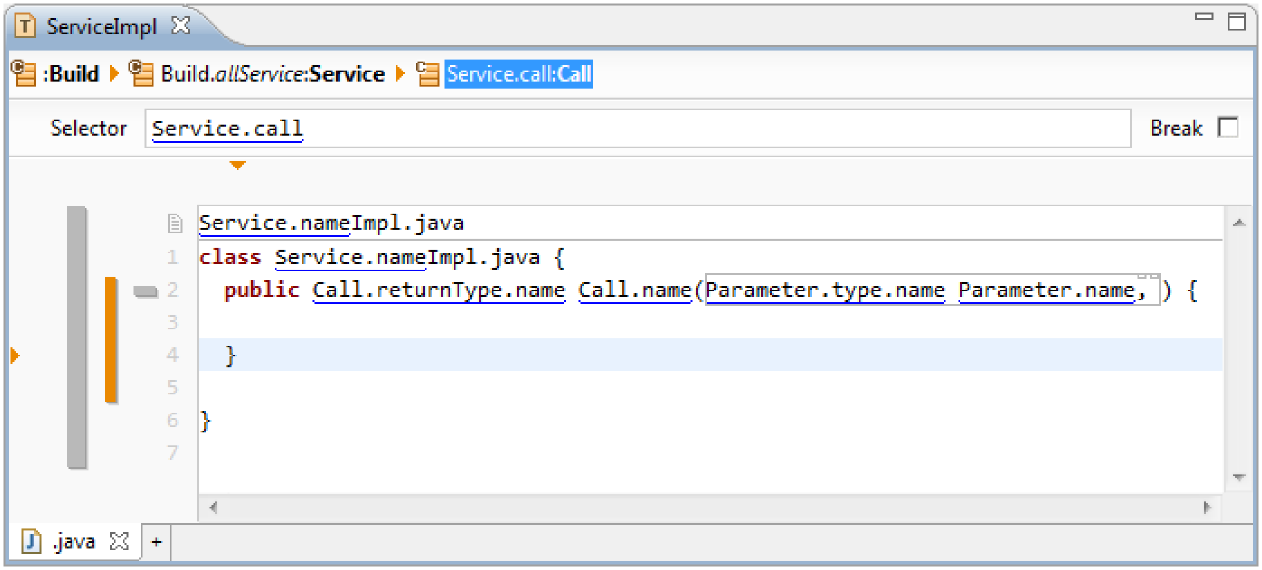
Create a Specific Domain Model



```
com.actifsource.complexservice.specific
typeOf      Service
name        Patient
call [1]    typeOf          Call
            name            Create
            parameter [1]   typeOf      Parameter
                            name        LastName
                            type        String
            parameter [2]   typeOf      Parameter
                            name        FirstName
                            type        String
            returnType      int
call [2]    typeOf          Call
            name            Delete
            parameter       typeOf      Parameter
                            name        Id
                            type        int
            returnType      boolean
```

- Create a Service named Patient in the **Package** *specific*
- Add the Calls Create and Delete
- Add the Parameter LastName, FirstName and Id as shown above
- Add the returnTypes as shown above

Create a Code Template

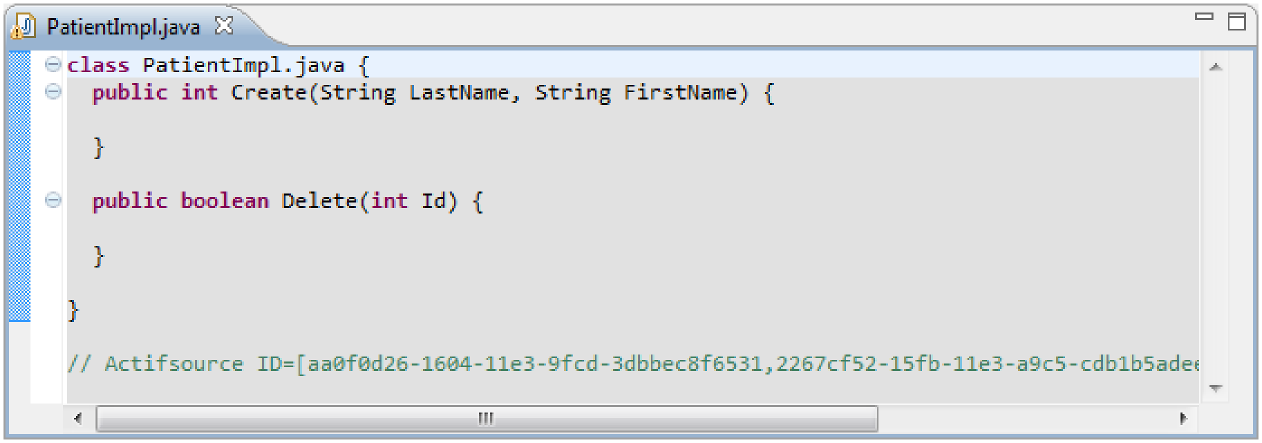


Selector: Service.call

```
Service.nameImpl.java
class Service.nameImpl.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

}
```

- Create a **Code Template** named *ServiceImpl* in the **Package** *template*
- Write code as shown above
- ⓘ The function shall be placed in the **Context** Call; **Selector** is Service.call
- ⓘ The function parameters shall be placed in the **Context** Parameter; **Selector** is Call.parameter
- Save the **Code Template**

Generate Code



PatientImpl.java

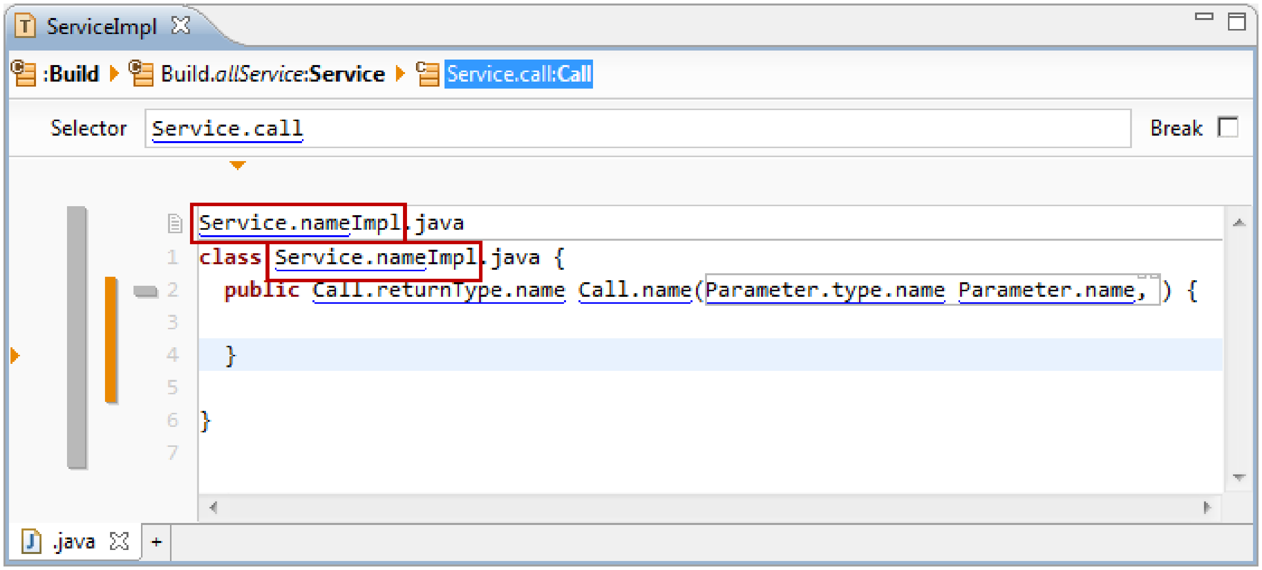
```
class PatientImpl.java {
  public int Create(String LastName, String FirstName) {

  }

  public boolean Delete(int Id) {

  }

}

// Actifsource ID=[aa0f0d26-1604-11e3-9fcd-3dbbec8f6531,2267cf52-15fb-11e3-a9c5-cdb1b5ade
```

ⓘ You'll find the generated code *PatientImpl.java* in the **Target Folder** *src*

Part II: Java Functions

- Use **Java Functions** to
    - extract recurring text fragments from your templates
    - capture complex expressions to keep your templates clean and easy to read
- Use **Java Classes** generated from your **Generic Domain Model** to write and maintain complex **Java Functions**

Extract Function



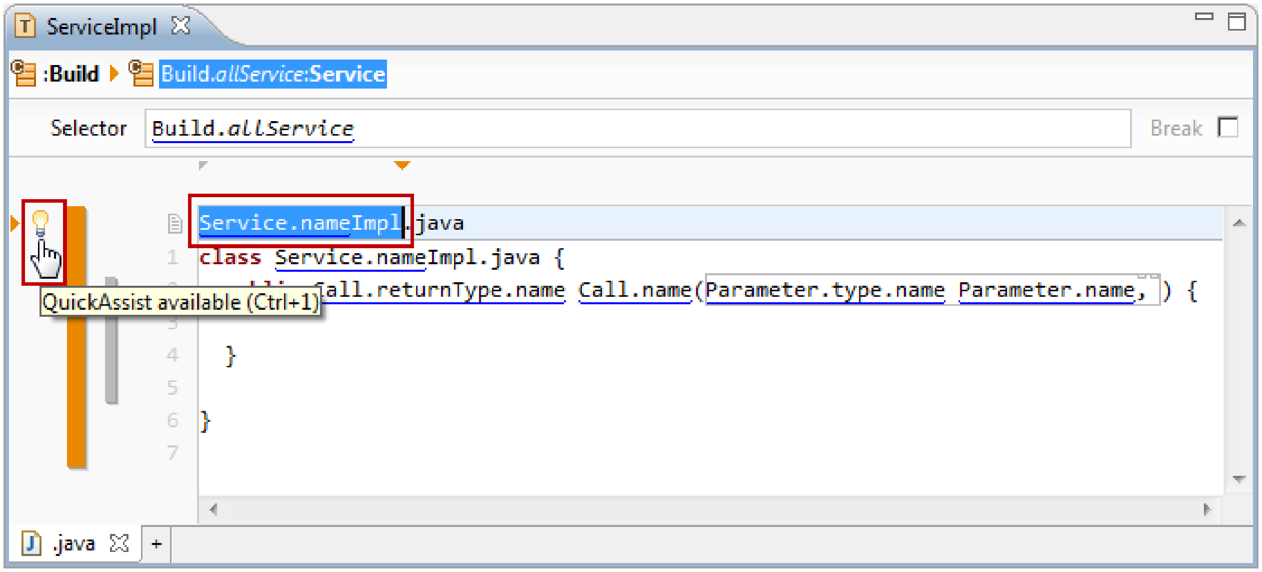
Selector: Service.call

```
Service.nameImpl.java
class Service.nameImpl.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

}
```

ⓘ Note that the term *Service.nameImpl* is used twice

ⓘ We should extract identical terms to honor the DRY principle (Don’t Repeat Yourself)

Extract Function

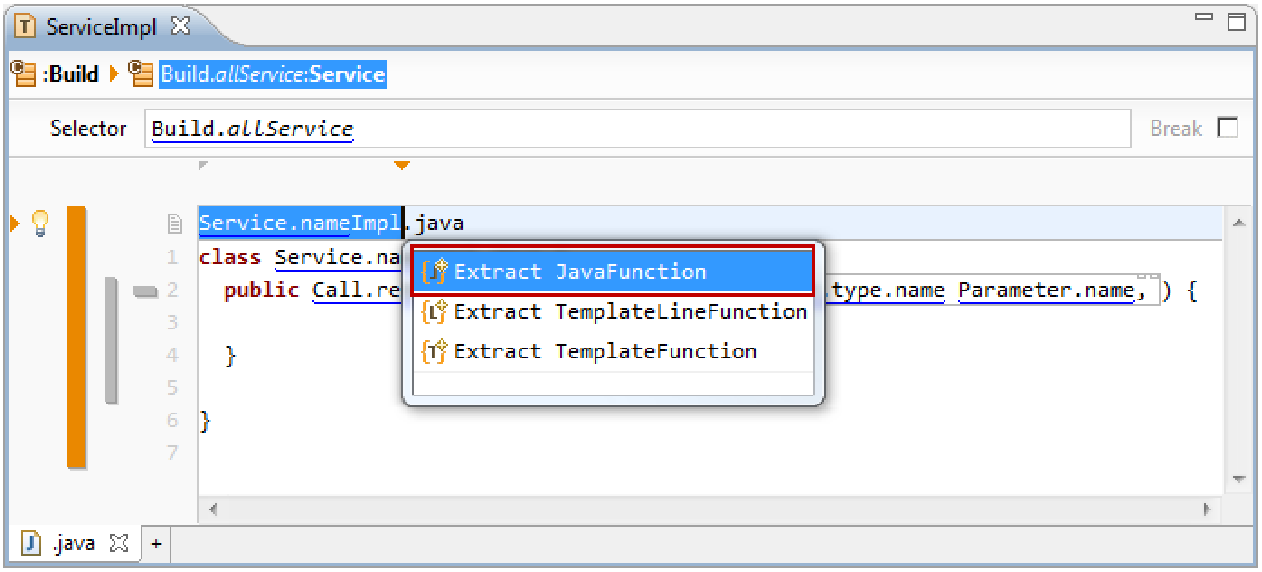


Selector: Build.allService

```
Service.nameImpl.java
class Service.nameImpl.java {
  publi Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

}
```

↳ In your template select the text you want to extract into a function

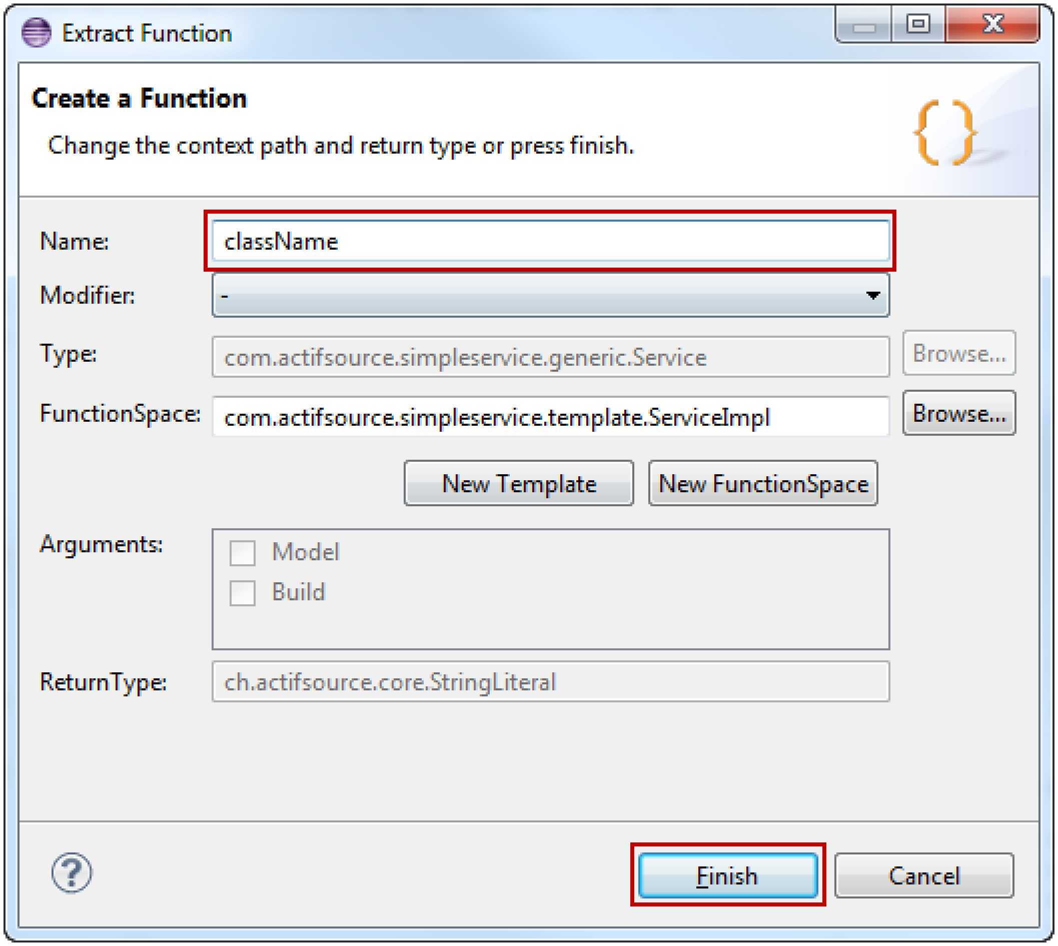
ⓘ The light bulb at the left hand indicates **Quick Assist** is available



Selector `Build.allService`

```
Service.nameImpl.java
class Service.na
  public Call.re                                    type.name Parameter.name, ) {

  }

}
```

- Activate **QuickAssist** by clicking the light bulb or by pressing Ctrl+1
- Click *Extract JavaFunction*


Extract Function

Create a Function

Change the context path and return type or press finish.

Name: className

Modifier: -

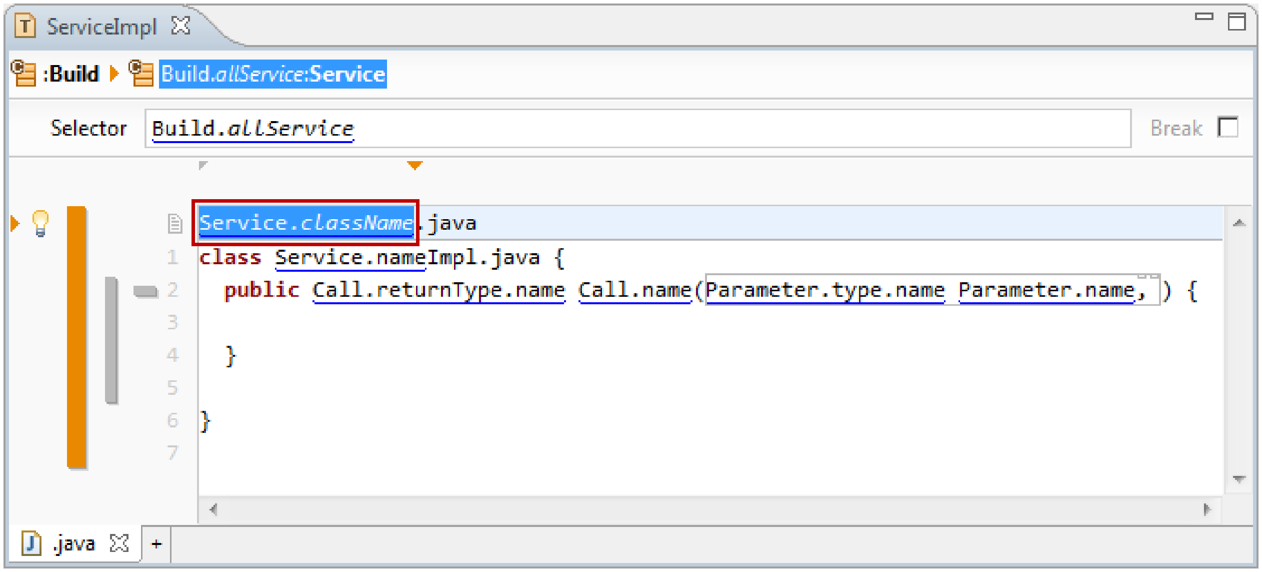
Type: com.actifsource.simpleservice.generic.Service

FunctionSpace: com.actifsource.simpleservice.template.ServiceImpl

Arguments: Model, Build

ReturnType: ch.actifsource.core.StringLiteral

- Name the function `className`
- Click *Finish*

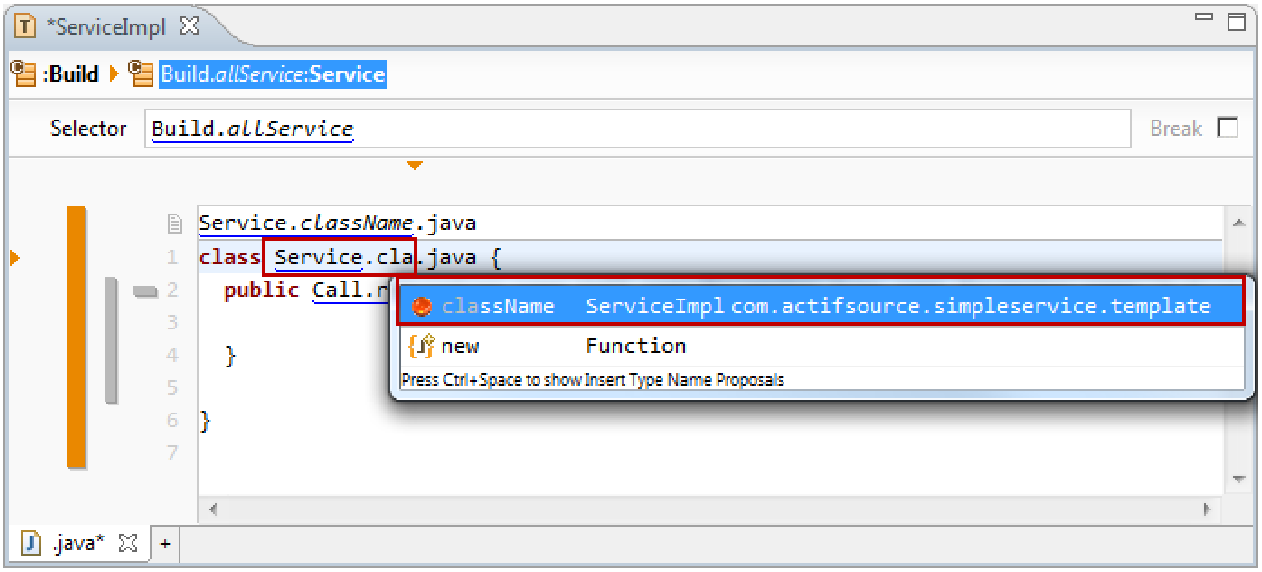


Build.allService

```
Service.className.java
class Service.nameImpl.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

}
```

- ⓘ The new function className returns the extracted fragment from your template
- ⓘ The static function className is added to the static Java class ServiceImpl.ServiceFunctions in class ServiceImpl; this class is automatically generated by actifsource
- ⓘ The term Service.nameImpl has been replaced by the function Service.*className*
- ⓘ **Java Functions** are shown in *italics* in the **actifsource Template Editor**

Using Functions

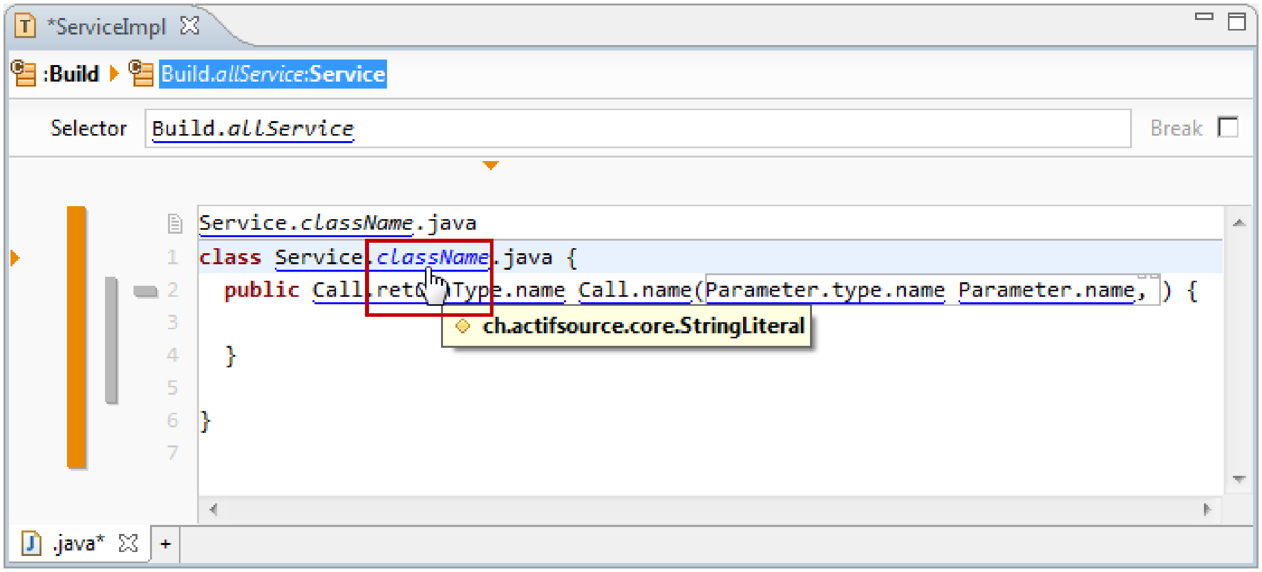


```
Service.className.java
class Service.cla.java {
  public Call.r
  
  }
  
}
```

```
className    ServiceImpl com.actifsource.simpleservice.template
new          Function
```

- Let’s replace the second occurrence of the term Service.nameImpl
- Use **Content Assist** (Ctrl+Space) on Service to insert the function *className* for your class name

Editing Functions



*ServiceImpl

:Build ▸ Build.allService:Service

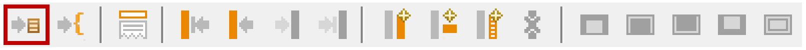
Selector Build.allService Break

```
Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

}
```

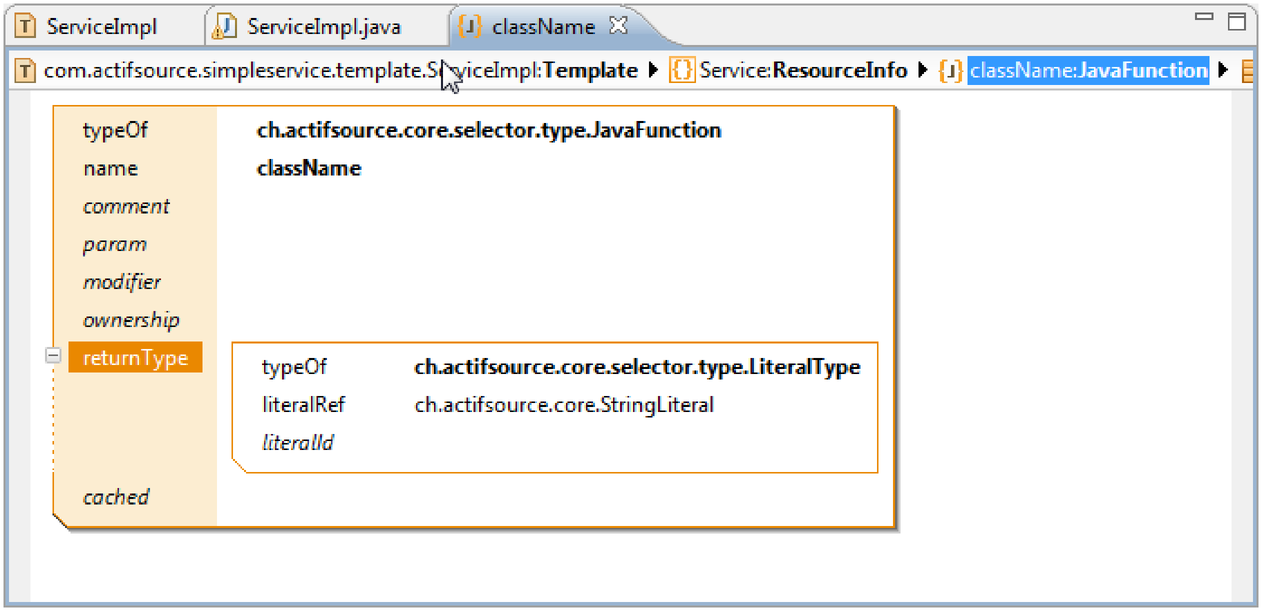
ch.actifsource.core.StringLiteral

.java*

- Open the underlying **Function Model** (Ctrl+Alt+Left-Click)
- Alternatively, you can use the **Tool** *Open Link in JavaEditor* from the **actifsource Template Editor** toolbar



Editing Functions



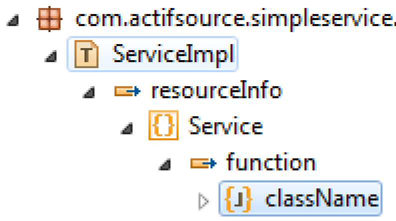
ServiceImpl | ServiceImpl.java | className

com.actifsource.simpleservice.template.ServiceImpl:Template ▸ Service:ResourceInfo ▸ className:JavaFunction ▸

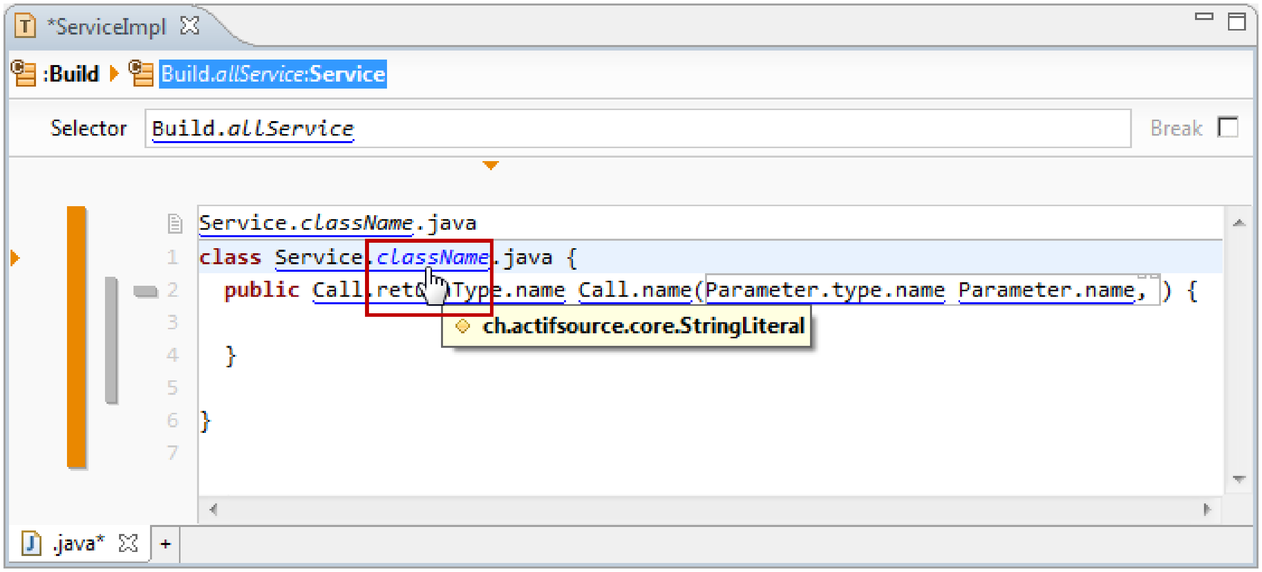
typeOf	ch.actifsource.core.selector.type.JavaFunction
name	className
comment	
param	
modifier	
ownership	
returnType	typeOf: ch.actifsource.core.selector.type.LiteralType; literalRef: ch.actifsource.core.StringLiteral; *literalId*
cached	

ⓘ Change function declaration here if needed

ⓘ Please notice that the Function declaration was placed in the template



```
com.actifsource.simpleservice.template
  ServiceImpl
    resourceInfo
      Service
        function
          className
```

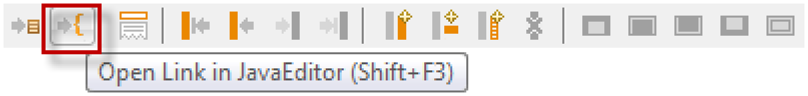


Selector: Build.allService

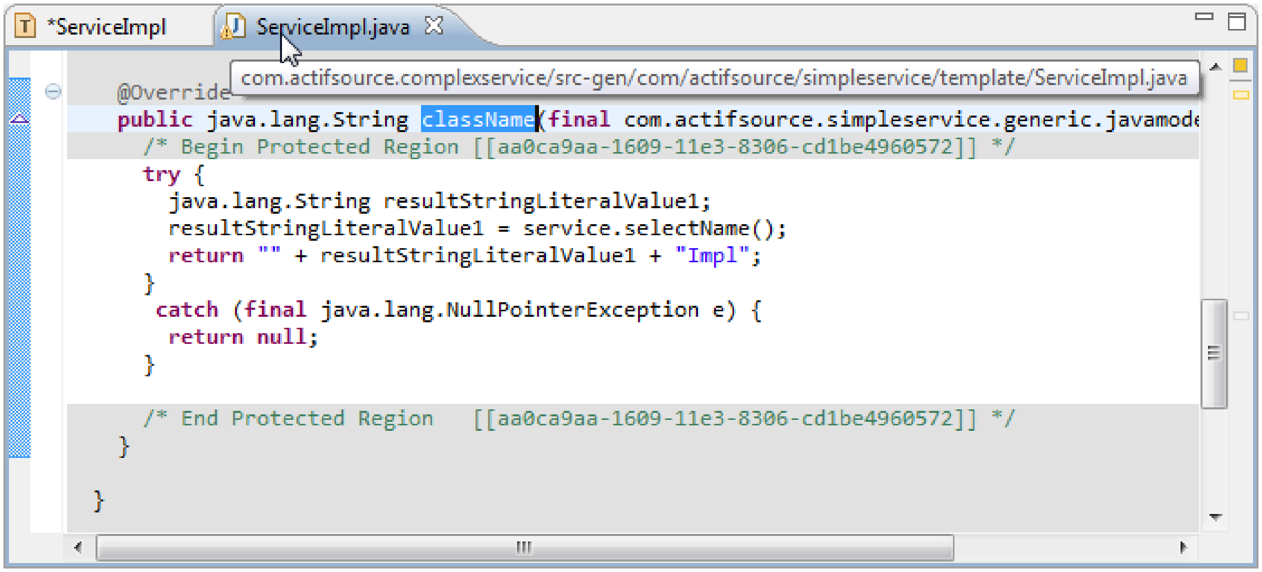
```
Service.className.java
class Service.className.java {
  public Call.ret(  )Type.name Call.name(Parameter.type.name Parameter.name, ) {

  }

}
```

Open the underlying **Java Function** (Ctrl+Left-Click)

Alternatively, you can use the **Tool** *Open Link in JavaEditor* from the **actifsource Template Editor** toolbar



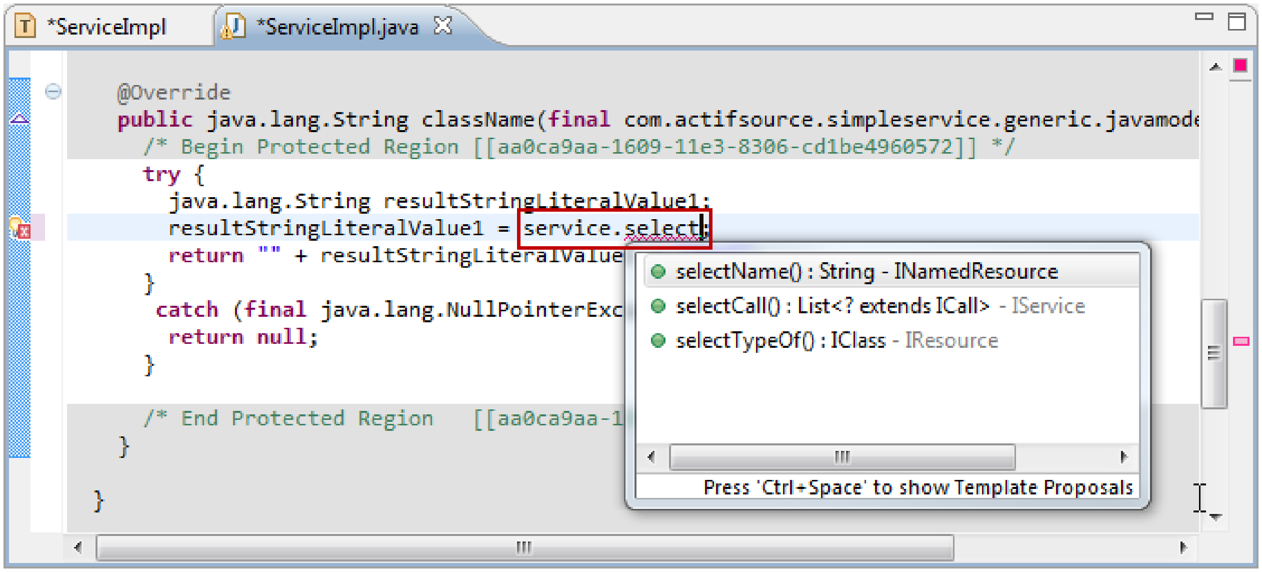
Editing Functions


*ServiceImpl
ServiceImpl.java
com.actifsource.complexservice/src-gen/com/actifsource/simpleservice/template/ServiceImpl.java

```
  @Override
  public java.lang.String className(final com.actifsource.simpleservice.generic.javamode
    /* Begin Protected Region [[aa0ca9aa-1609-11e3-8306-cd1be4960572]] */
    try {
      java.lang.String resultStringLiteralValue1;
      resultStringLiteralValue1 = service.selectName();
      return "" + resultStringLiteralValue1 + "Impl";
    }
     catch (final java.lang.NullPointerException e) {
      return null;
    }

    /* End Protected Region   [[aa0ca9aa-1609-11e3-8306-cd1be4960572]] */
  }

}
```

ⓘ The class ServiceImpl is opened in the **Java Editor** showing your function className

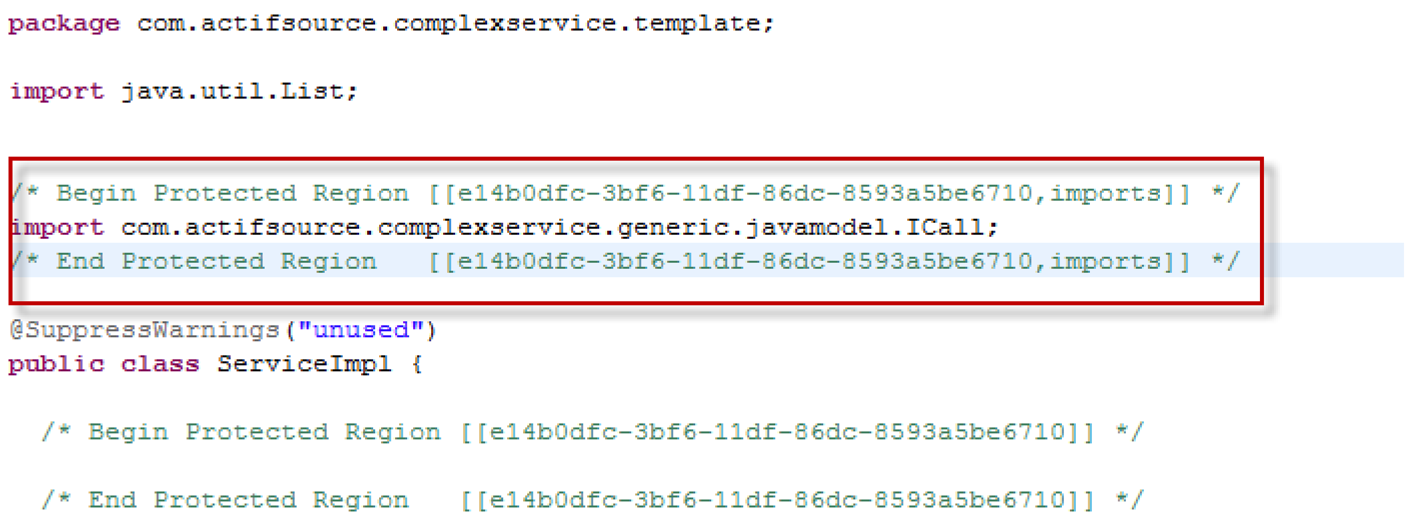
Editing Functions



```
@Override
public java.lang.String className(final com.actifsource.simpleservice.generic.javamode
  /* Begin Protected Region [[aa0ca9aa-1609-11e3-8306-cd1be4960572]] */
  try {
    java.lang.String resultStringLiteralValue1;
    resultStringLiteralValue1 = service.select;
    return "" + resultStringLiteralValue
  }
  catch (final java.lang.NullPointerExc
    return null;
  }

  /* End Protected Region   [[aa0ca9aa-1
}

}
```

```
selectName() : String - INamedResource
selectCall() : List<? extends ICall> - IService
selectTypeOf() : IClass - IResource
```

ⓘ Note that actifsource generates a *select* method for each property of the corresponding class in the **Generic Domain Model.** You may use these methods to traverse your **Generic Domain Model** using the respective *selectPROPERTY()* methods in your **Java Functions**



```
package com.actifsource.complexservice.template;

import java.util.List;

/* Begin Protected Region [[e14b0dfc-3bf6-11df-86dc-8593a5be6710,imports]] */
import com.actifsource.complexservice.generic.javamodel.ICall;
/* End Protected Region   [[e14b0dfc-3bf6-11df-86dc-8593a5be6710,imports]] */

@SuppressWarnings("unused")
public class ServiceImpl {

  /* Begin Protected Region [[e14b0dfc-3bf6-11df-86dc-8593a5be6710]] */

  /* End Protected Region   [[e14b0dfc-3bf6-11df-86dc-8593a5be6710]] */
```

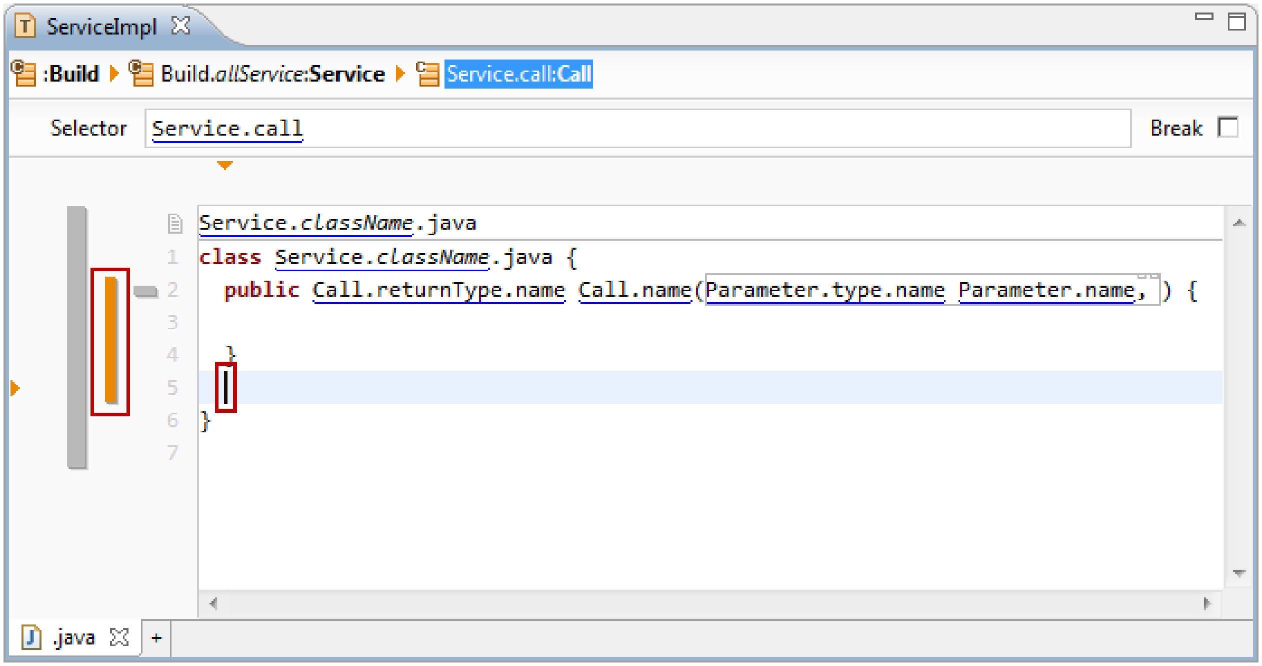
ⓘ Make sure to place additional imports within the corresponding **Protected Regions**

ⓘ Please note that all code outside **Protected Regions** will be overwritten if the respective source file is re-generated.

Part III: Function Spaces

- **Function Declarations** are managed as **actifsource Resources**
- All **Function Declarations** are placed in **Functions Spaces**
- **Templates** are **Functions Spaces** by default
- **Functions Spaces** can exist without **Templates**
- **Function Spaces** are **Resources** and can therefore be placed in **Packages**

Advanced Context Operations



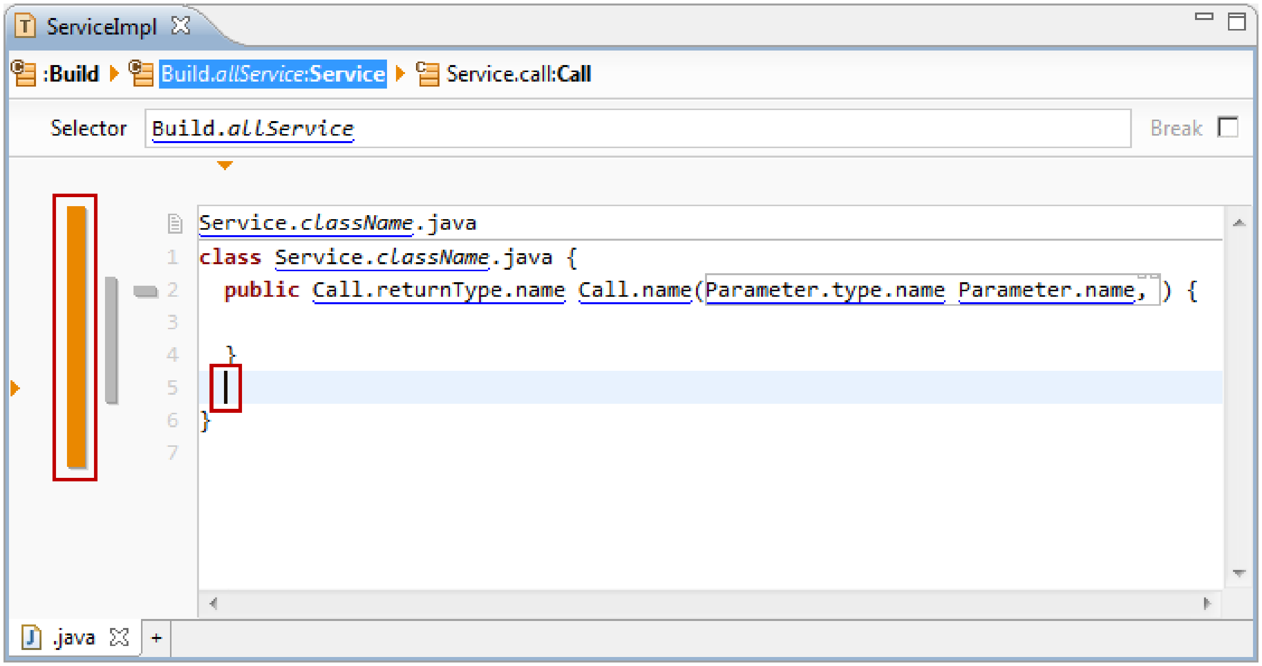
```
Service.call
```

```
Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

}
```

ⓘ Let’s add a new line after the Call **Context** in the Service **Context**

Place cursor on the last position of the Call **Context**

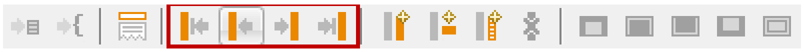
ⓘ Note that the corresponding **Context Bar** is highlighted

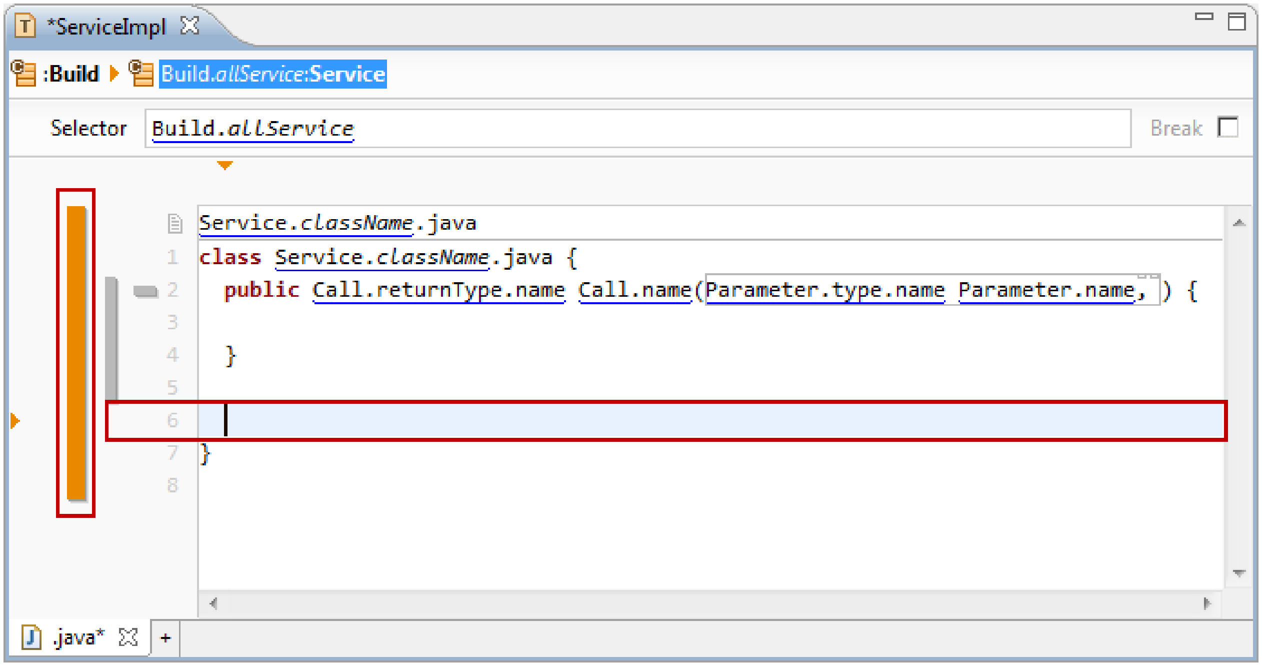


Selector: Build.allService

```
Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

}
```

- Press Cursor-Right
- ⓘ While the cursor stays at its position, the Service **Context** is now highlighted
- ⓘ Alternatively, you can use the **context navigation** from the **actifsource Template Editor** toolbar





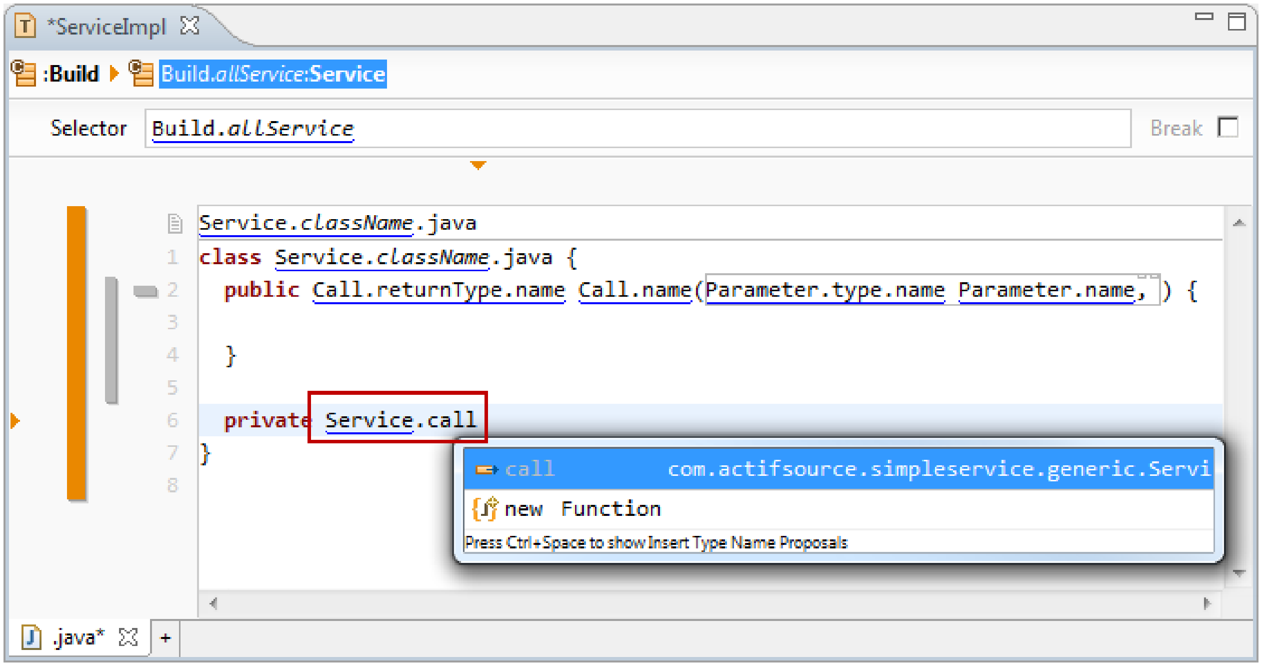
```
Build.allService
```

```
Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }


}
```

Press Enter

A new line has been added in the Parent **Context**

Add Context via Quick Assist



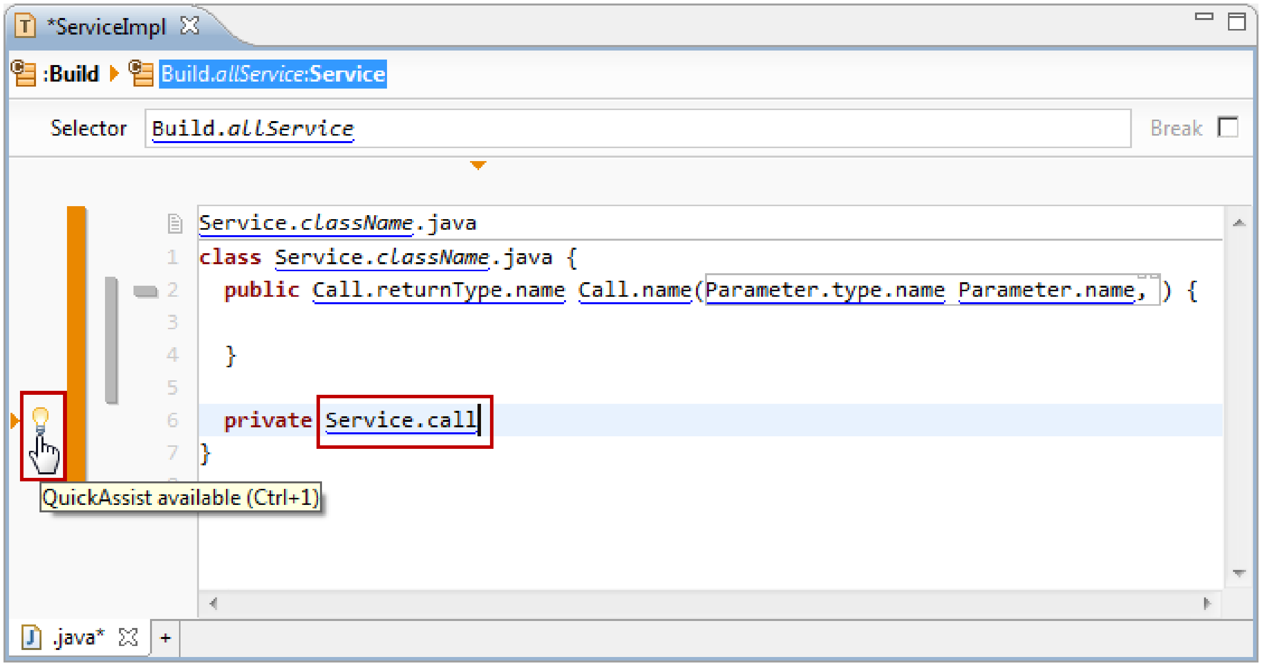
```
Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  private Service.call
}
```

```
call        com.actifsource.simpleservice.generic.Servi
new  Function
Press Ctrl+Space to show Insert Type Name Proposals
```

ⓘ Let’s look at a quick and easy way to insert a new **Context**

↳ Insert the **Variable** <u>Service.call</u> using **Content Assist** (Ctrl+Space)

Add Context via Quick Assist



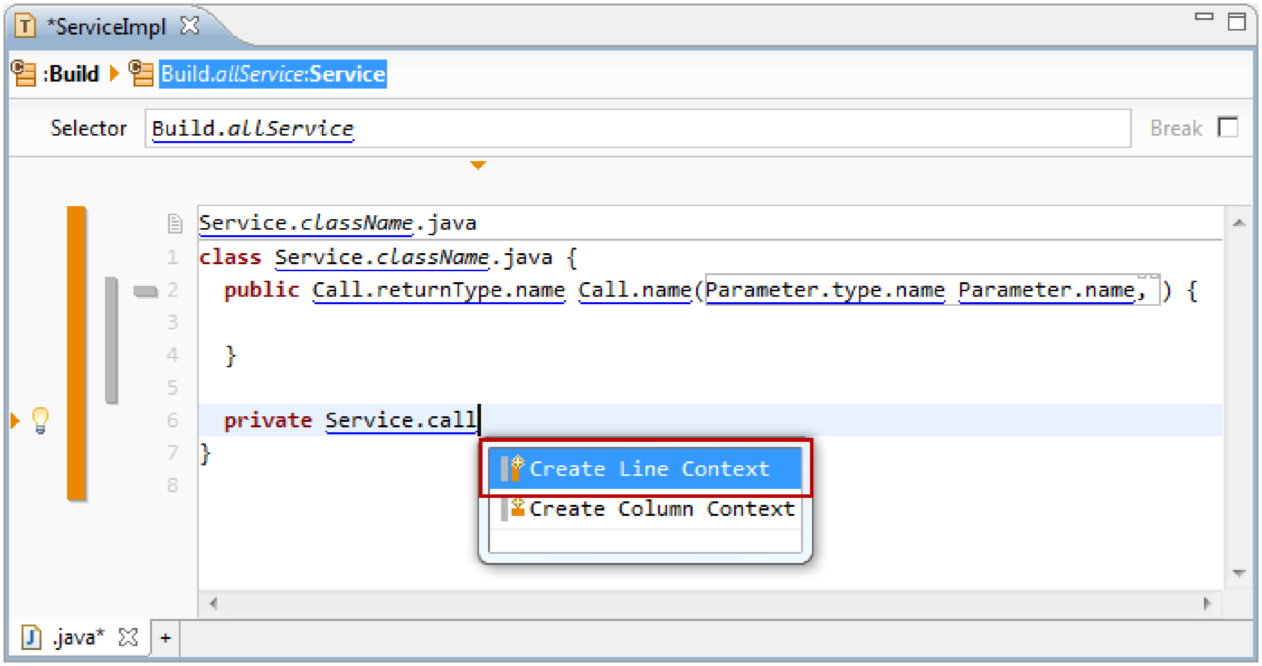
Build.allService

```
Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  private Service.call
}
```

ⓘ The light bulb at the left hand indicates **Quick Assist** is available

↳ Activate **QuickAssist** by clicking the light bulb or pressing Ctrl+1

Add Context via Quick Assist

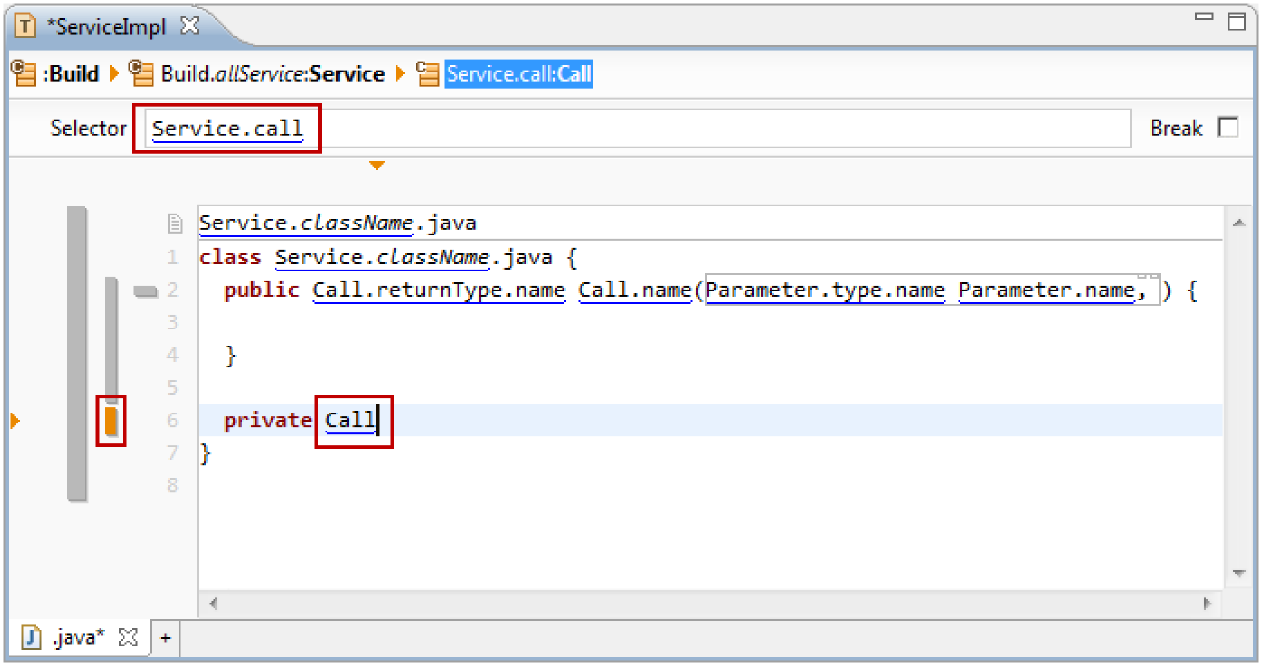


```
Build.allService
```

```
Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  private Service.call
}
```

↳ Click on *Create Line Context*

Add Context via Quick Assist



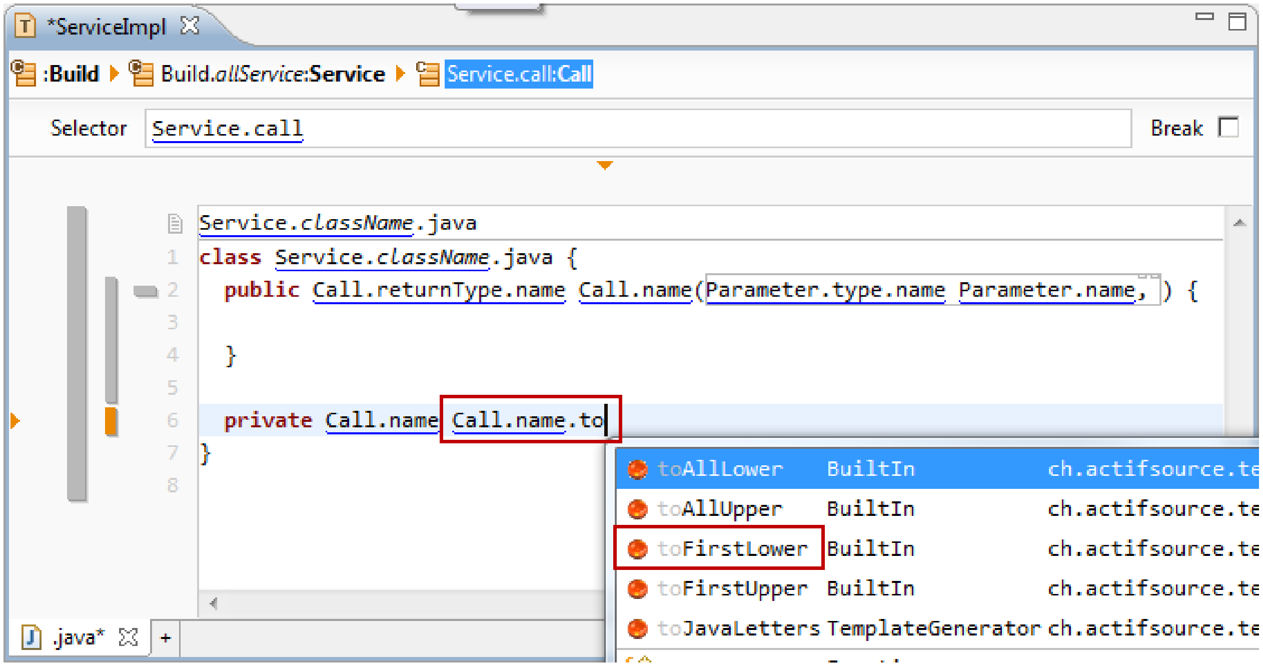
```
Selector Service.call
```

```
Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  private Call
}
```

ⓘ Note that a new Call **Context** (**Selector**: Service.call) has been added

ⓘ The Variable Service.call has been replaced by Call

Built-In Functions


*ServiceImpl

:Build ▸ Build.allService:Service ▸ Service.call:Call

Selector Service.call Break

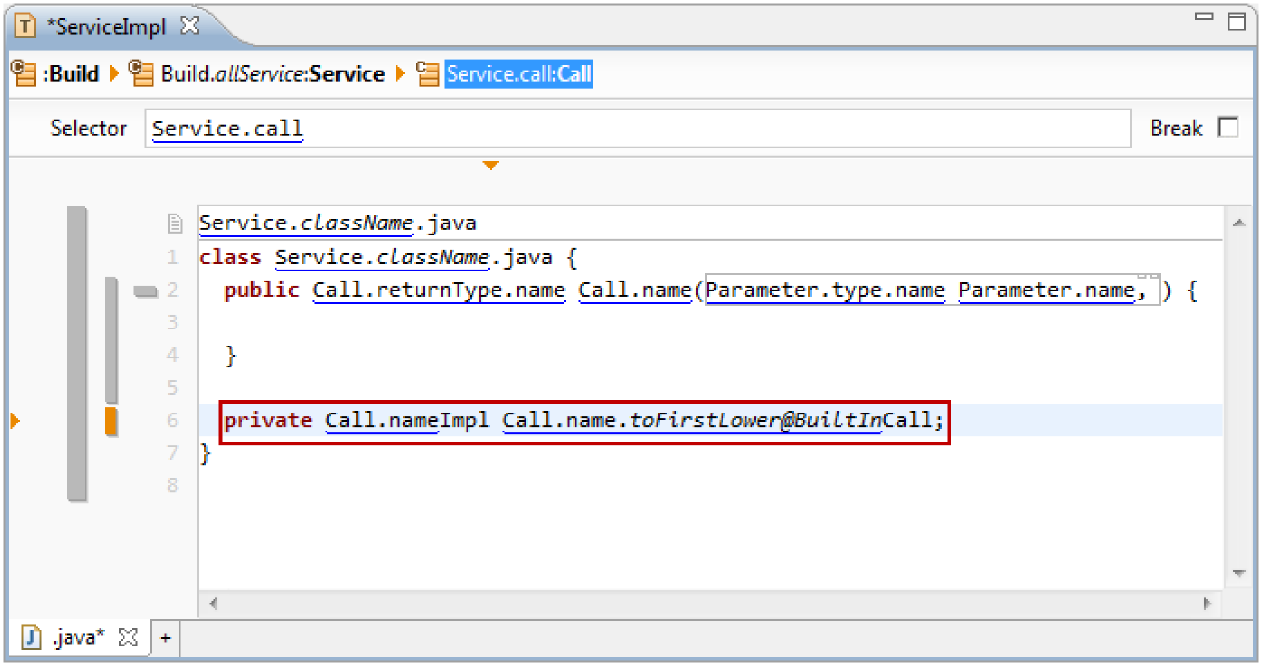
```
Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  private Call.name Call.name.to
}
```

```
toAllLower     BuiltIn           ch.actifsource.te
toAllUpper     BuiltIn           ch.actifsource.te
toFirstLower   BuiltIn           ch.actifsource.te
toFirstUpper   BuiltIn           ch.actifsource.te
toJavaLetters  TemplateGenerator ch.actifsource.te
```

.java*

- ⓘ Let’s use **Built-In Functions** on **Attributes**
- Press '.' (dot) and **Content Assist** (Ctrl+Space) after **name** to see all available **Built-In Functions**

Built-In Functions

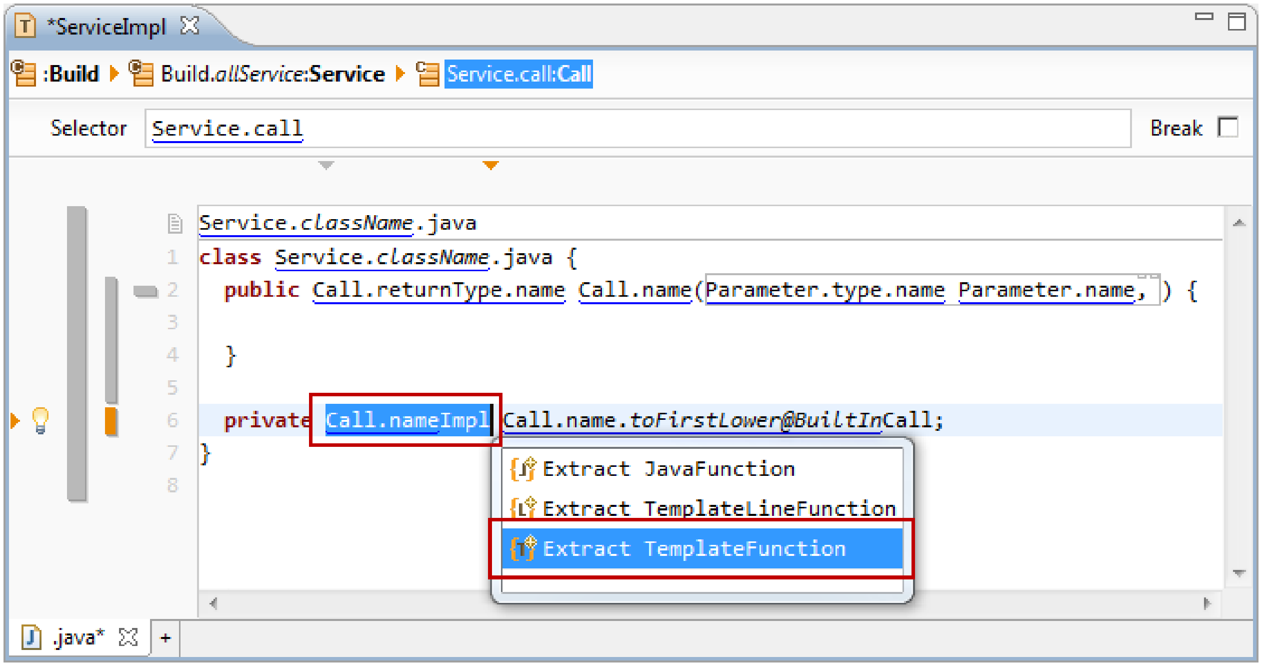


Selector: Service.call

```
Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  private Call.nameImpl Call.name.toFirstLower@BuiltInCall;
}
```

↳ Complete the member variable declaration as shown above

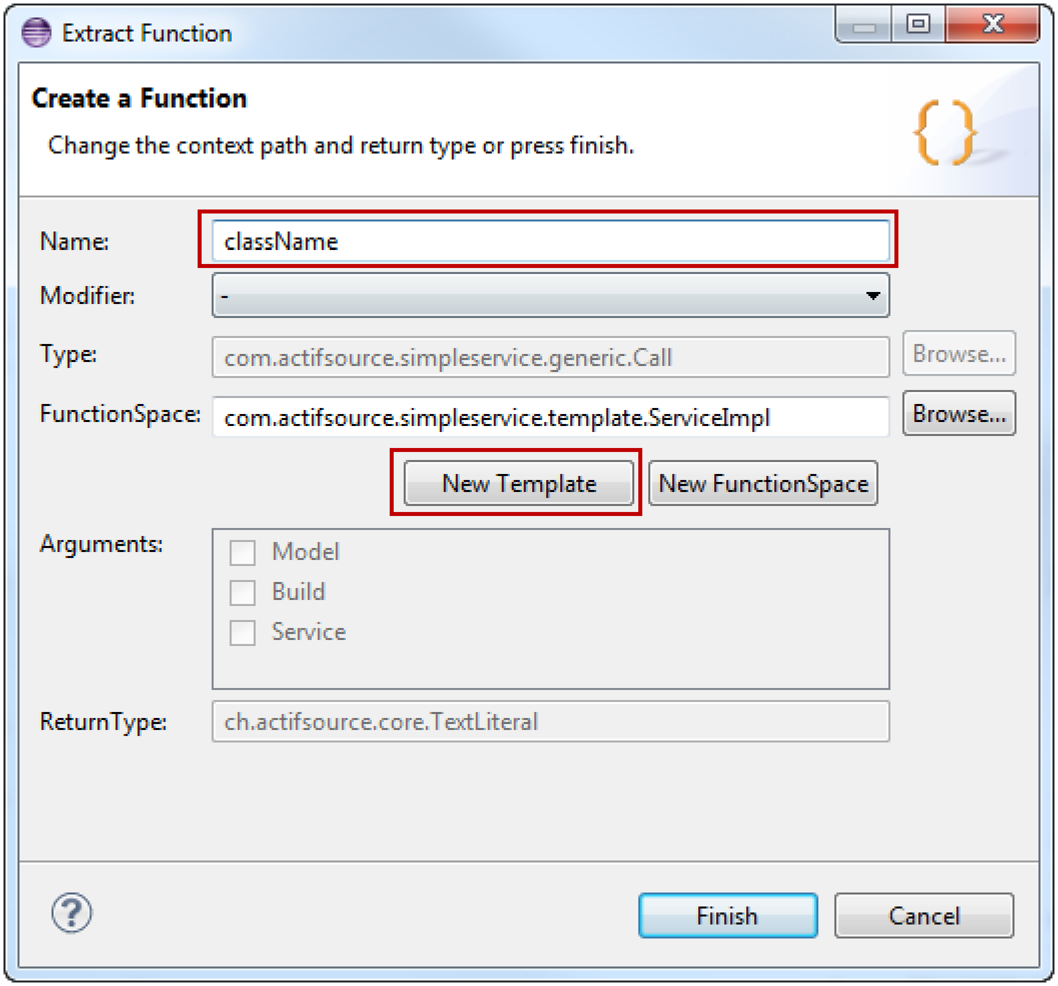
Extract Function in Function Space



Selector: Service.call

```
Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  private Call.nameImpl Call.name.toFirstLower@BuiltInCall;
}
```

- ⓘ Call.nameImpl shall be the name of a new Template
- Select the term Call.nameImpl
- Activate **QuickAssist** by clicking on the light bulb or pressing Ctrl+1
- Click *Extract TemplateFunction*
- ⓘ Template Functions behave like templates and are easier to handle than Java functions


Extract Function

Create a Function

Change the context path and return type or press finish.

Name: className

Modifier: -

Type: com.actifsource.simpleservice.generic.Call

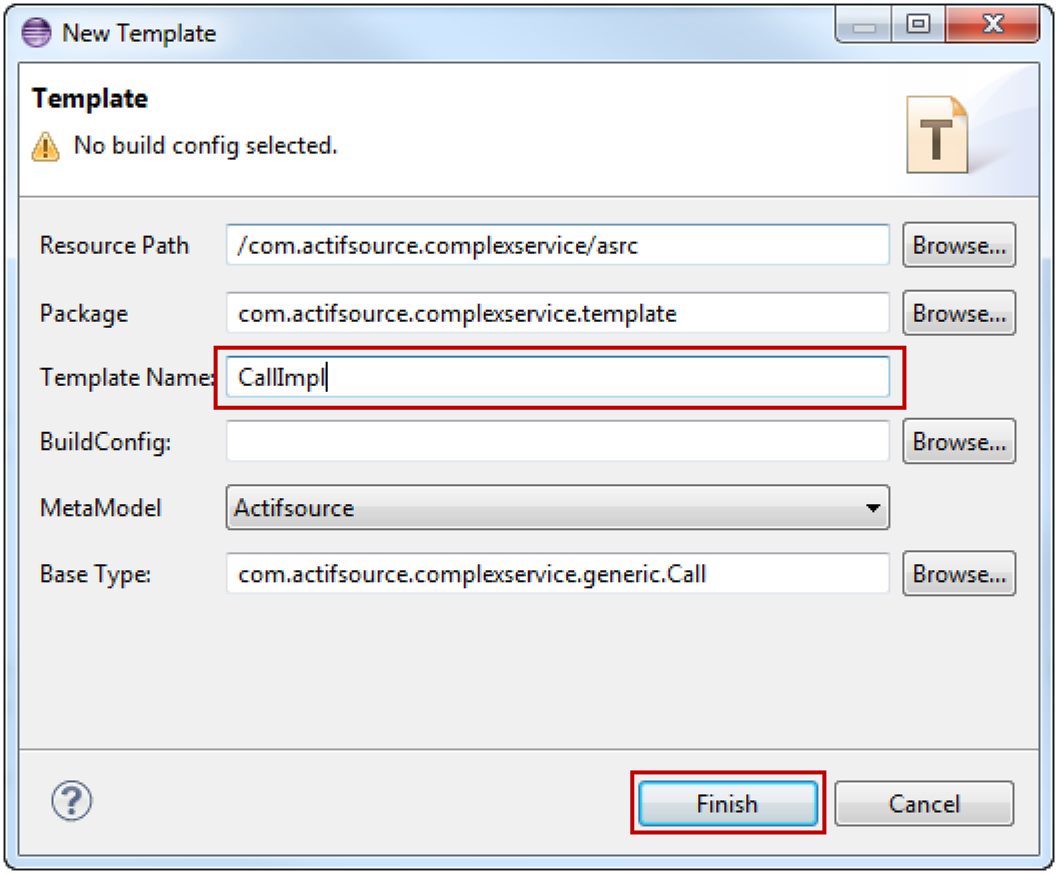
FunctionSpace: com.actifsource.simpleservice.template.ServiceImpl

Arguments:
- [ ] Model
- [ ] Build
- [ ] Service

ReturnType: ch.actifsource.core.TextLiteral

- Name the function `className`
- ⓘ Note that the default **Function Space** for this new function is the **Template** *ServiceImpl*
- Click *New Template* to create a new template which acts as **Function Space** for the new function *className*

Extract Function in Function Space



Template

No build config selected.

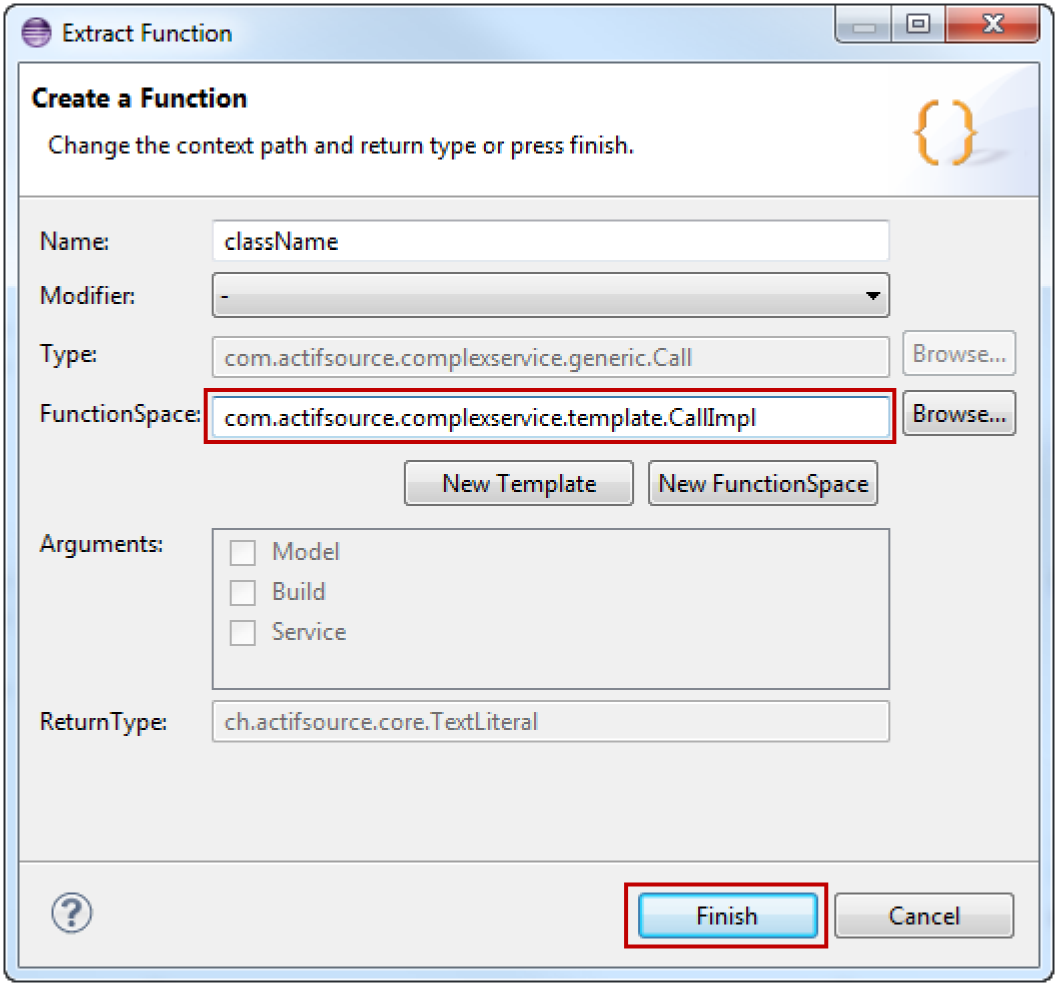
Resource Path	/com.actifsource.complexservice/asrc
Package	com.actifsource.complexservice.template
Template Name:	CallImpl
BuildConfig:	
MetaModel	Actifsource
Base Type:	com.actifsource.complexservice.generic.Call

ⓘ Check the **Package**

↳ Name the **Template** `CallImpl`

↳ Press *Finish*

Extract Function in Function Space



Create a Function

Change the context path and return type or press finish.

Name: className

Modifier: -

Type: com.actifsource.complexservice.generic.Call

FunctionSpace: com.actifsource.complexservice.template.CallImpl

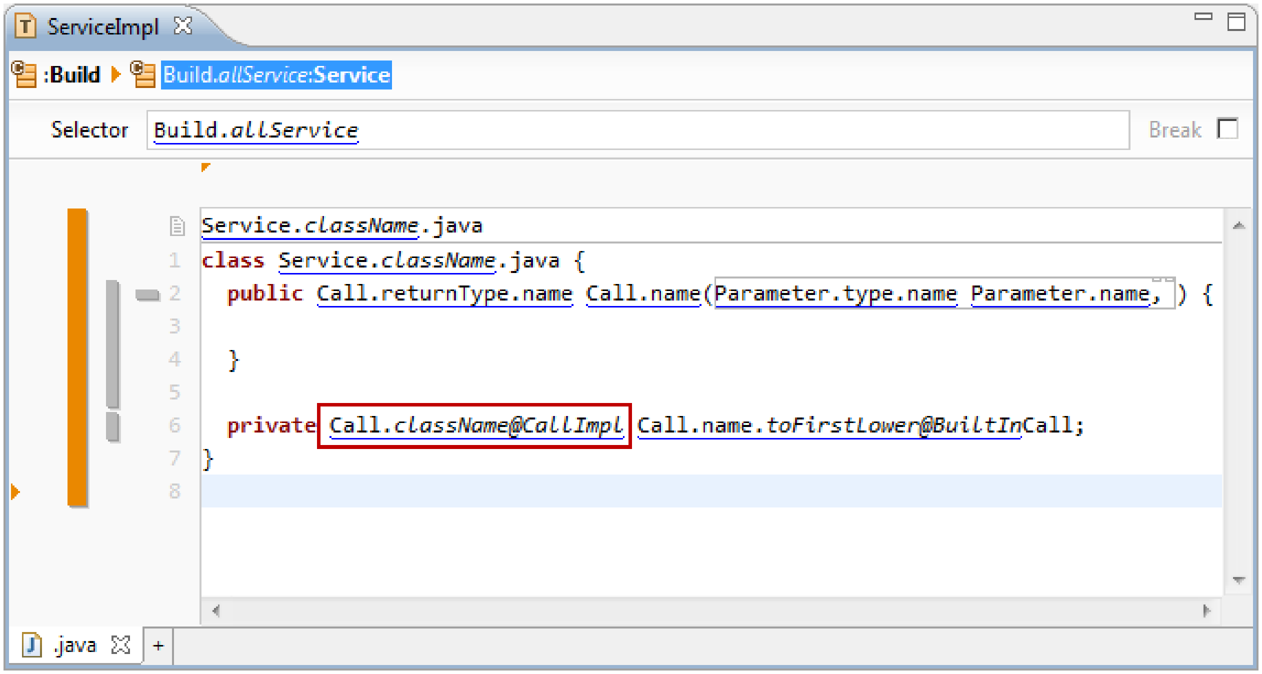
Arguments:
- [ ] Model
- [ ] Build
- [ ] Service

ReturnType: ch.actifsource.core.TextLiteral

ⓘ Note that the **Function Space** has been changed from *ServiceImpl* to *CallImpl*

Press *Finish*

Extract Function in Function Space



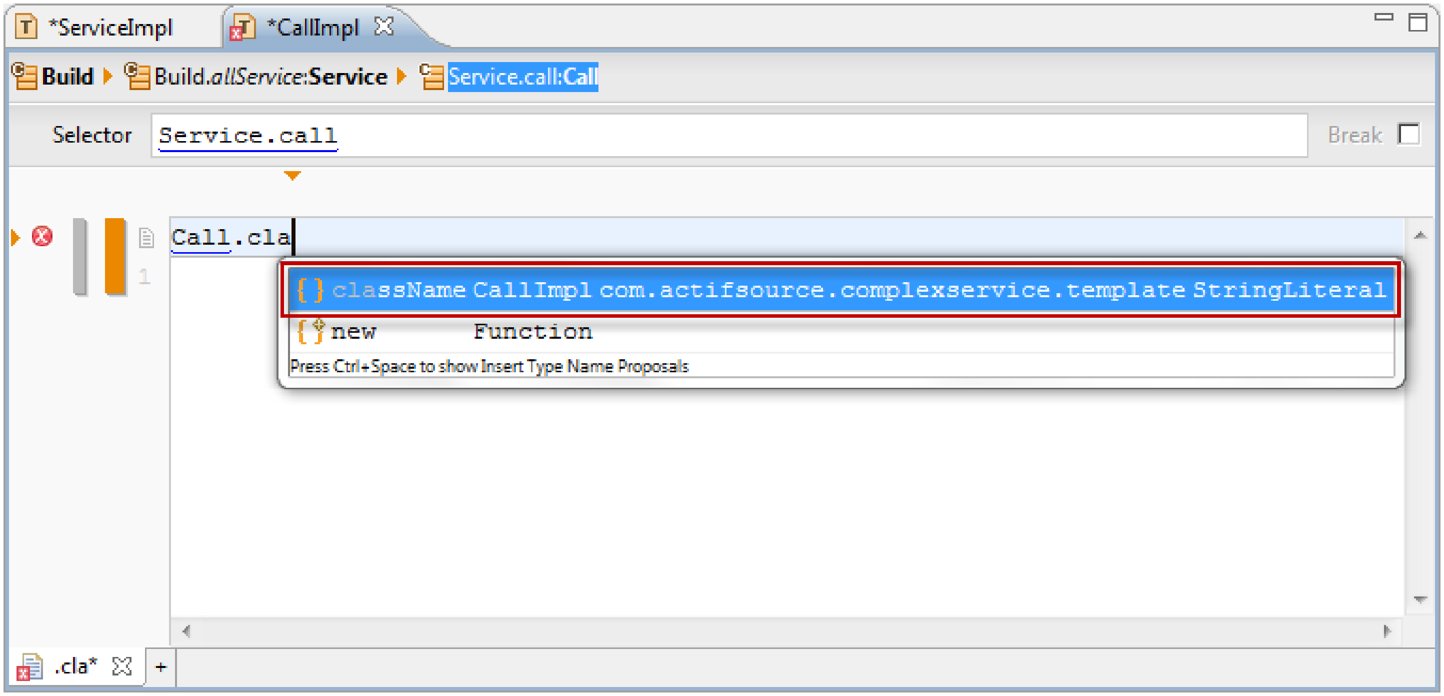
Selector Build.allService

```
Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  private Call.className@CallImpl Call.name.toFirstLower@BuiltInCall;
}
```

ⓘ The Term Call.nameImpl has been replaced by *Call.className@CallImpl*

ⓘ *className@CallImpl* indicates that the **Function** *className* belongs to the **Function Space** CallImpl

Extract Function in Function Space

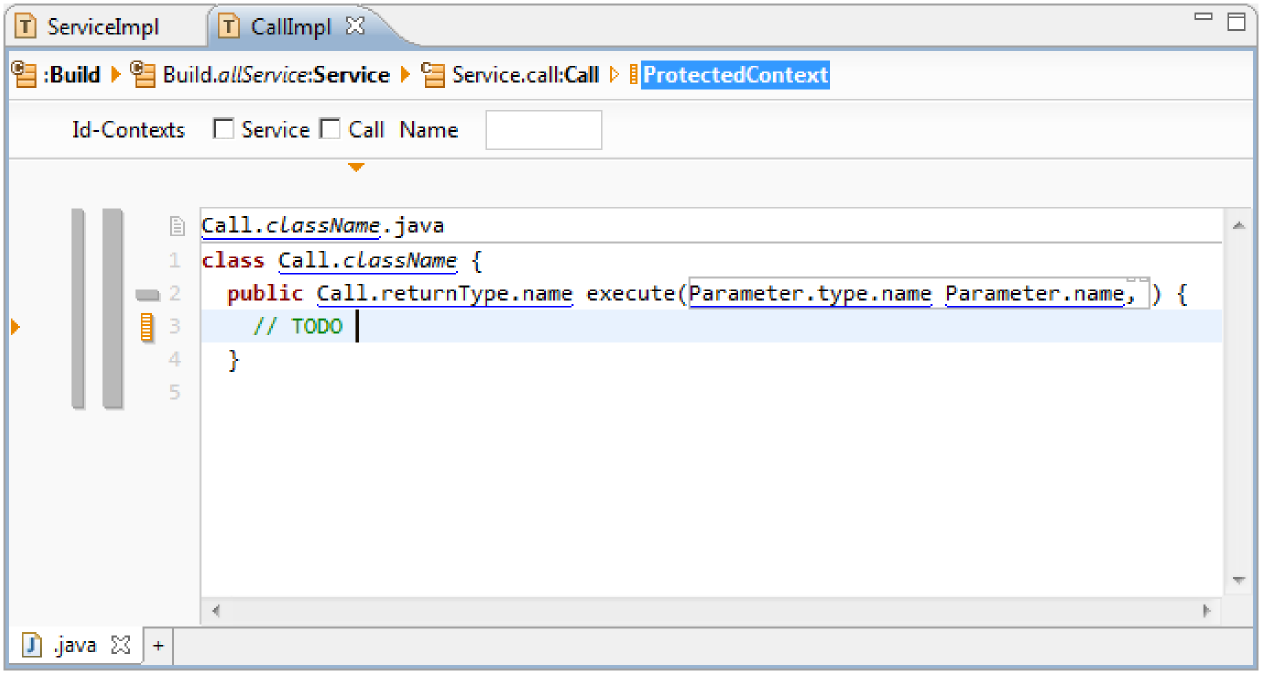


```
Service.call
```

```
Call.cla
{ } className CallImpl com.actifsource.complexservice.template StringLiteral
{ } new        Function
Press Ctrl+Space to show Insert Type Name Proposals
```

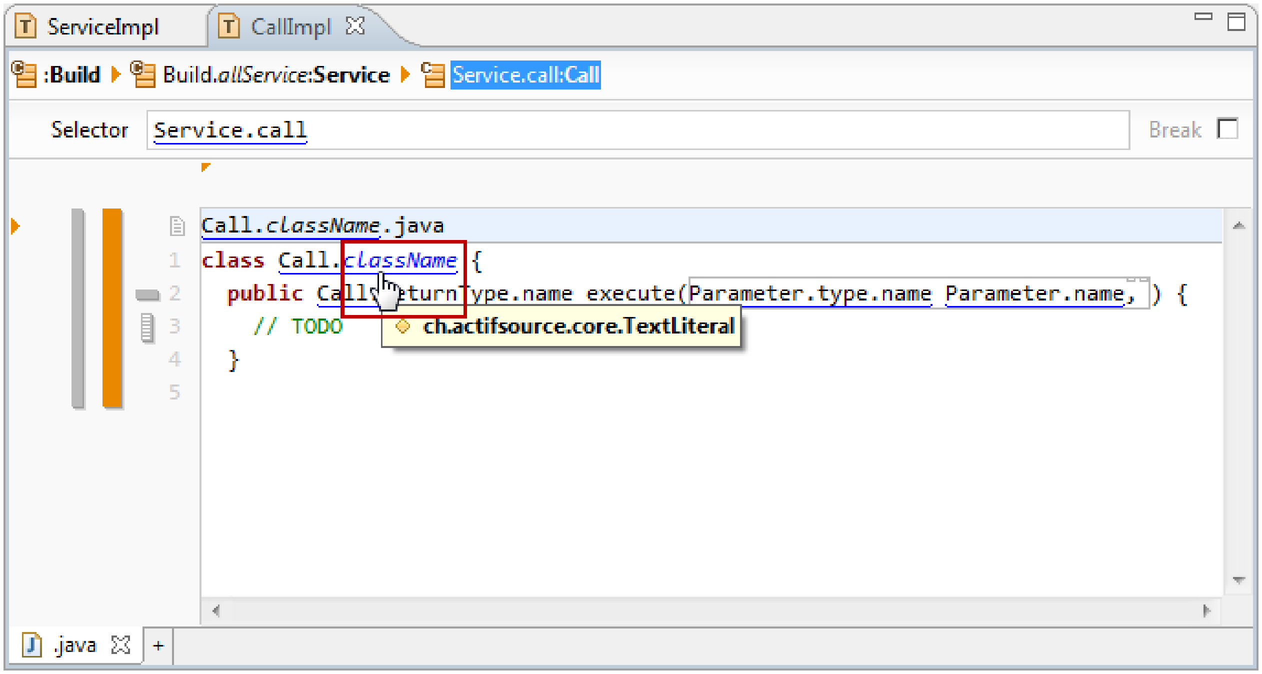
- ⓘ A new **Template** named *CallImpl* has been created in the **Package** *template*
- ↳ Use the **Function** *className* in the file line of your template
- ⓘ Note that *className* is the **Function** which we extracted in the **template** *ServiceImpl* before

Extract Function in Function Space



```
Call.className.java
class Call.className {
  public Call.returnType.name execute(Parameter.type.name Parameter.name, ) {
    // TODO
  }
```

- Write a simple class as shown above
- Write a method *execute* with returnType and Parameter
- Please notice that you might copy the whole parameter expression from the ServiceImpl **Template**
- Place a **Protected Context** in the function body

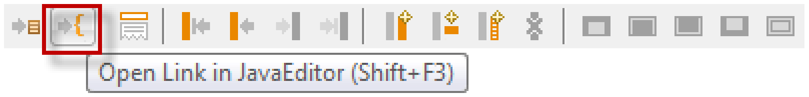


Service.call

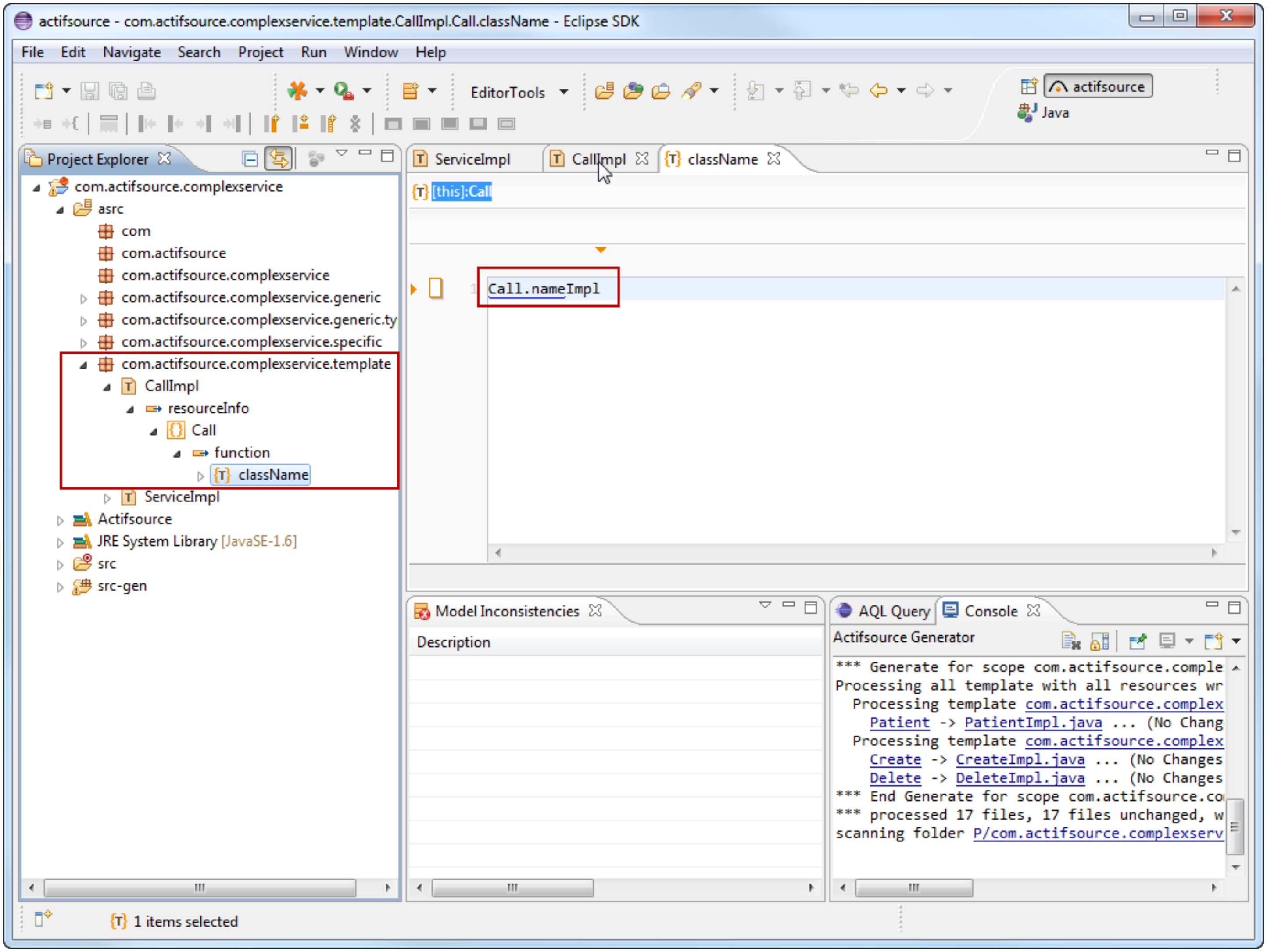
```
Call.className.java
class Call.className {
  public Call.returnType.name execute(Parameter.type.name Parameter.name, ) {
    // TODO
  }
```

↳ Open the underlying **Template Function** for className (Ctrl+Left-Click)

ⓘ Alternatively, you can use the **Tool** *Open Link in JavaEditor* from the **actifsource Template Editor** toolbar



Templates and Folders

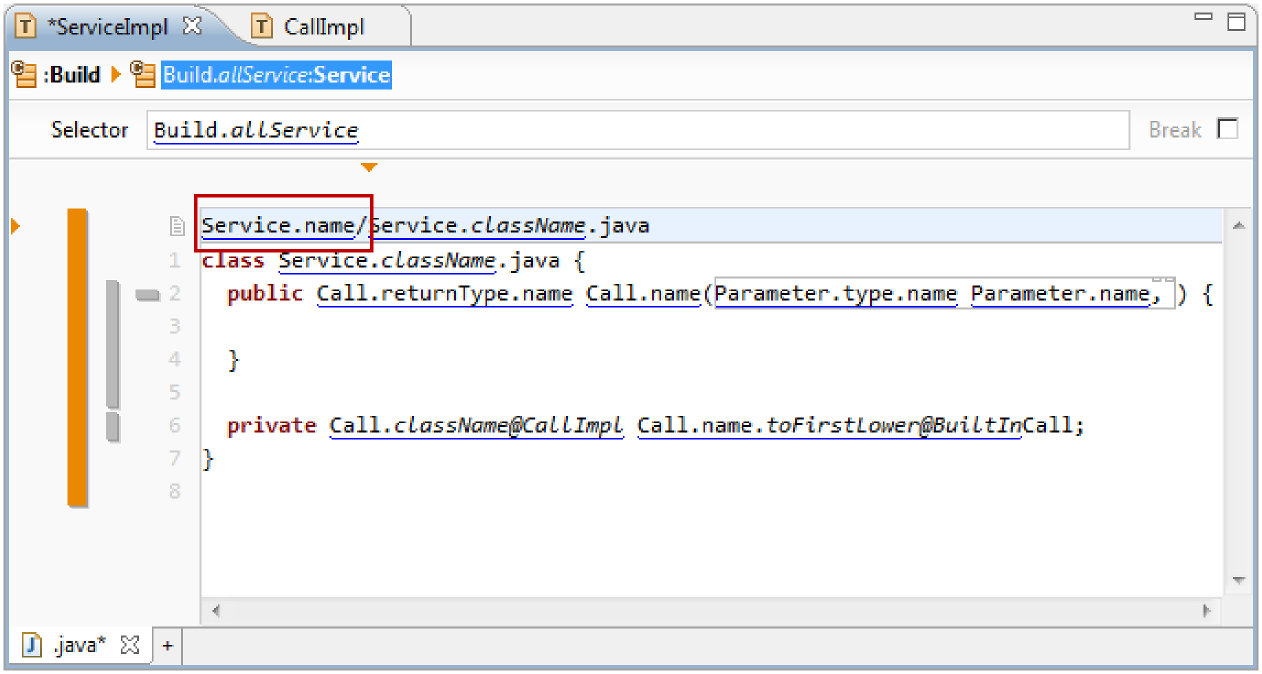


```
com.actifsource.complexservice
  asrc
    com
    com.actifsource
    com.actifsource.complexservice
    com.actifsource.complexservice.generic
    com.actifsource.complexservice.generic.ty
    com.actifsource.complexservice.specific
    com.actifsource.complexservice.template
      CallImpl
        resourceInfo
          Call
            function
              className
      ServiceImpl
  Actifsource
  JRE System Library [JavaSE-1.6]
  src
  src-gen
```

```
Call.nameImpl
```

```
*** Generate for scope com.actifsource.comple
Processing all template with all resources wr
  Processing template com.actifsource.complex
    Patient -> PatientImpl.java ... (No Chang
  Processing template com.actifsource.complex
    Create -> CreateImpl.java ... (No Changes
    Delete -> DeleteImpl.java ... (No Changes
*** End Generate for scope com.actifsource.co
*** processed 17 files, 17 files unchanged, w
scanning folder P/com.actifsource.complexserv
```

- ⓘ The function className is handled as a **Template Function** (partial template)
- ⓘ Template Functions are easy to handle
- ⓘ A **TemplateFunction** may call itself to follow recursive meta model designs (Composite Pattern)

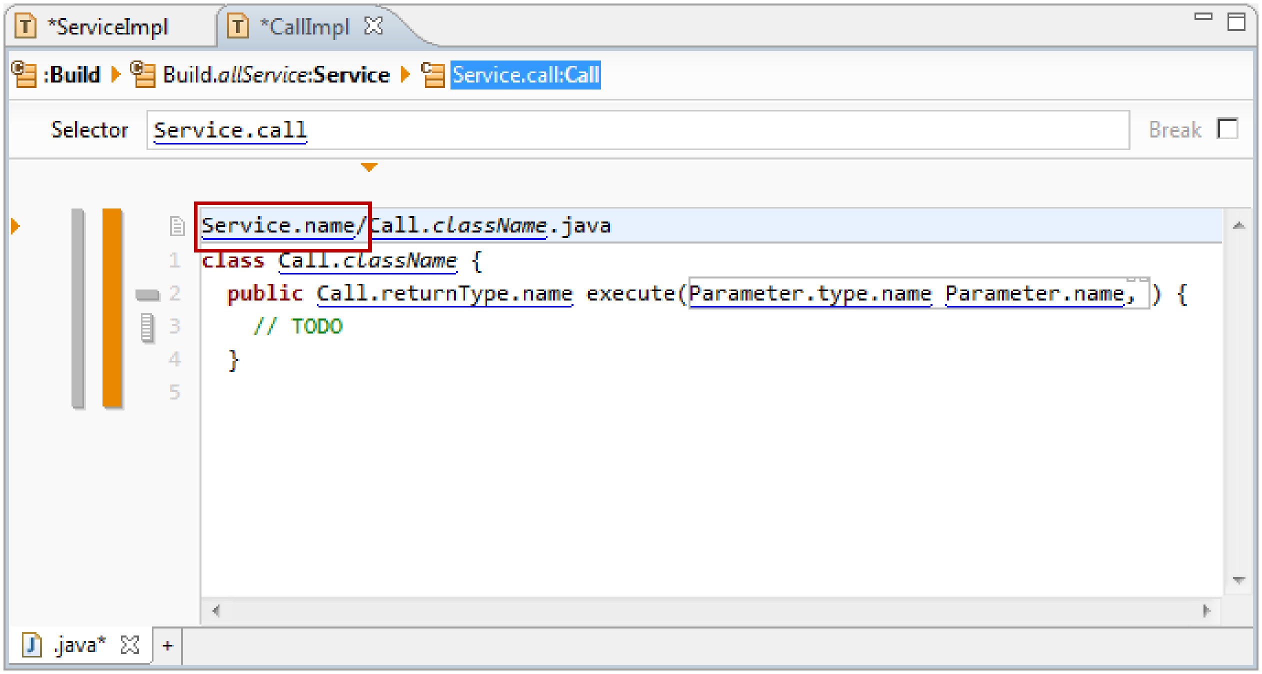


Selector: Build.allService

```
Service.name/Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  private Call.className@CallImpl Call.name.toFirstLower@BuiltInCall;
}
```

- ⓘ Generated artifacts are placed in the **Target Folder** of your project
- ⓘ You may want to place generated artifacts in specific sub folders
- Add Service.name/ as folder information in the file line of the **Template** *ServiceImpl* as shown above

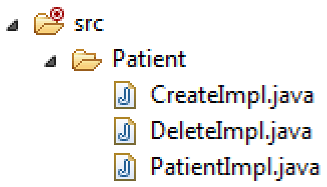
Templates and Folders



```
Service.call
```

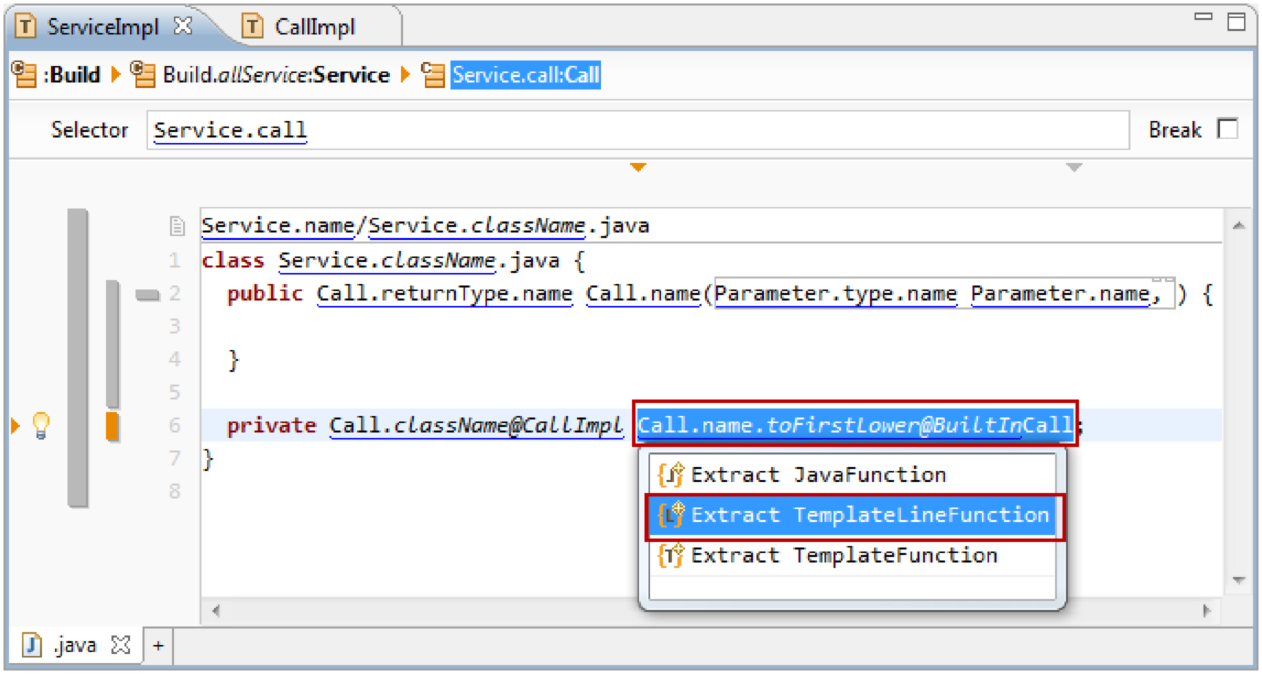
```
Service.name/Call.className.java
class Call.className {
  public Call.returnType.name execute(Parameter.type.name Parameter.name, ) {
    // TODO
  }
```

- ⓘ We want all Call implementations to be generated in the same folder as their corresponding Service
- Add Service.name/ as the folder name in the file line of the Template CallImpl as shown above
- Save the **Templates** *CallImpl* and *ServiceImpl*
- ⓘ Note that files generated from this template are moved to the new location automatically
- ⓘ **Protected Regions** of the generated files are preserved



```
src
  Patient
    CreateImpl.java
    DeleteImpl.java
    PatientImpl.java
```

Copy with Context

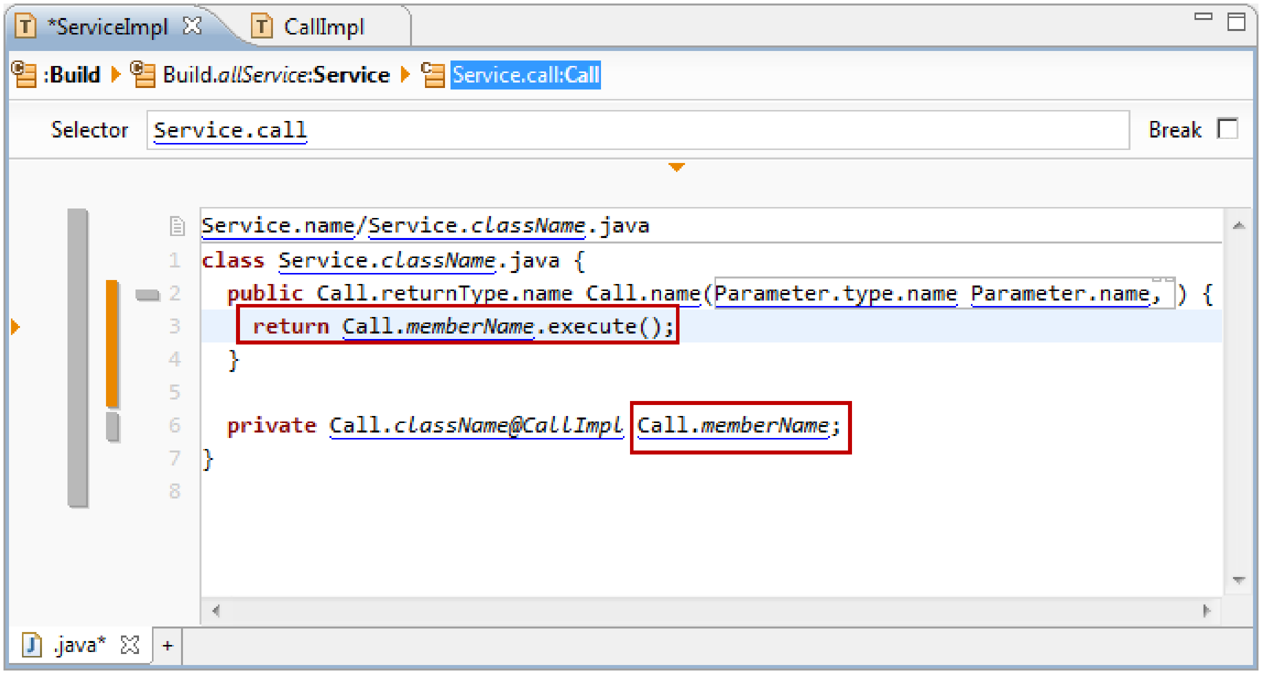


Service.call

```
Service.name/Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  private Call.className@CallImpl Call.name.toFirstLower@BuiltInCall;
}
```

- Extract a TemplateLineFunction *memberName* for the member variable name

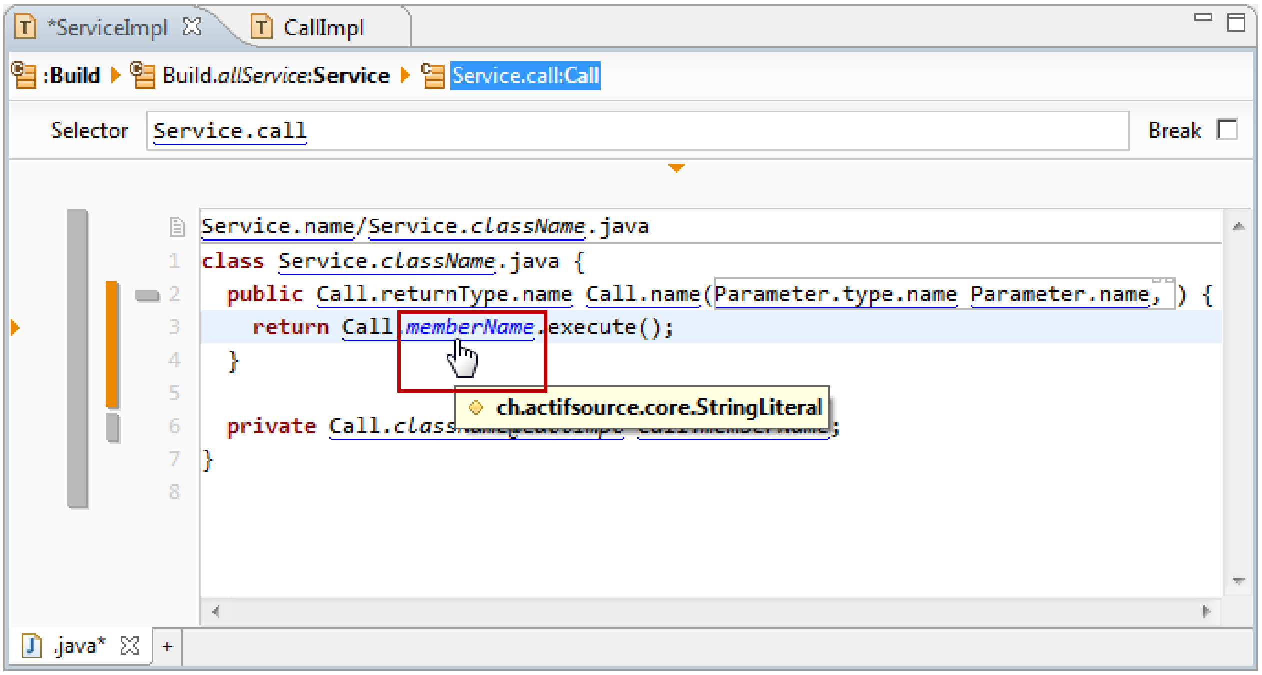
Copy with Context



Selector Service.call Break ☐

```
Service.name/Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {
    return Call.memberName.execute();
  }

  private Call.className@CallImpl Call.memberName;
}
```

↳ Also use the function *memberName* in the function body as shown above

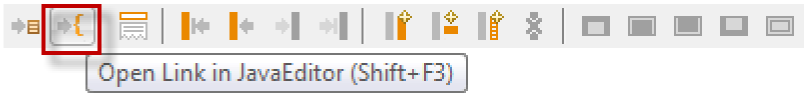


Selector: Service.call

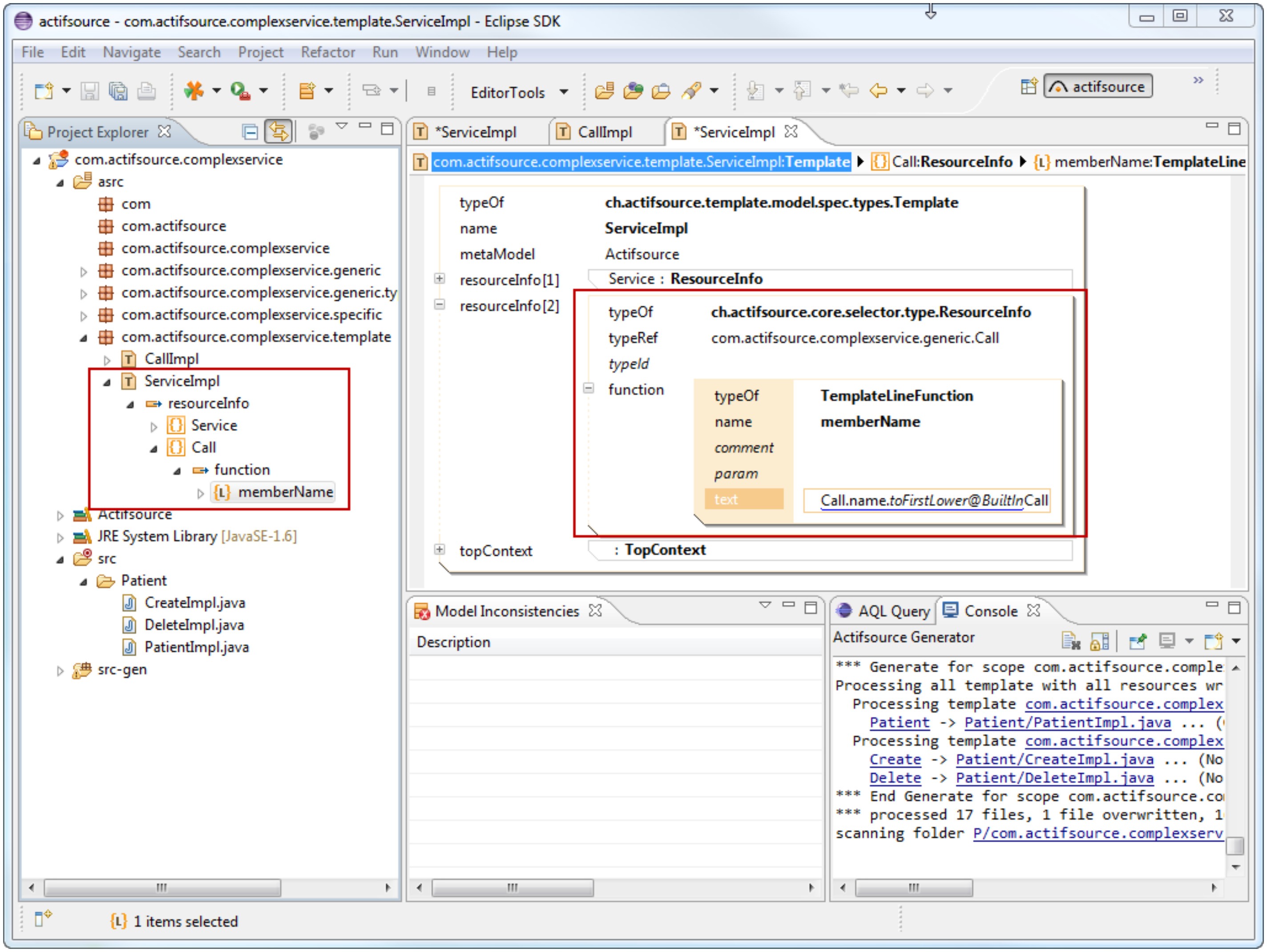
```
Service.name/Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {
    return Call.memberName.execute();
  }

  private Call.className@DefaultImpl Call.memberName;
}
```

ch.actifsource.core.StringLiteral

- Open the underlying **TemplateLineFunction** for memberName (Ctrl+Left-Click)
- Alternatively, you can use the **Tool** *Open Link in JavaEditor* from the **actifsource Template Editor** toolbar



Templates and Folders

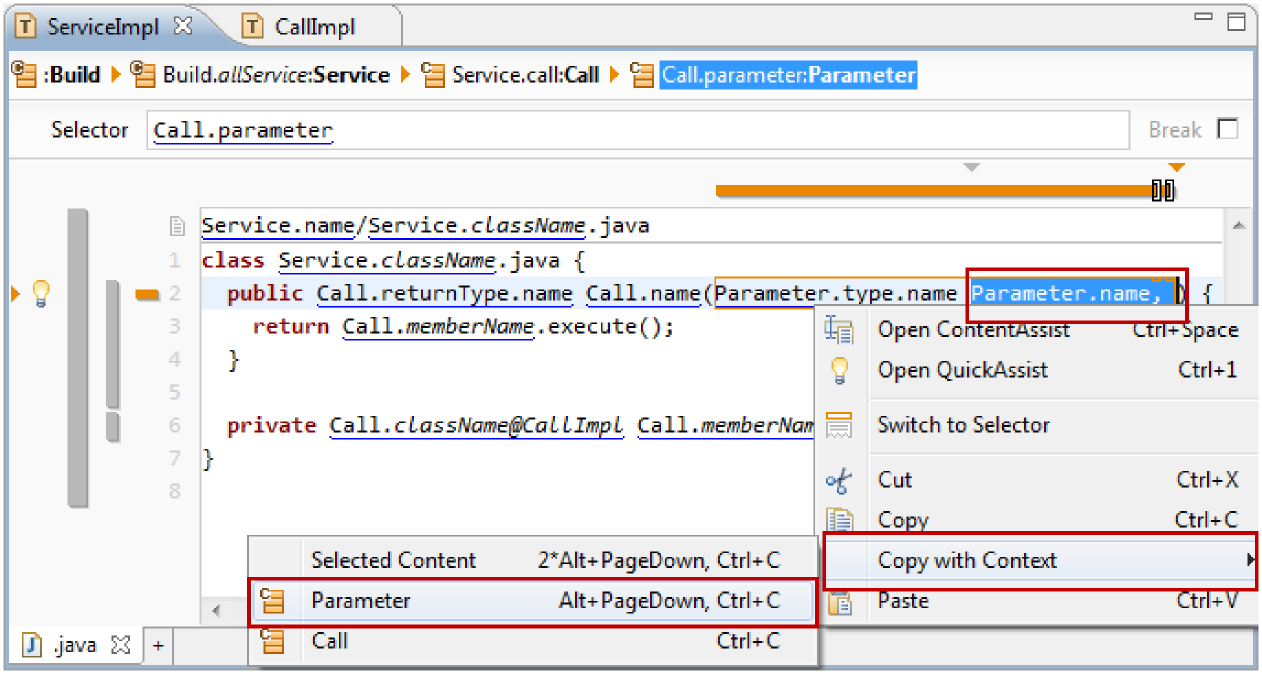

actifsource - com.actifsource.complexservice.template.ServiceImpl - Eclipse SDK
File Edit Navigate Search Project Refactor Run Window Help
EditorTools
actifsource
Project Explorer
com.actifsource.complexservice
asrc
com
com.actifsource
com.actifsource.complexservice
com.actifsource.complexservice.generic
com.actifsource.complexservice.generic.ty
com.actifsource.complexservice.specific
com.actifsource.complexservice.template
CallImpl
ServiceImpl
resourceInfo
Service
Call
function
memberName
Actifsource
JRE System Library [JavaSE-1.6]
src
Patient
CreateImpl.java
DeleteImpl.java
PatientImpl.java
src-gen
*ServiceImpl
CallImpl
*ServiceImpl
com.actifsource.complexservice.template.ServiceImpl:Template
Call:ResourceInfo
memberName:TemplateLine
Model Inconsistencies
Description
AQL Query
Console
Actifsource Generator
1 items selected

typeOf	ch.actifsource.template.model.spec.types.Template
name	ServiceImpl
metaModel	Actifsource
resourceInfo[1]	Service : ResourceInfo
resourceInfo[2]	typeOf: ch.actifsource.core.selector.type.ResourceInfo; typeRef: com.actifsource.complexservice.generic.Call; typeId; function: typeOf TemplateLineFunction, name memberName, comment, param, text Call.name.toFirstLower@BuiltInCall
topContext	: TopContext

```
*** Generate for scope com.actifsource.comple
Processing all template with all resources wr
  Processing template com.actifsource.complex
    Patient -> Patient/PatientImpl.java ... (
  Processing template com.actifsource.complex
    Create -> Patient/CreateImpl.java ... (No
    Delete -> Patient/DeleteImpl.java ... (No
*** End Generate for scope com.actifsource.co
*** processed 17 files, 1 file overwritten, 1
scanning folder P/com.actifsource.complexserv
```

- ⓘ The function <u>className</u> is handled as a **TemplateLineFunction**
- ⓘ Template Line Functions are the easiest way to reuse information
- ⓘ Template Line Functions do not allow context

Copy with Context



```
Call.parameter
```

```
Service.name/Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {
    return Call.memberName.execute();
  }

  private Call.className@CallImpl Call.memberNam
}
```

ⓘ We want to copy Parameter.name including the Parameter **Context** and the separating comma from the functions parameter list

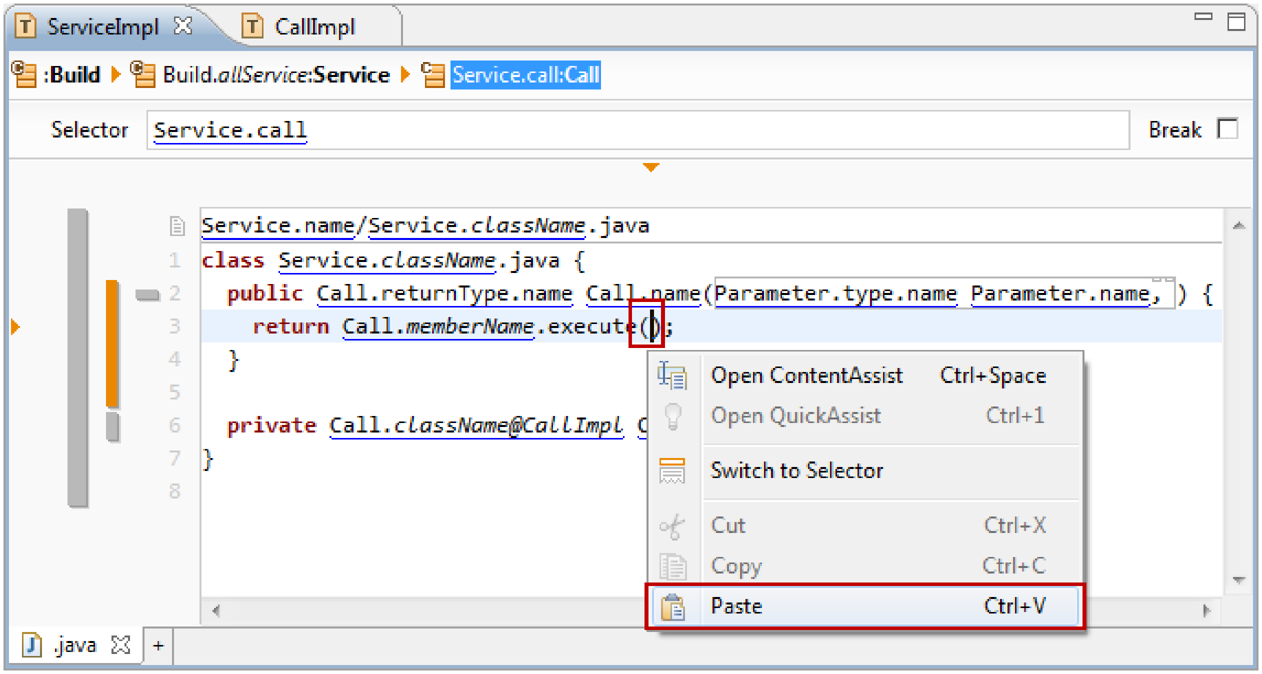
↳ Select the Term "`Parameter.name, `"

↳ From the *Context Menu*, select *Copy with Context*

↳ From the *Subcontext Menu*, select Parameter

ⓘ Note also the shortcuts Alt+PageUp to select the parent context, and ; Ctrl+C to copy a context

Copy with Context

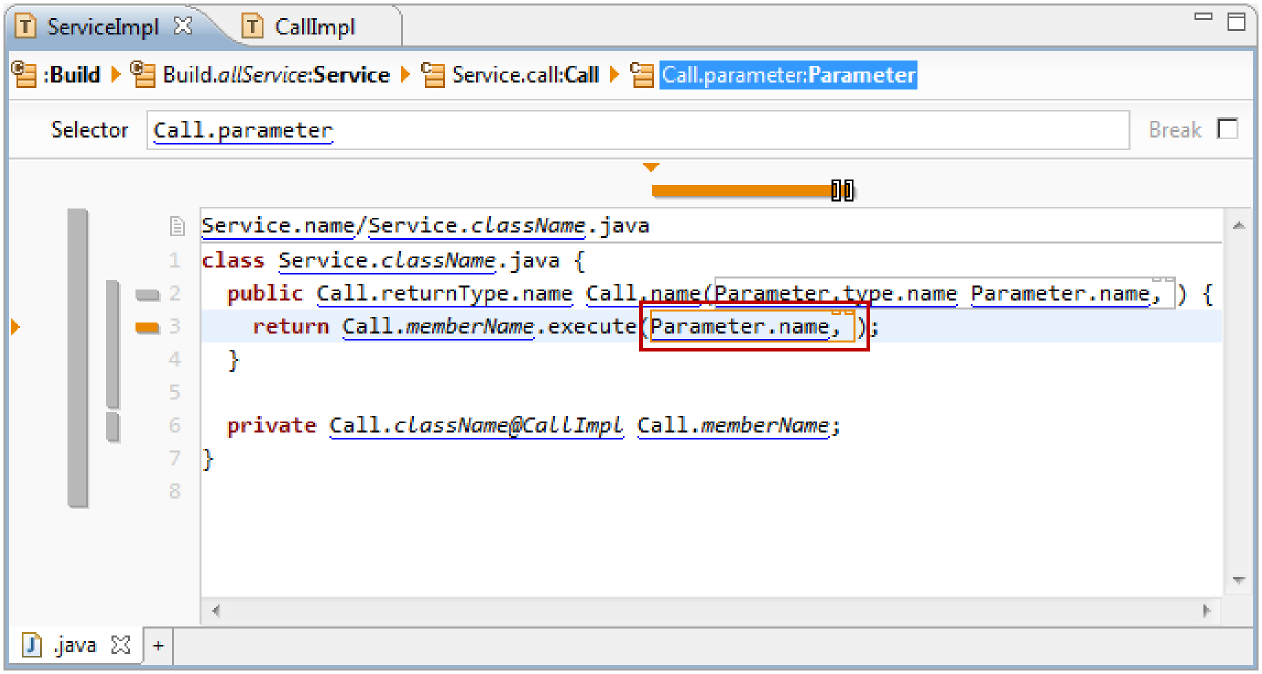


Selector: Service.call

```
Service.name/Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {
    return Call.memberName.execute();
  }

  private Call.className@CallImpl C
}
```

- Place your cursor between the brackets
- Select *Paste* from the *Context Menu* (Ctrl+V)

Copy with Context



Selector: Call.parameter

```
Service.name/Service.className.java
class Service.className.java {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {
    return Call.memberName.execute(Parameter.name, );
  }

  private Call.className@CallImpl Call.memberName;
}
```

ⓘ The text and its corresponding context are inserted