Tutorial

Complex Service

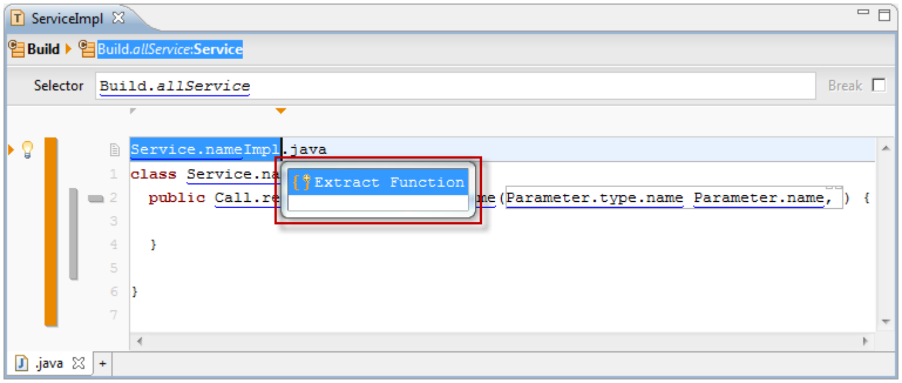
© 2010 actifsource GmbH, Switzerland - all rights reserved.

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 • 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.

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Overview

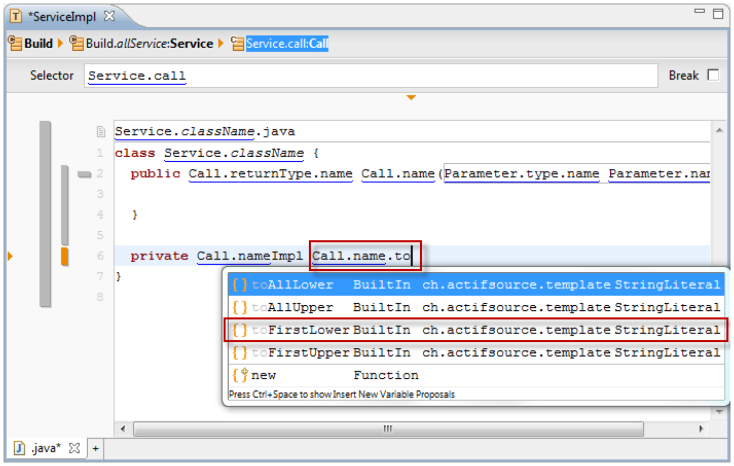
- Prepare a new actifsource Project as seen in the Actifsource Tutorial – Simple Service
- Learn how to extract Java Functions from template code to cope with complex situations



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

  }

}
```

Selector: `Build.allService`

- 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



Selector: `Service.call`

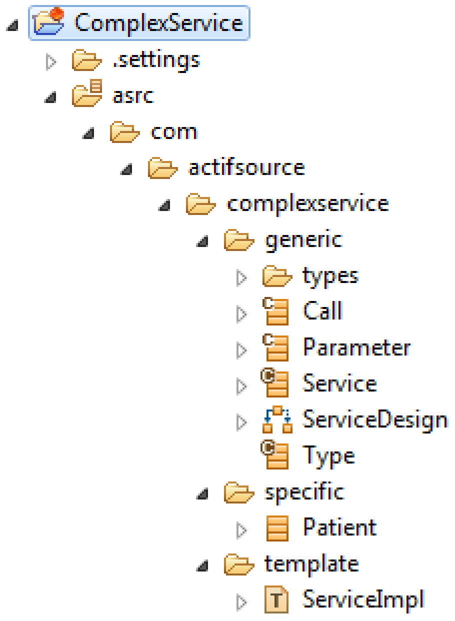
```
Service.className.java
class Service.className {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.nar

  }

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

```
toAllLower   BuiltIn  ch.actifsource.template StringLiteral
toAllUpper   BuiltIn  ch.actifsource.template StringLiteral
toFirstLower BuiltIn  ch.actifsource.template StringLiteral
toFirstUpper BuiltIn  ch.actifsource.template StringLiteral
new          Function
Press Ctrl+Space to show Insert New Variable Proposals
```

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

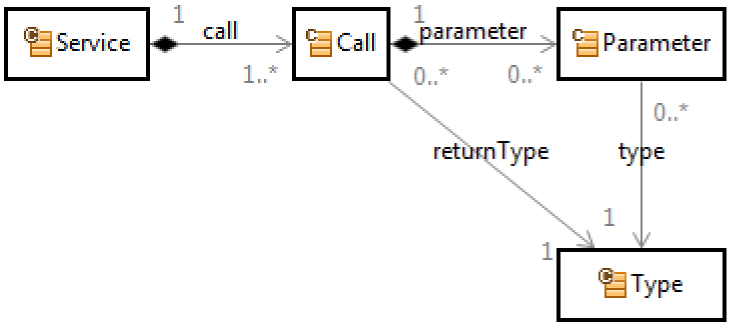
Part I: Preparation

- 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



```
ComplexService
  .settings
  asrc
    com
      actifsource
        complexservice
          generic
            types
            Call
            Parameter
            Service
            ServiceDesign
            Type
          specific
            Patient
          template
            ServiceImpl
```

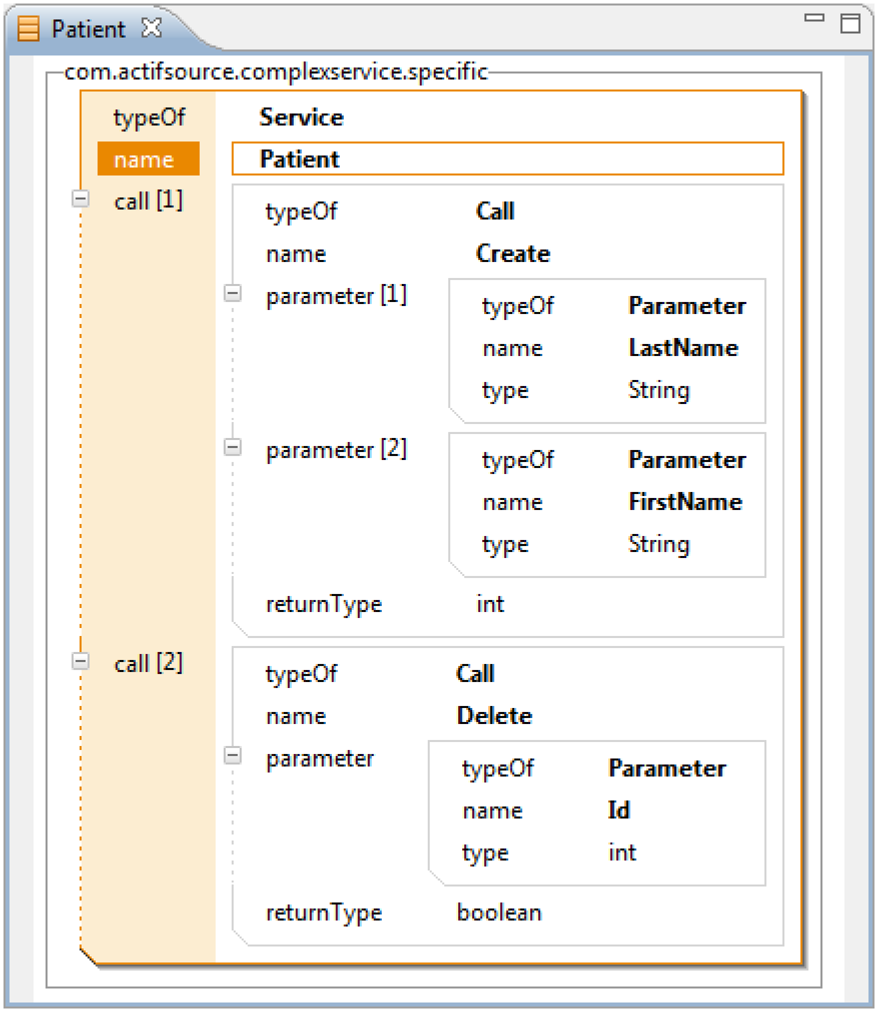
© 2010 actifsource GmbH, Switzerland - all rights reserved.



- 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

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Create a Specific Domain Model



Patient

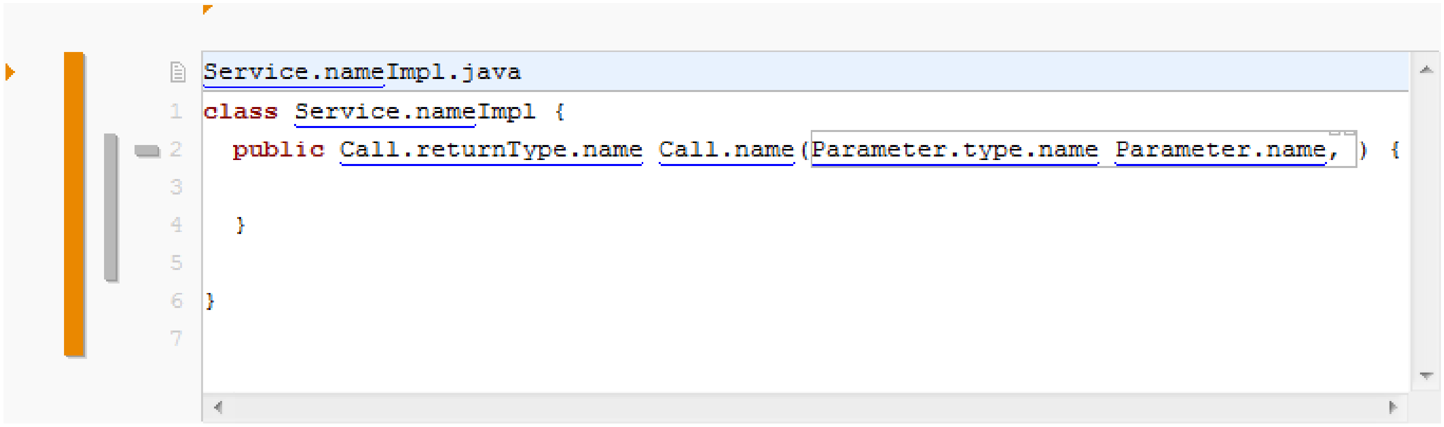
com.actifsource.complexservice.specific

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

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Create a Code Template

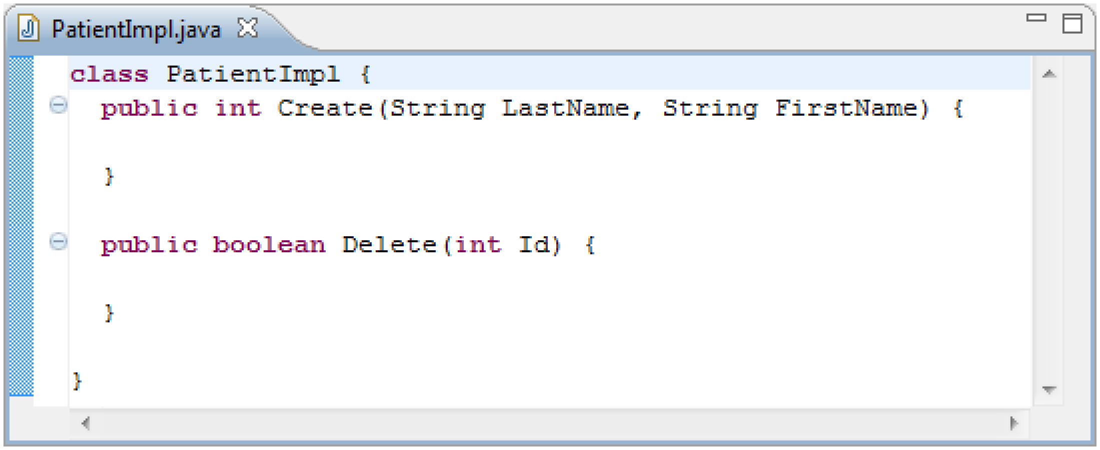


```
Service.nameImpl.java
class Service.nameImpl {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

}
```

- 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**

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Generate Code



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

  }

  public boolean Delete(int Id) {

  }

}
```

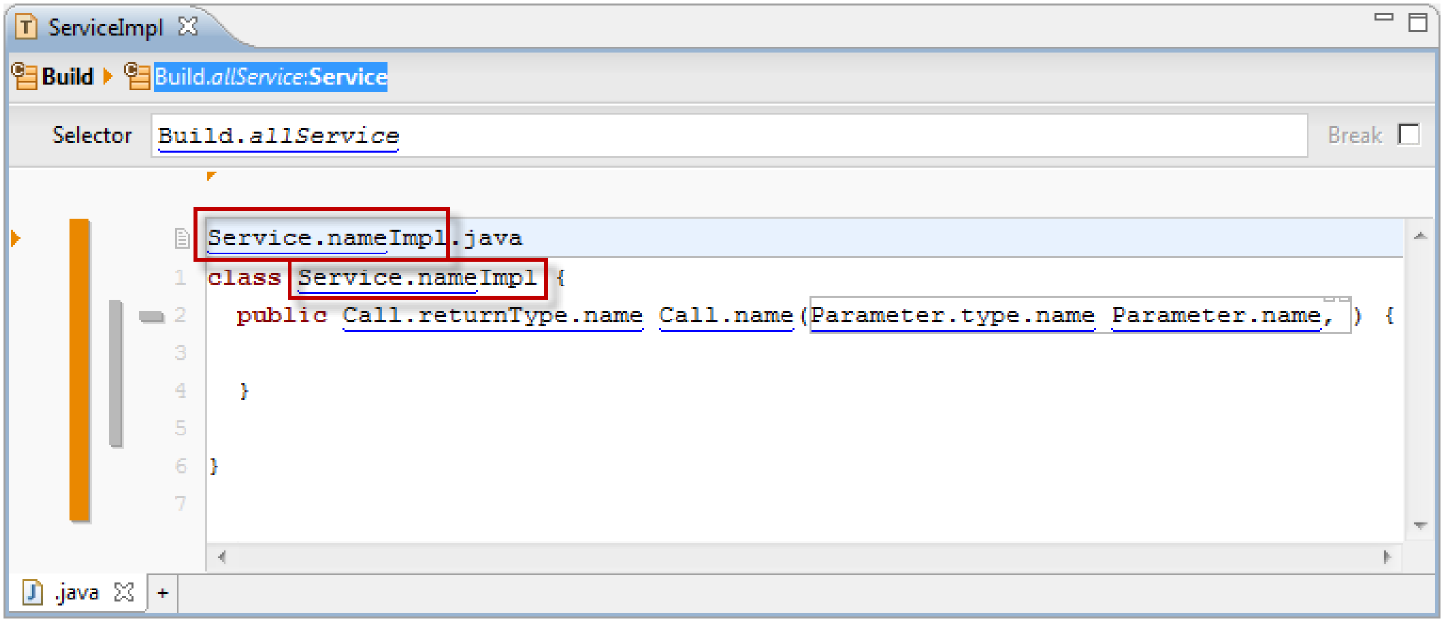
ⓘ You’ll find the generated code *PatientImpl.java* in the **Target Folder** *src*

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**

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Extract Function

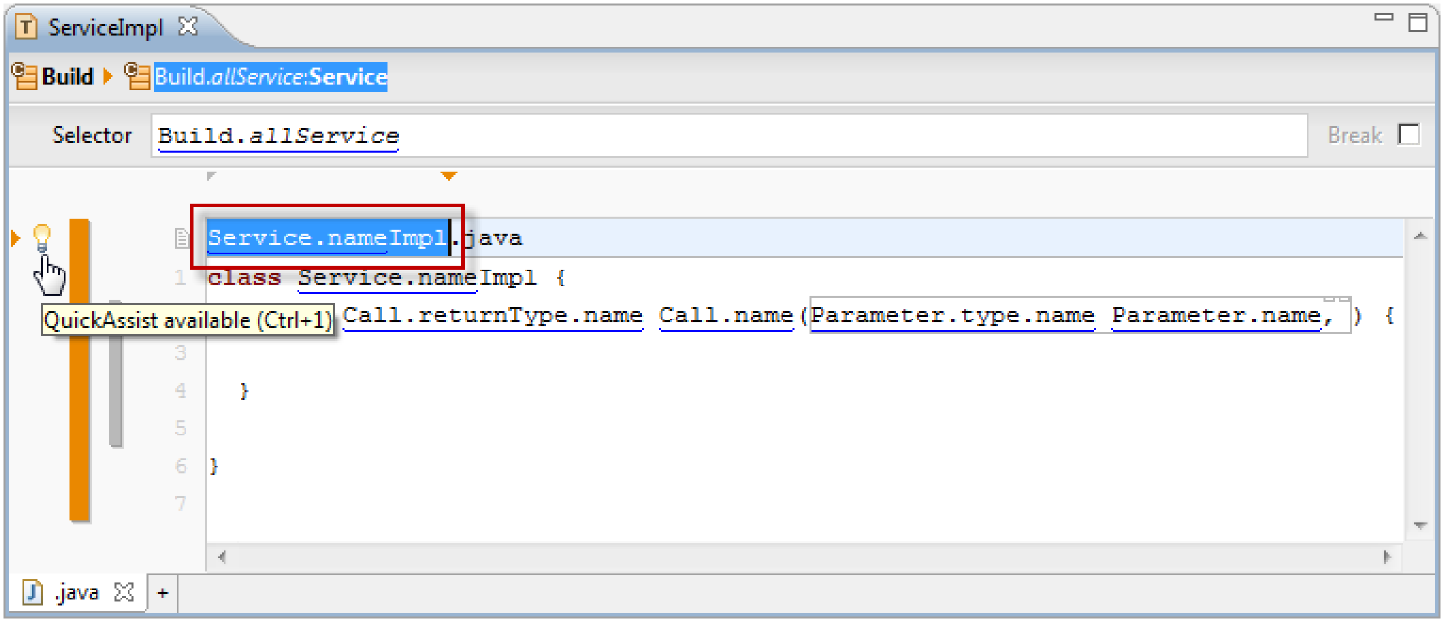


```
Build.allService
```

```
Service.nameImpl.java
class Service.nameImpl {
  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

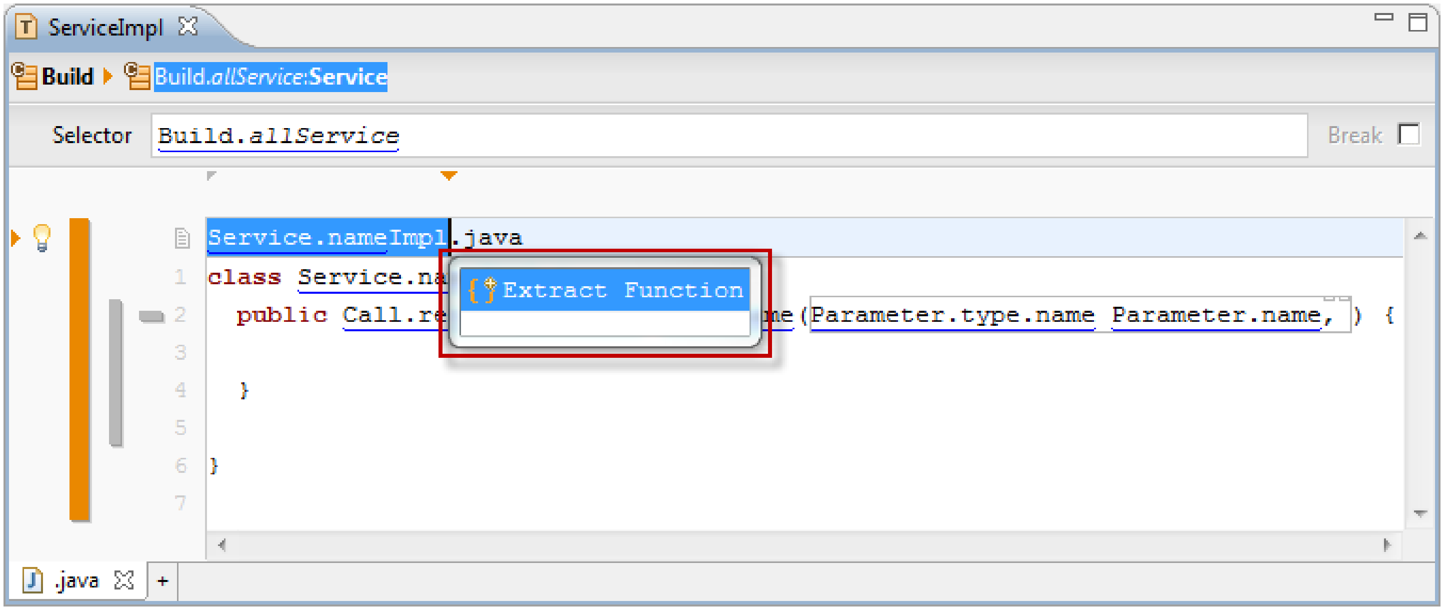


```
Build.allService
```

```
Service.nameImpl.java
class Service.nameImpl {
  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

© 2010 actifsource GmbH, Switzerland - all rights reserved.

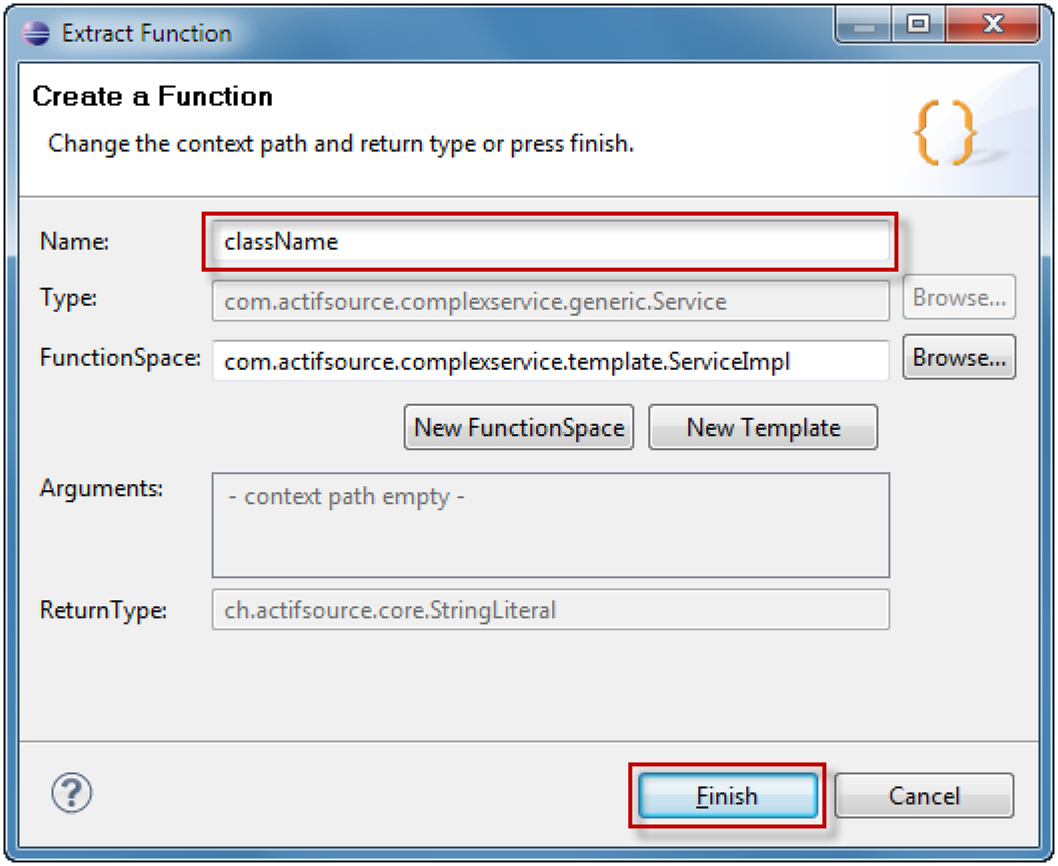


```
Build.allService
```

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

  }

}
```

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

© 2010 actifsource GmbH, Switzerland - all rights reserved.



Change the context path and return type or press finish.

Name: className

Type: com.actifsource.complexservice.generic.Service

FunctionSpace: com.actifsource.complexservice.template.ServiceImpl

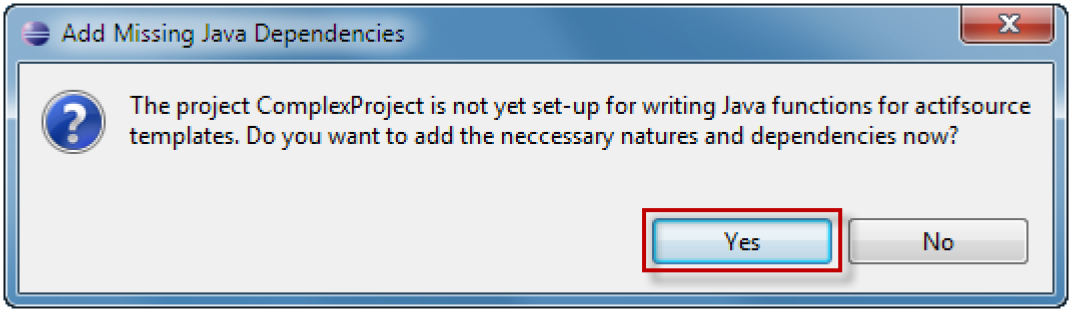
Arguments: - context path empty -

ReturnType: ch.actifsource.core.StringLiteral

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

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Enable Java Functions for Templates

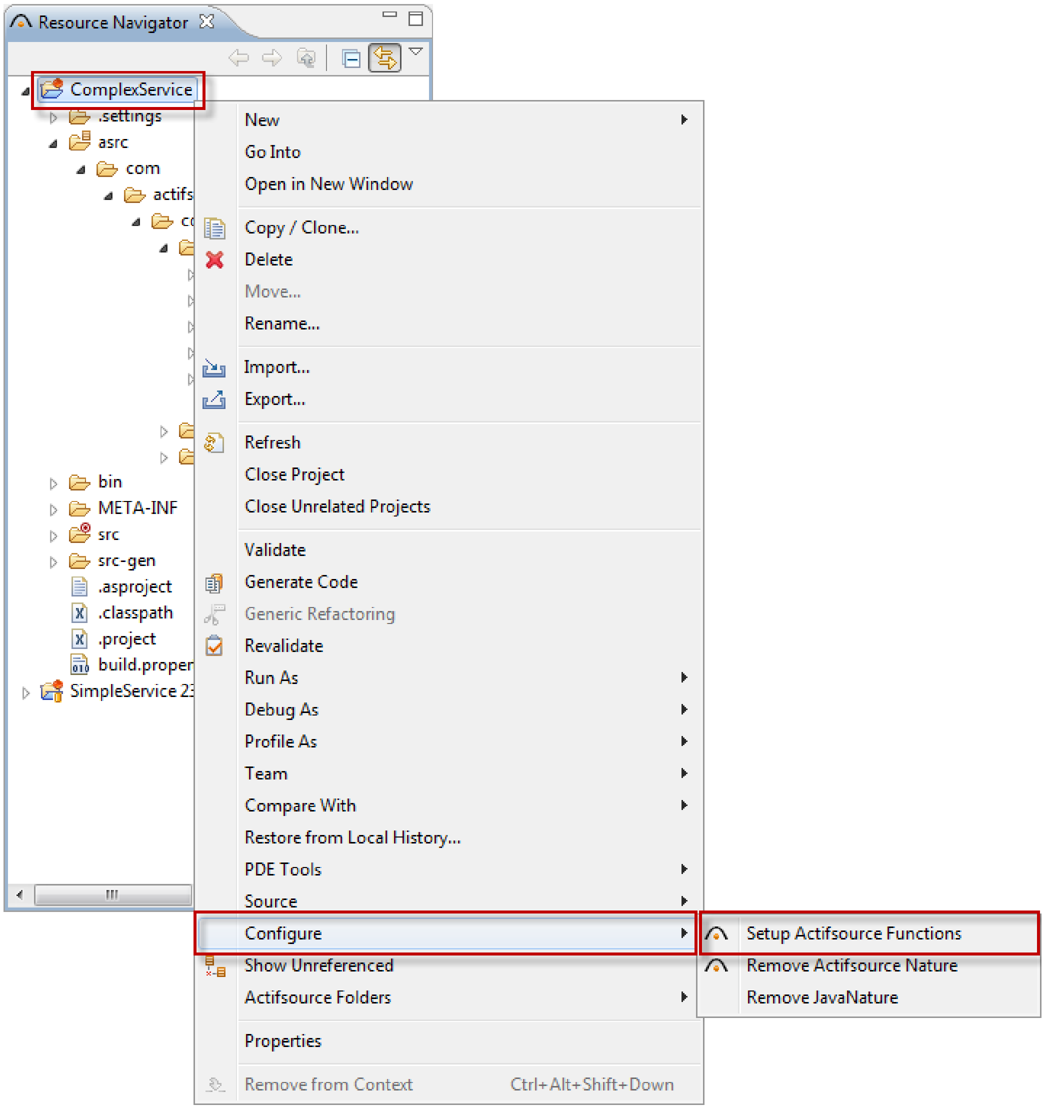

Add Missing Java Dependencies

The project ComplexProject is not yet set-up for writing Java functions for actifsource templates. Do you want to add the neccessary natures and dependencies now?

- ⓘ **Java Functions** are evaluated during code generation
- ⓘ Therefore, your Project has to be a **Java Project**
- ↳ Click *Yes* to add the necessary settings to your Project

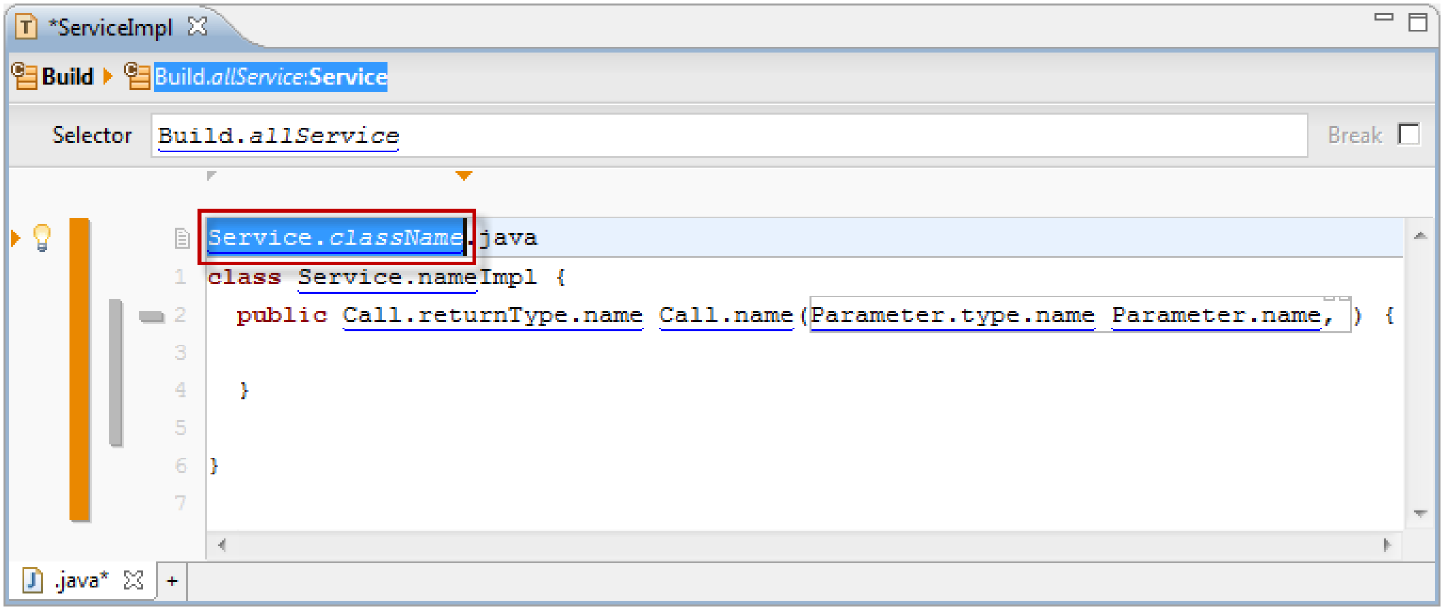
© 2010 actifsource GmbH, Switzerland - all rights reserved.

Enable Java Functions for Templates



ⓘ You can enable **Java Functions** anytime as shown above

Using Functions



Selector: `Build.allService`

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

  }

}
```

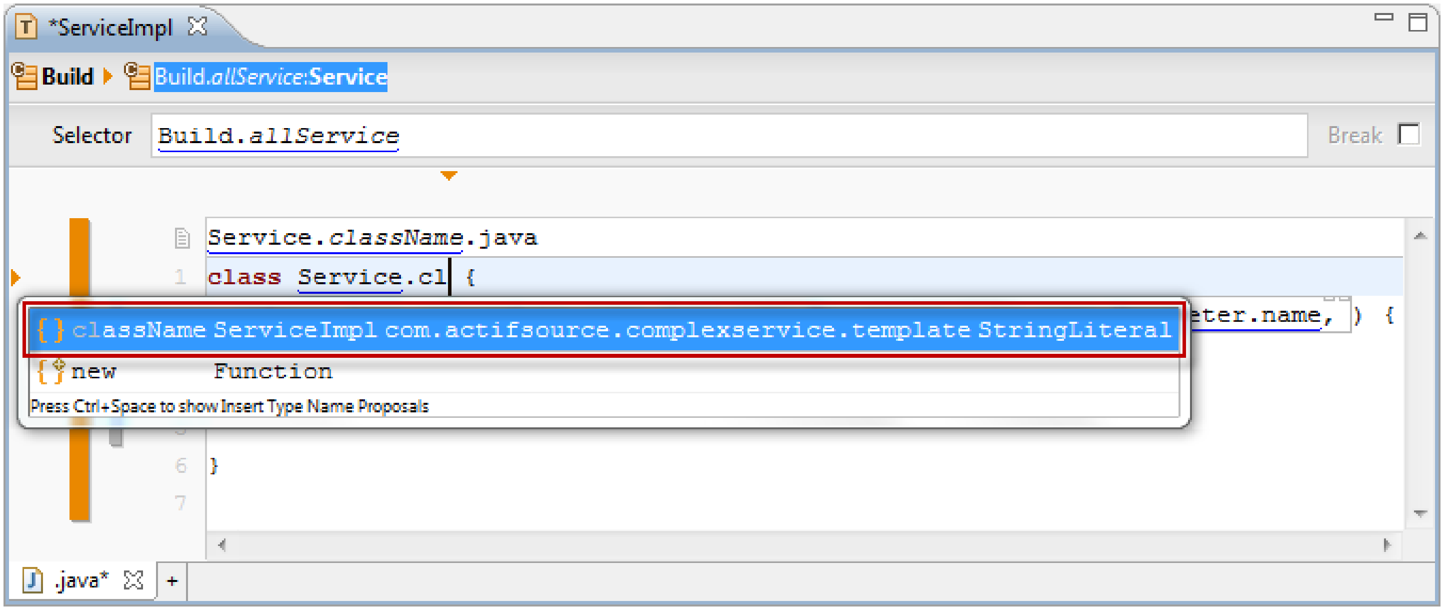
ⓘ The new function className returns the extracted fragment from your template

ⓘ 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**

© 2010 actifsource GmbH, Switzerland - all rights reserved.



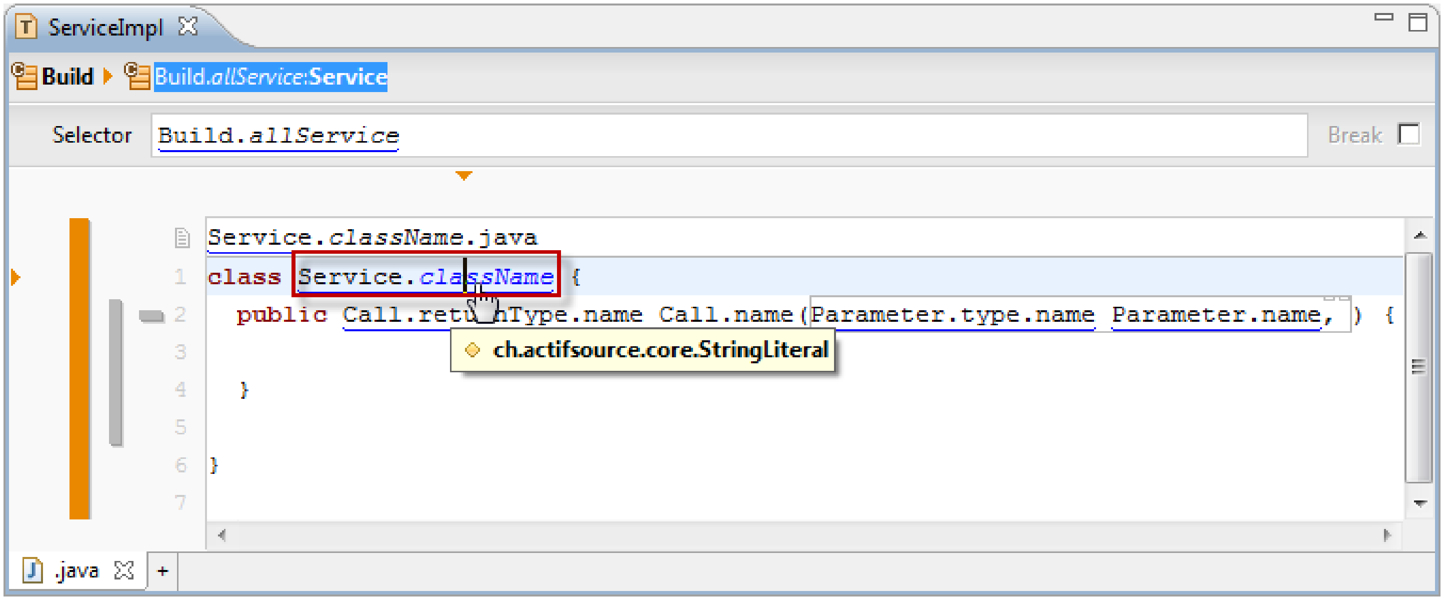
```
Build.allService
```

```
Service.className.java
class Service.cl {
eter.name, ) {
}
```

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

ⓘ 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

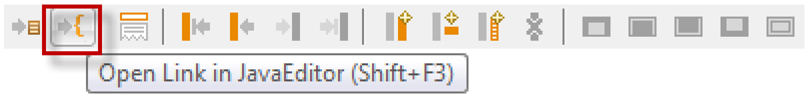


```
Build.allService
```

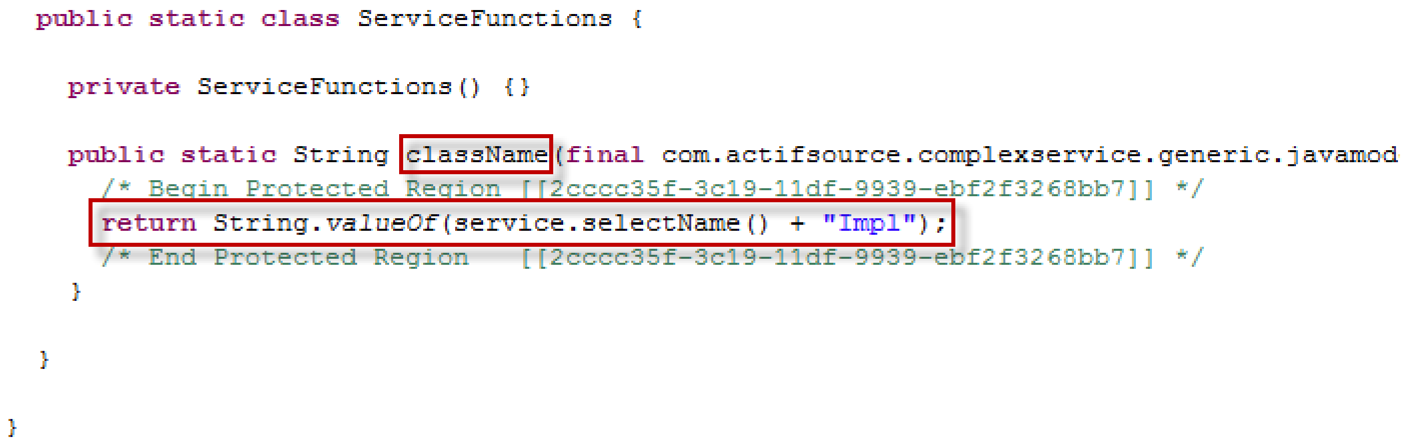
```
Service.className.java
class Service.className {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

}
```

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

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

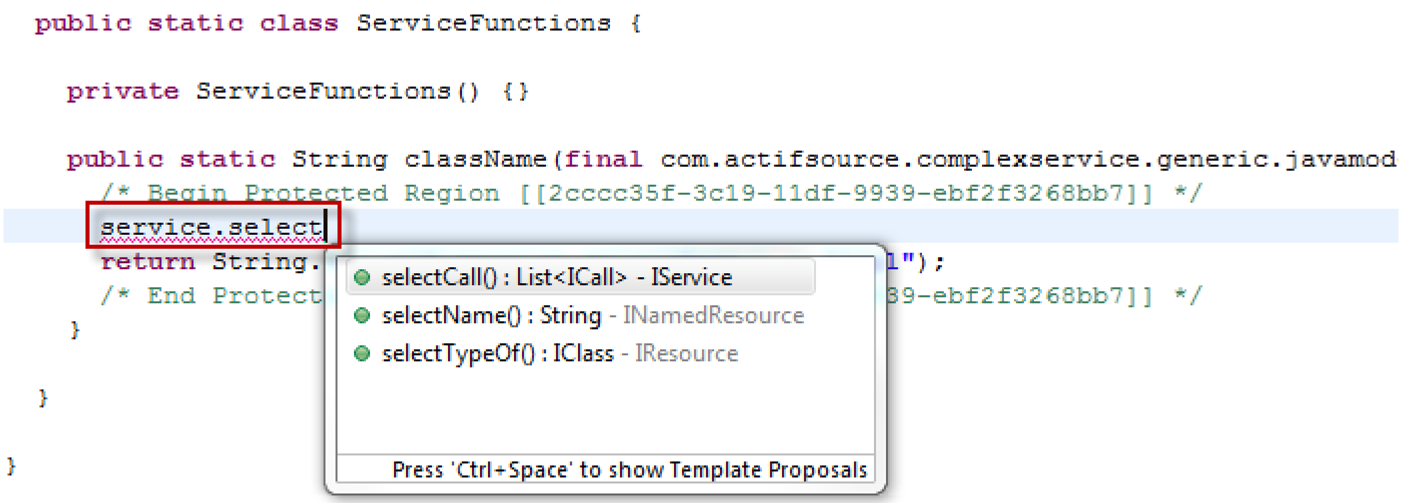


Editing Functions



```
  public static class ServiceFunctions {

    private ServiceFunctions() {}

    public static String className(final com.actifsource.complexservice.generic.javamod
      /* Begin Protected Region [[2cccc35f-3c19-11df-9939-ebf2f3268bb7]] */
      return String.valueOf(service.selectName() + "Impl");
      /* End Protected Region   [[2cccc35f-3c19-11df-9939-ebf2f3268bb7]] */
    }

  }

}
```

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

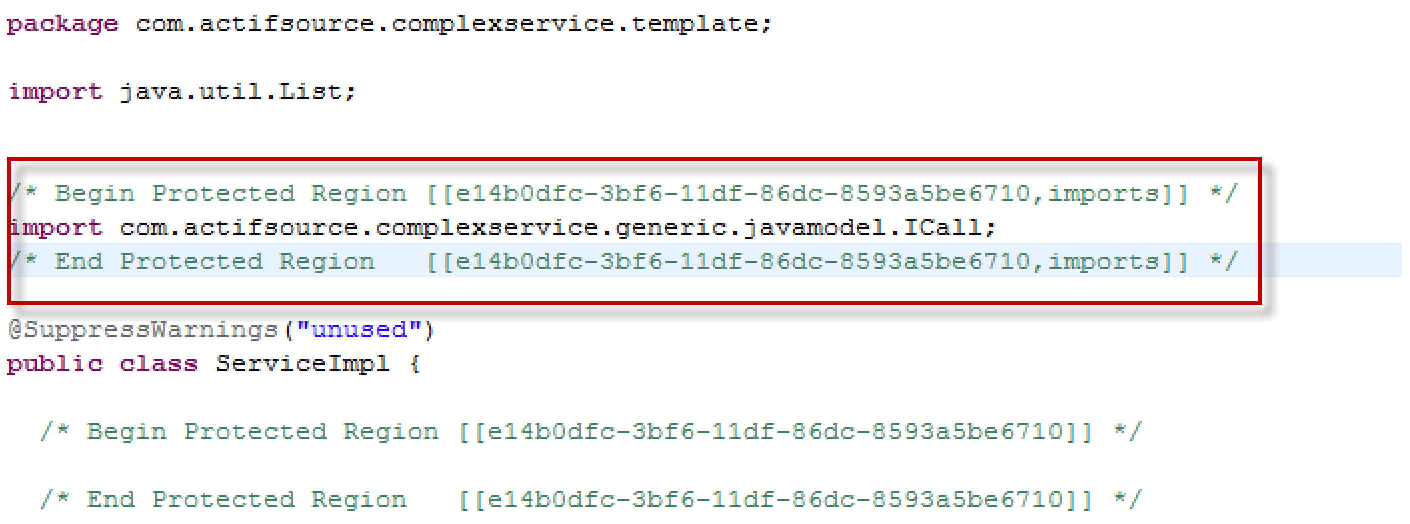


```
  public static class ServiceFunctions {

    private ServiceFunctions() {}

    public static String className(final com.actifsource.complexservice.generic.javamod
      /* Begin Protected Region [[2cccc35f-3c19-11df-9939-ebf2f3268bb7]] */
      service.select
      return String.                                                   1");
      /* End Protect                                                   39-ebf2f3268bb7]] */
    }

  }

}
```

```
selectCall() : List<ICall> - IService
selectName() : String - INamedResource
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**

© 2010 actifsource GmbH, Switzerland - all rights reserved.



```
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.

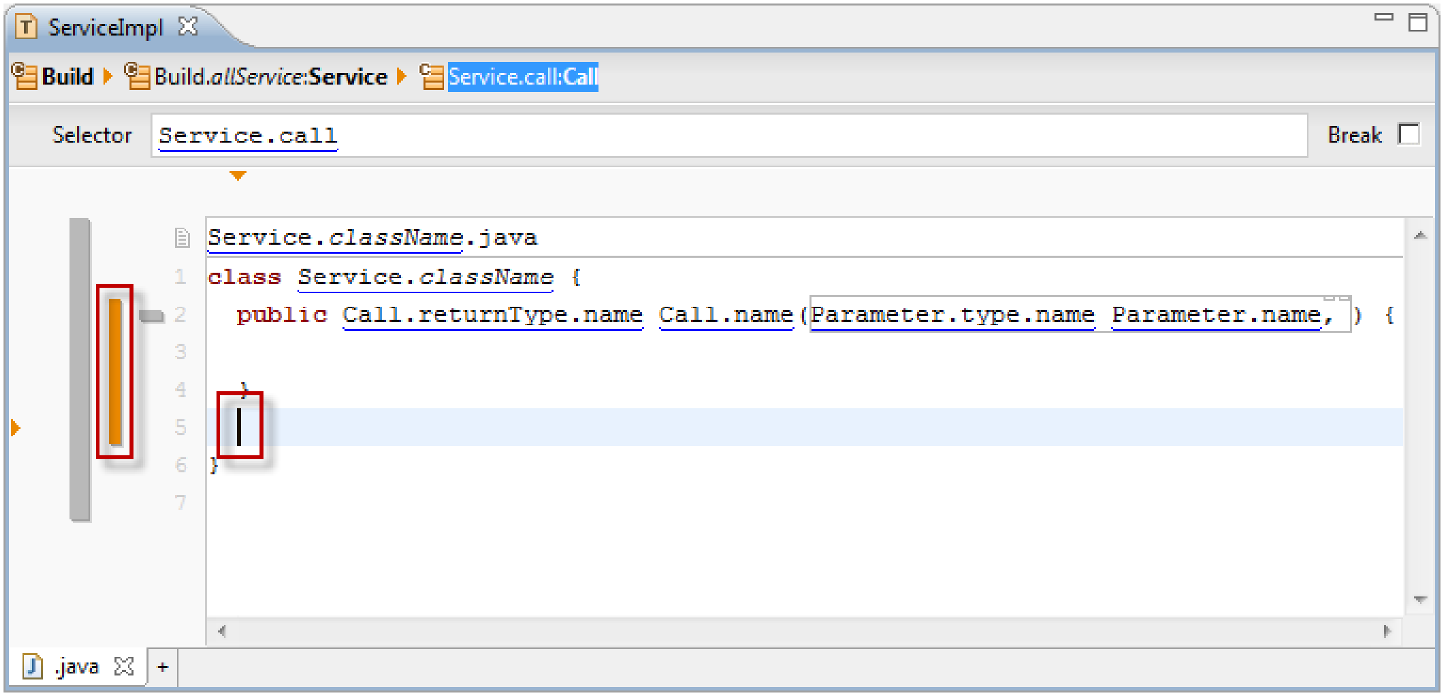
© 2010 actifsource GmbH, Switzerland - all rights reserved.

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**

© 2010 actifsource GmbH, Switzerland - all rights reserved.



Selector: `Service.call`

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

  }

}
```

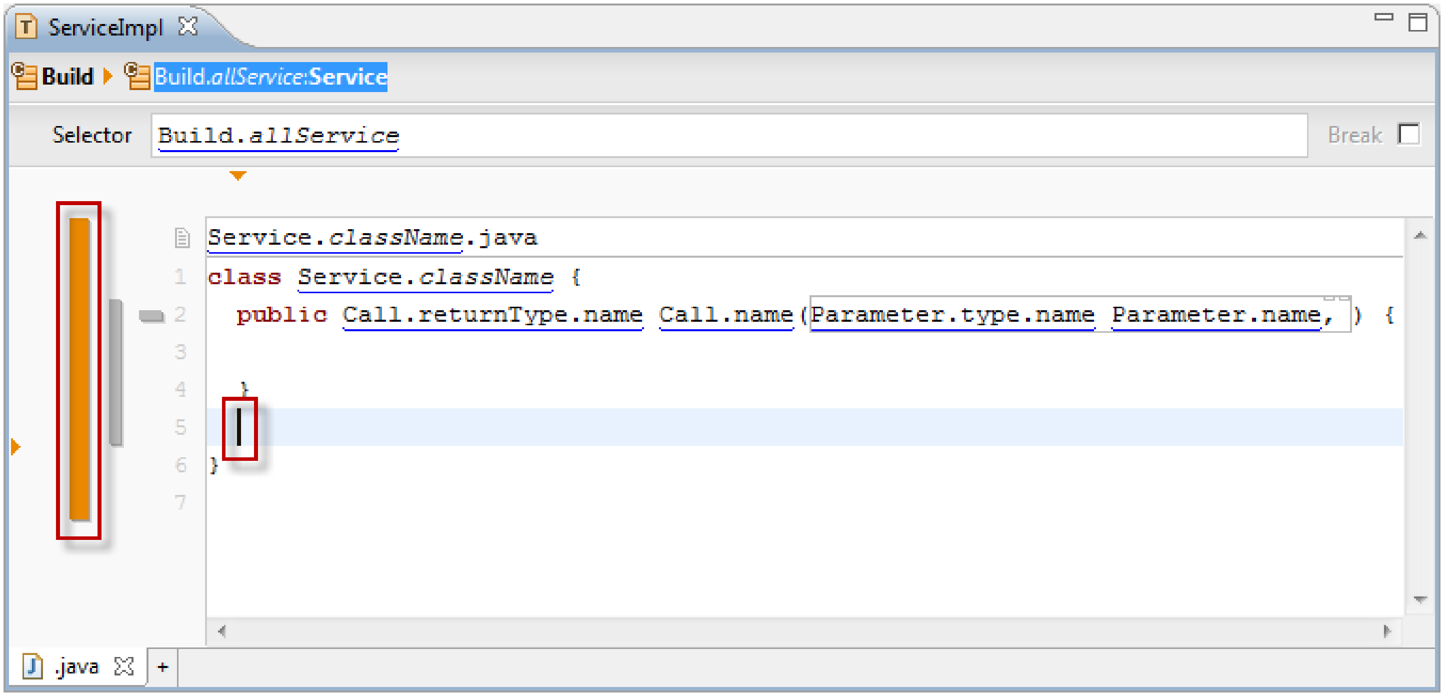
ⓘ Let's add a new line after the Call **Context** in the Service **Context**

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

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

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Advanced Context Operations



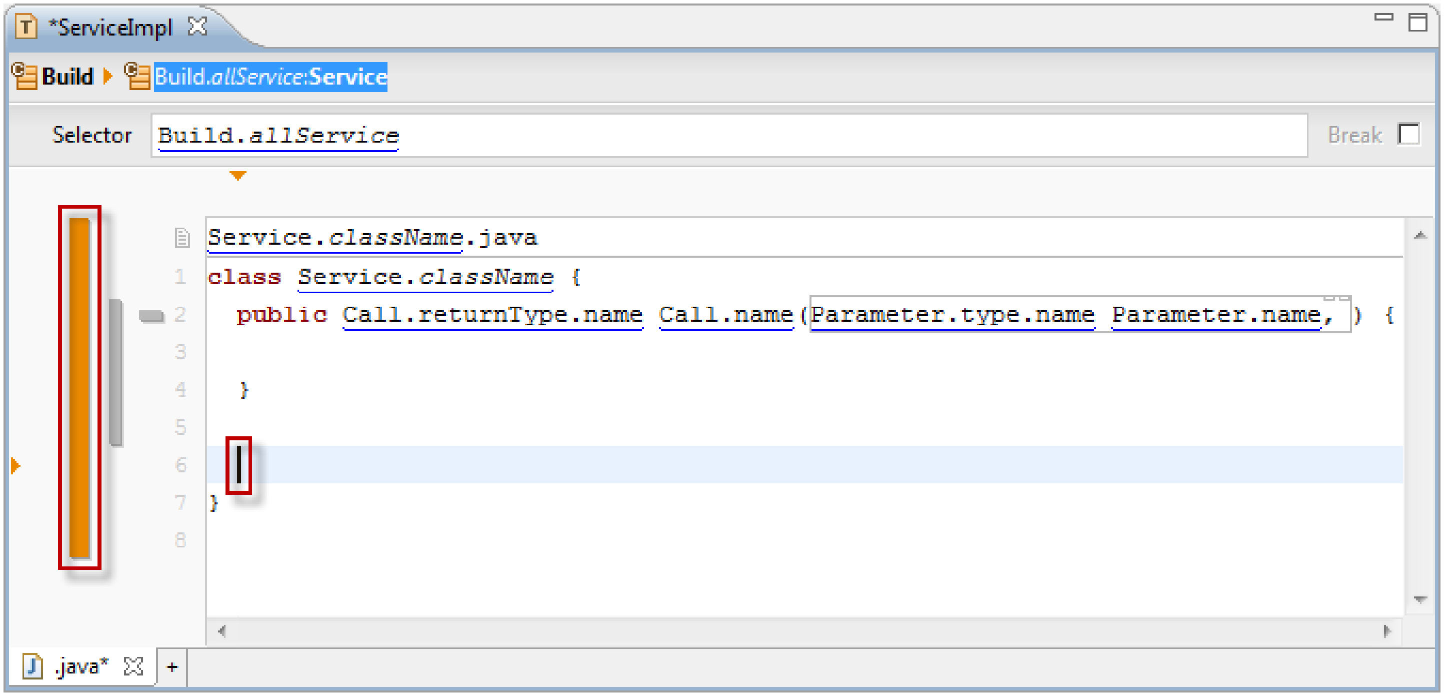
```
Build.allService
```

```
Service.className.java
class Service.className {
  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

© 2010 actifsource GmbH, Switzerland - all rights reserved.



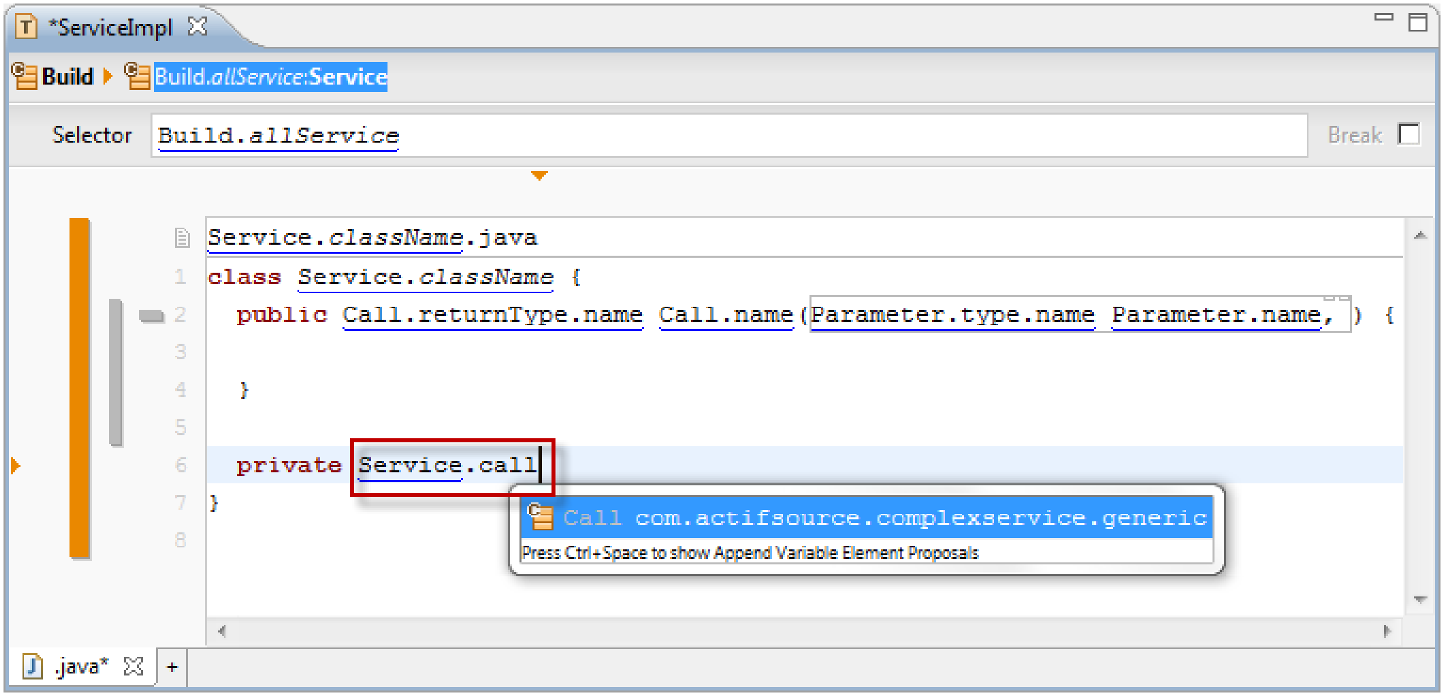
```
Build.allService
```

```
Service.className.java
class Service.className {
  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



```
Build.allService
```

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

  }

  private Service.call
}
```

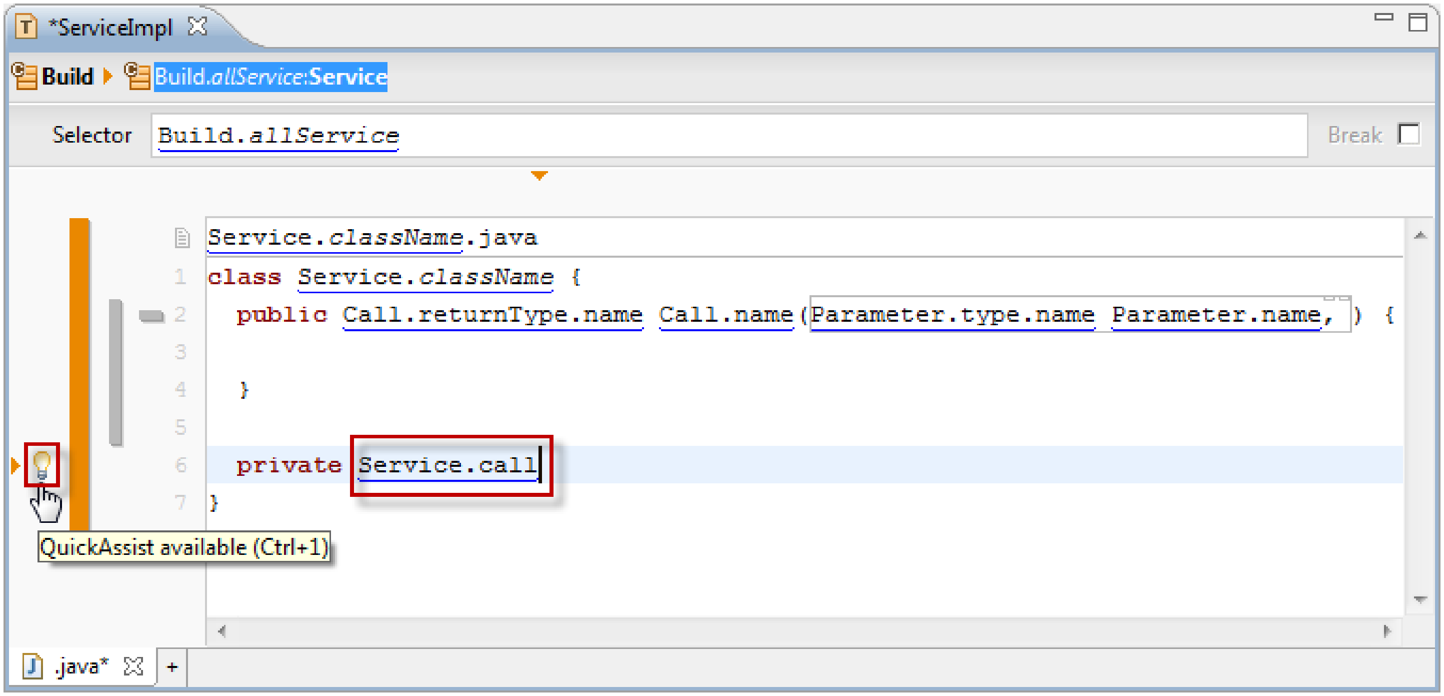
```
Call com.actifsource.complexservice.generic
Press Ctrl+Space to show Append Variable Element Proposals
```

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

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

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Add Context via Quick Assist



```
Build.allService
```

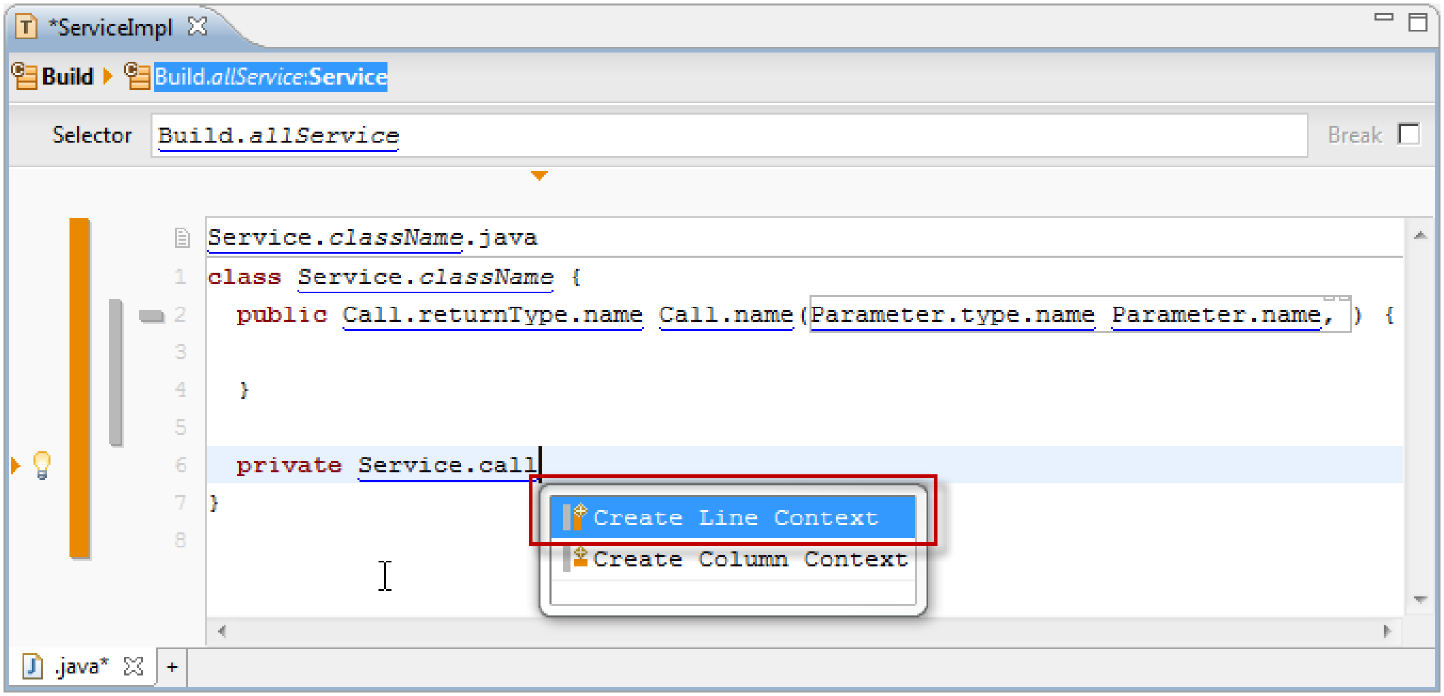
```
Service.className.java
class Service.className {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  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

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Add Context via Quick Assist



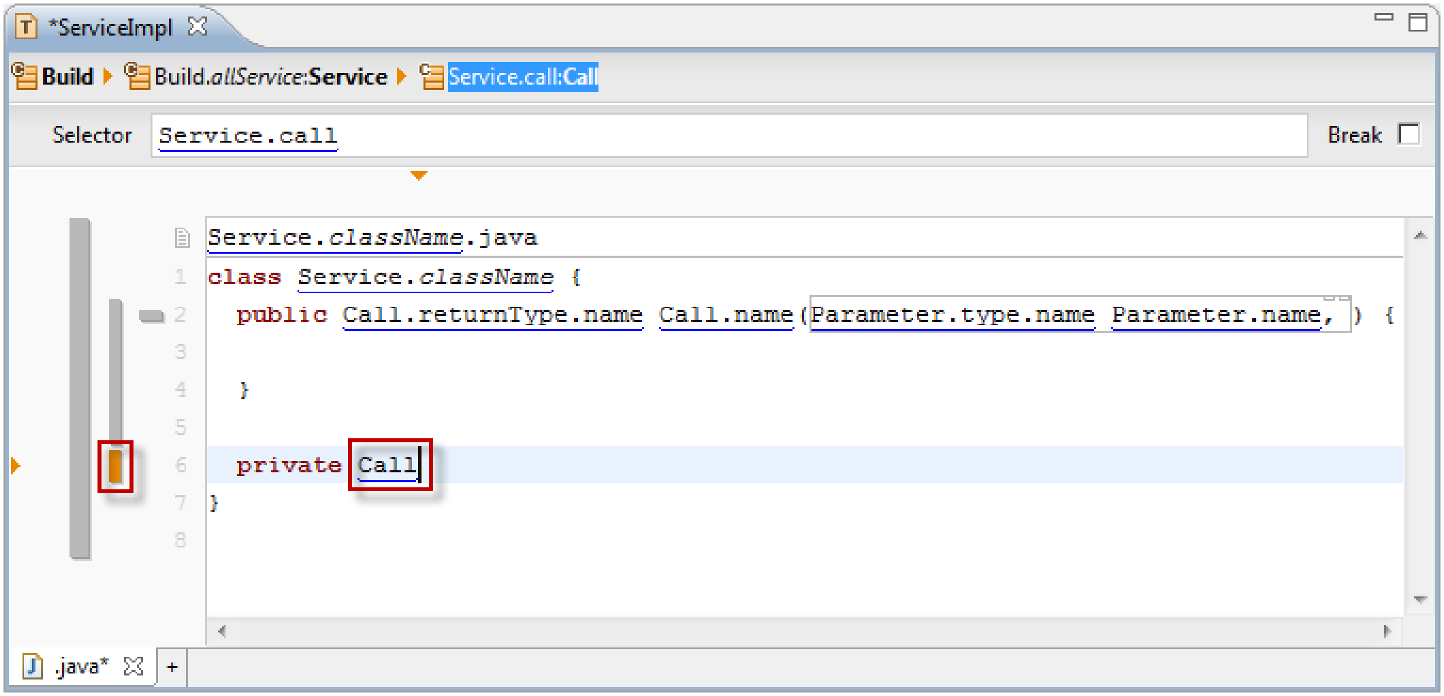
```
Build.allService
```

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

  }

  private Service.call
}
```

↳ Click on *Create Line Context*

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Add Context via Quick Assist



Selector: Service.call

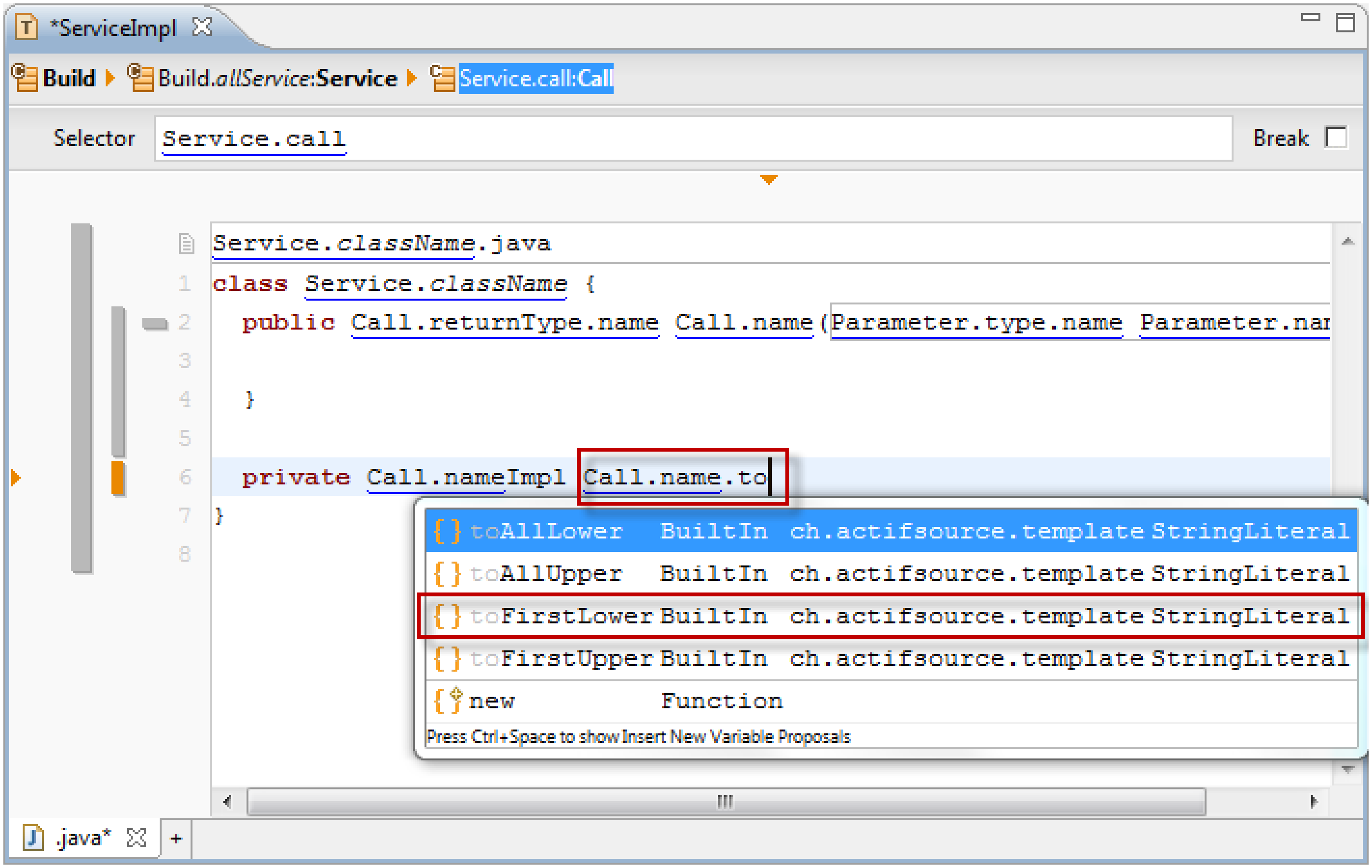
```
Service.className.java
class Service.className {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  private Call
}
```

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

ⓘ The Variable Service.call has been replaced by Call

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Built-In Functions


*ServiceImpl

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

Selector `Service.call` Break ☐

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

  }

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

```
{} toAllLower   BuiltIn  ch.actifsource.template StringLiteral
{} toAllUpper   BuiltIn  ch.actifsource.template StringLiteral
{} toFirstLower BuiltIn  ch.actifsource.template StringLiteral
{} toFirstUpper BuiltIn  ch.actifsource.template StringLiteral
{} new          Function
Press Ctrl+Space to show Insert New Variable Proposals
```

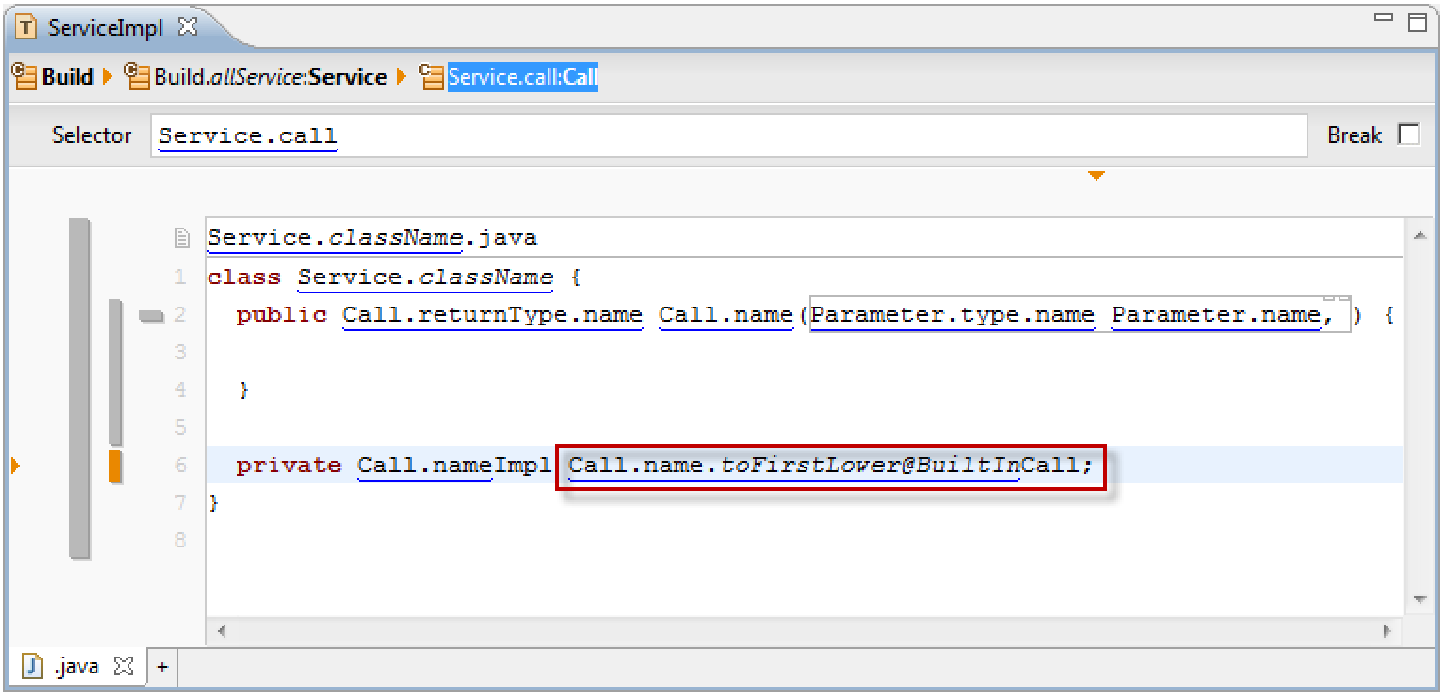
.java*

ⓘ Let’s use **Built-In Functions** on **Attributes**

Press '.' (dot) and **Content Assist** (Ctrl+Space) after <u>**name**</u> to see all available **Built-In Functions**

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Built-In Functions



ServiceImpl

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

Selector `Service.call` Break ☐

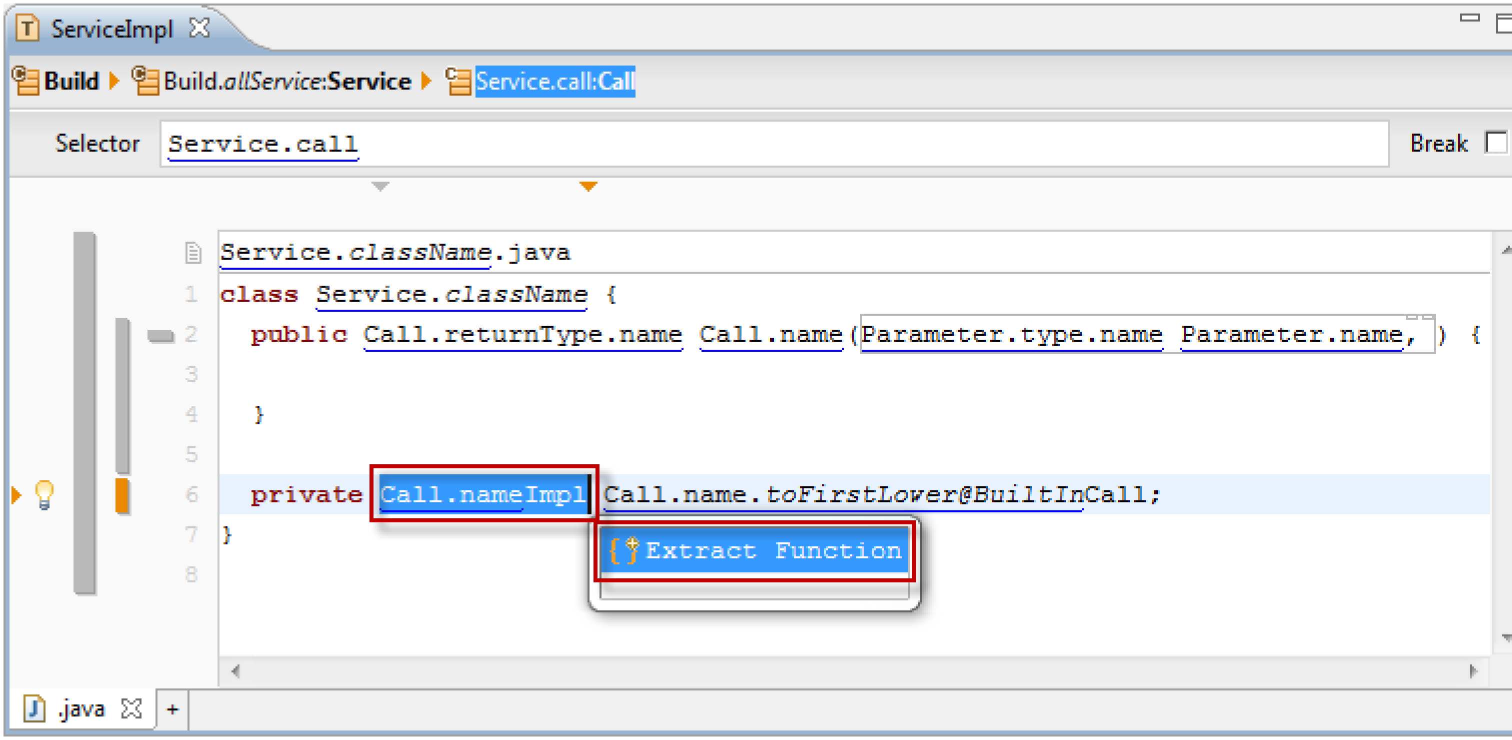
```
Service.className.java
class Service.className {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

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

.java +

↳ Complete the member variable declaration as shown above

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Extract Function in Function Space



Service.call

```
Service.className.java
class Service.className {
  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

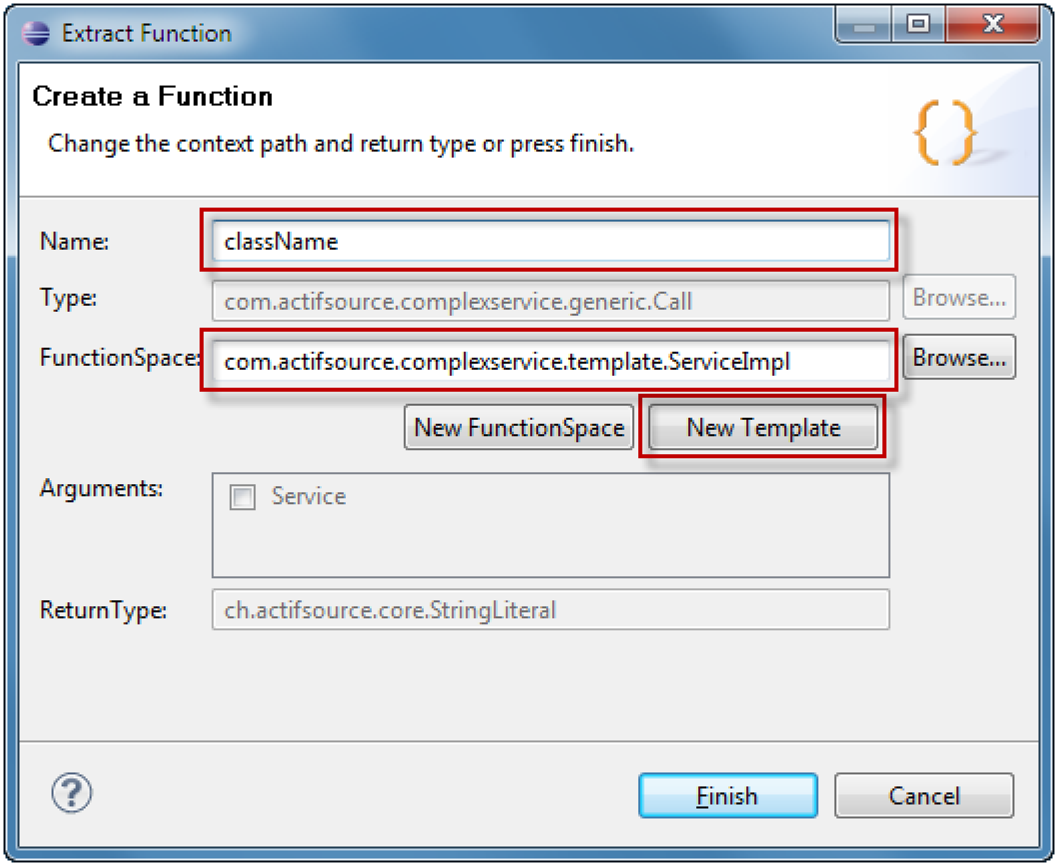
↳ Select the term Call.nameImpl

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

↳ Click *Extract Function*

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Extract Function in Function Space



Create a Function

Change the context path and return type or press finish.

Name: className

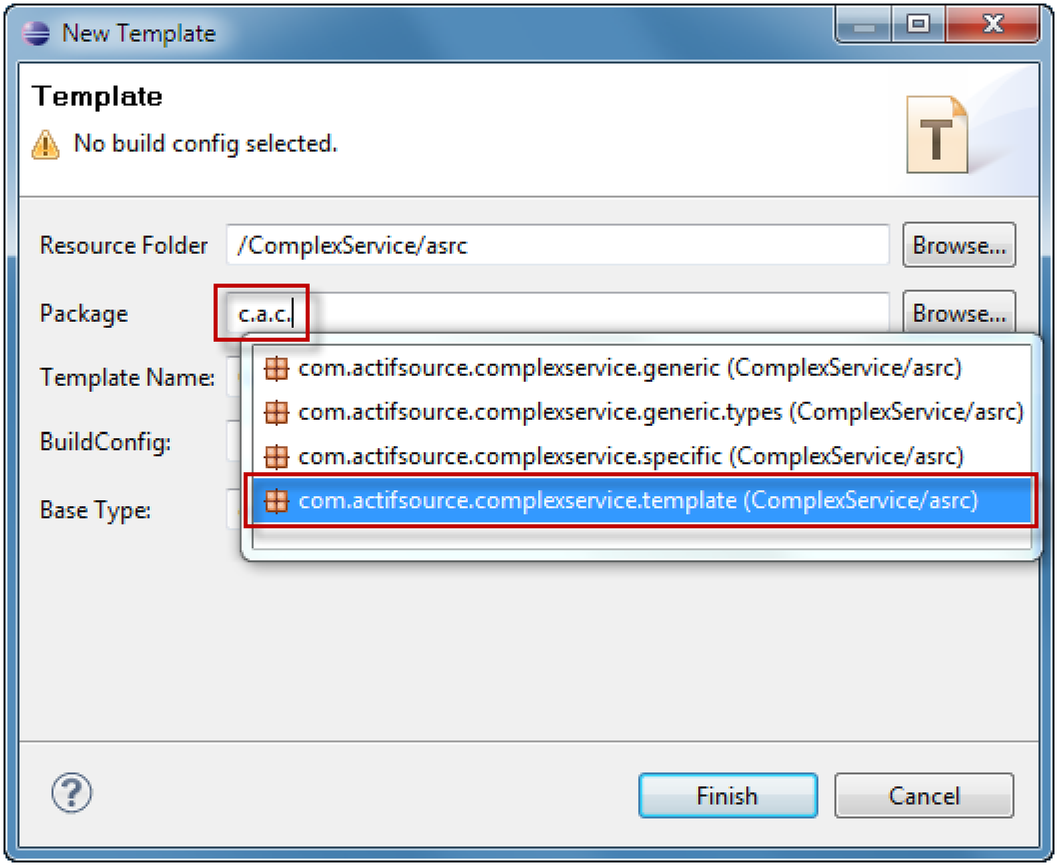
Type: com.actifsource.complexservice.generic.Call

FunctionSpace: com.actifsource.complexservice.template.ServiceImpl

Arguments: ☐ Service

ReturnType: ch.actifsource.core.StringLiteral

- 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*



Resource Folder /ComplexService/asrc

Package c.a.c.

Template Name:

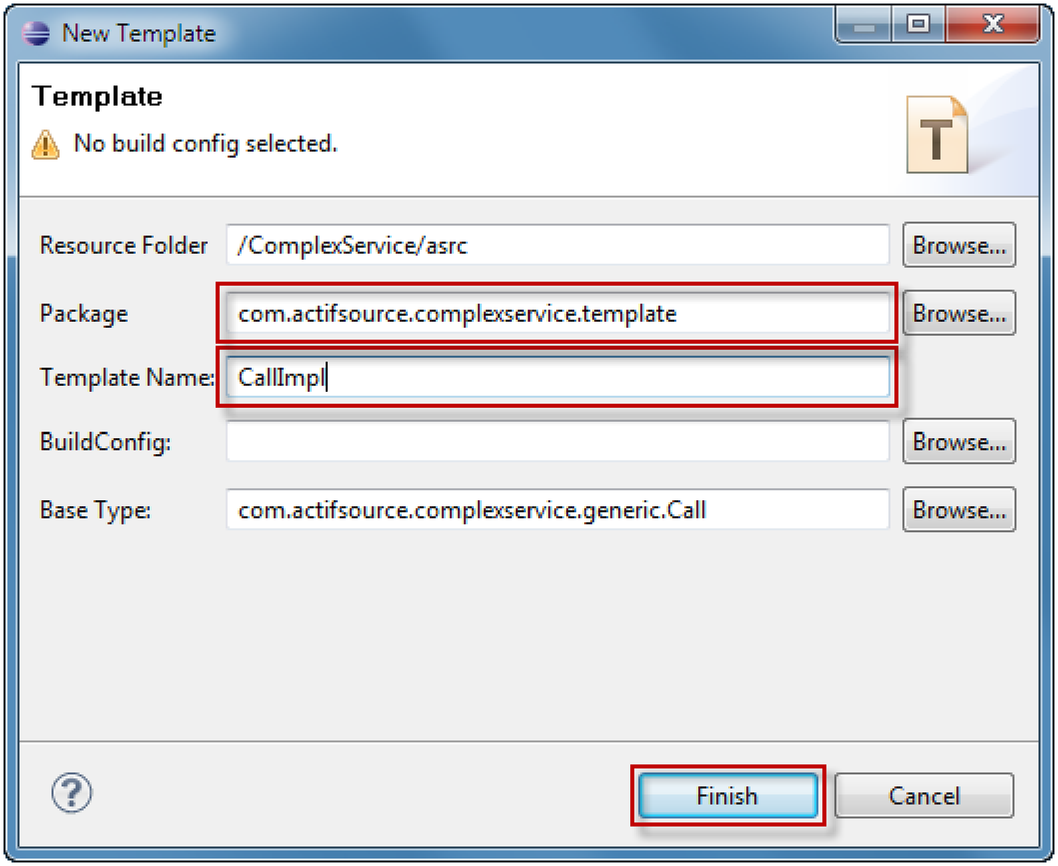
BuildConfig:

Base Type:

com.actifsource.complexservice.generic (ComplexService/asrc)
com.actifsource.complexservice.generic.types (ComplexService/asrc)
com.actifsource.complexservice.specific (ComplexService/asrc)
com.actifsource.complexservice.template (ComplexService/asrc)

↳ Select the **Package** for the new **Template** as shown above using **Content Assist** (Ctrl+Space) or the *Browse…* button

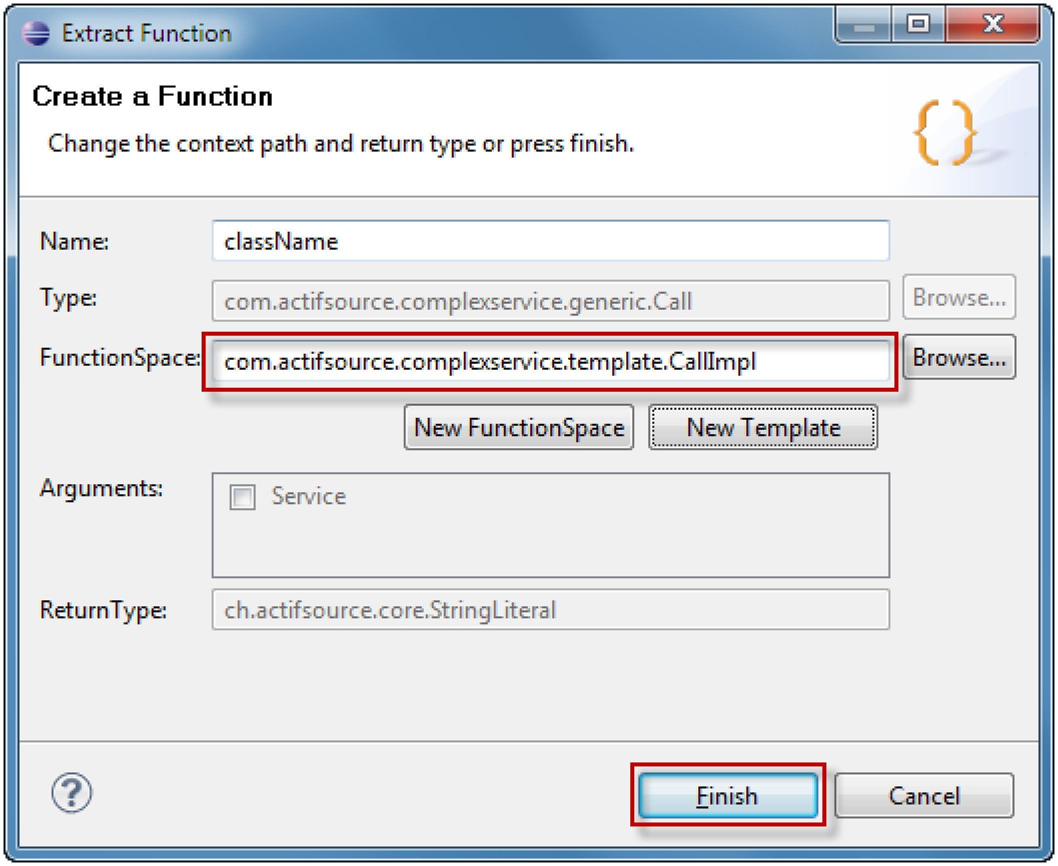
© 2010 actifsource GmbH, Switzerland - all rights reserved.



Resource Folder: /ComplexService/asrc
Package: com.actifsource.complexservice.template
Template Name: CallImpl
BuildConfig:
Base Type: com.actifsource.complexservice.generic.Call

- ⓘ Check the **Package**
- ↳ Name the **Template** `CallImpl`
- ↳ Press *Finish*

© 2010 actifsource GmbH, Switzerland - all rights reserved.



Create a Function

Change the context path and return type or press finish.

Name: className

Type: com.actifsource.complexservice.generic.Call

FunctionSpace: com.actifsource.complexservice.template.CallImpl

Arguments: ☐ Service

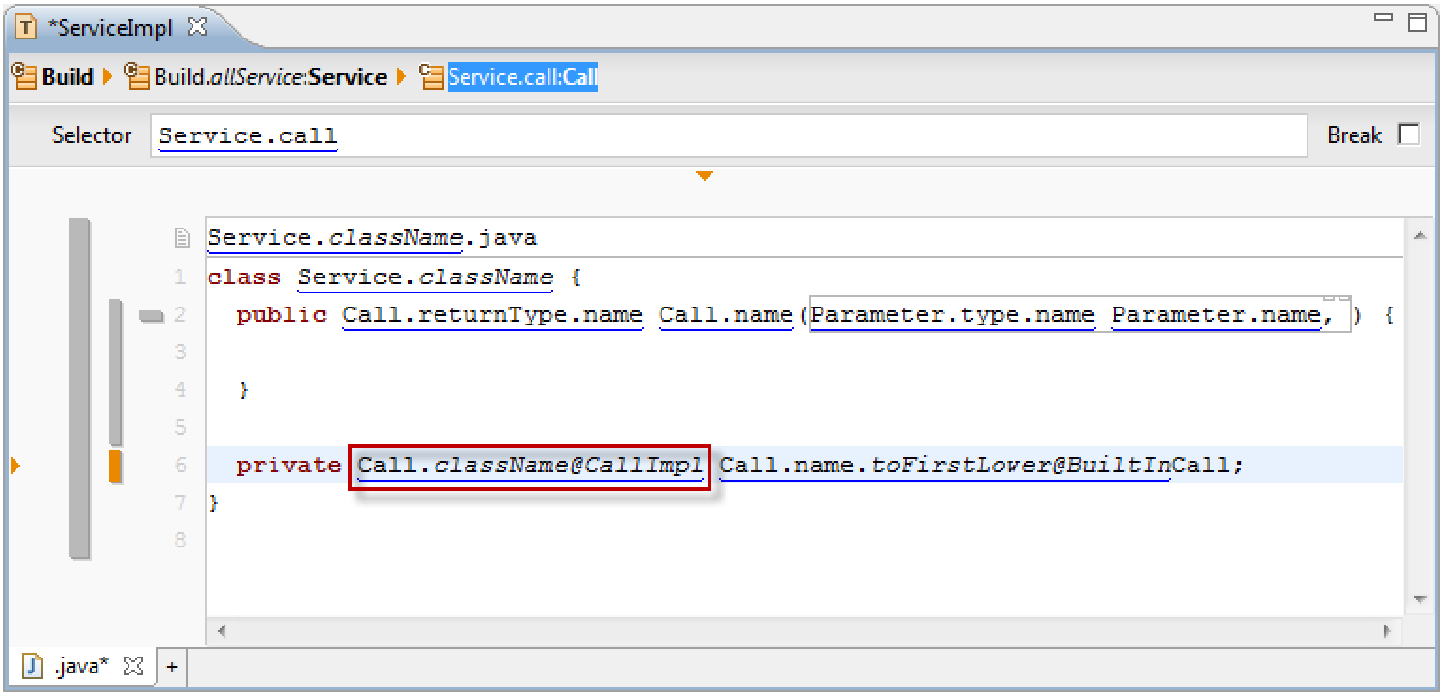
ReturnType: ch.actifsource.core.StringLiteral

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

↳ Press *Finish*

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Extract Function in Function Space



Selector: Service.call

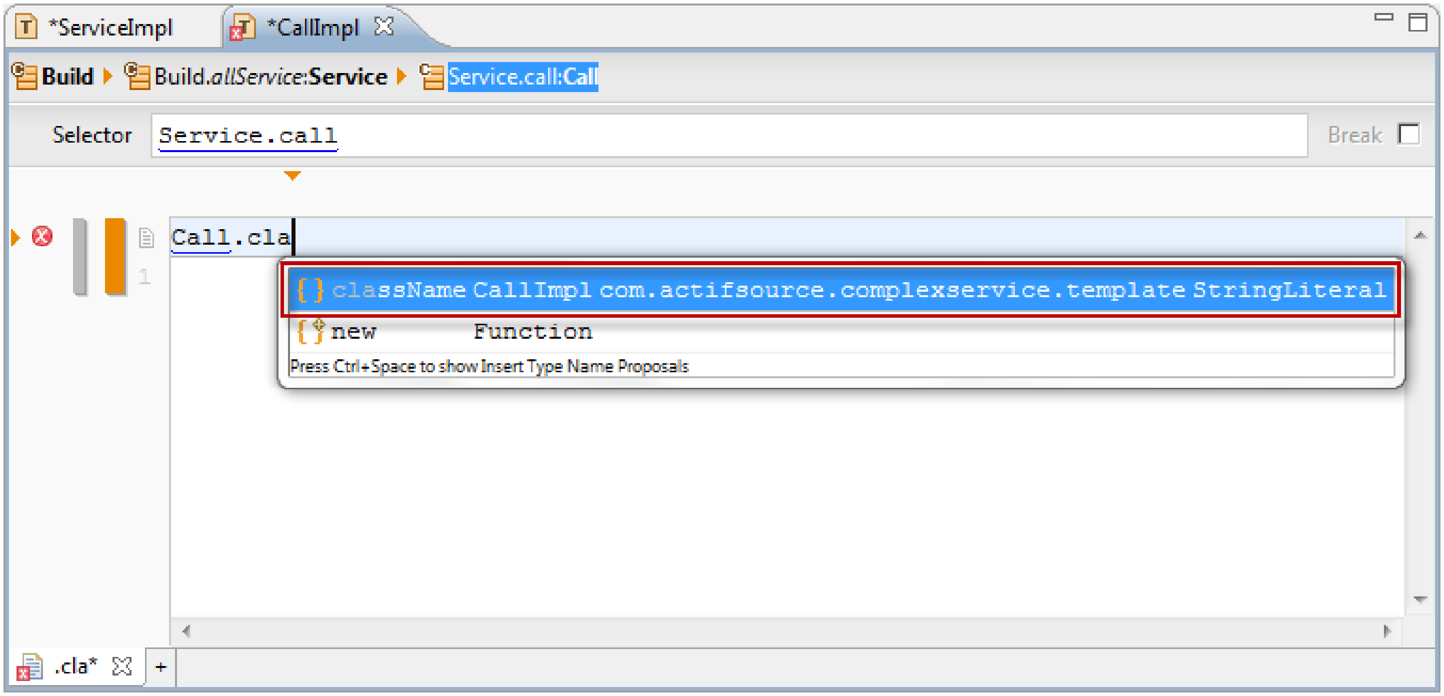
```
Service.className.java
class Service.className {
  public Call.returnType.name Call.name(Parameter.type.name Parameter.name, ) {

  }

  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

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Extract Function in Function Space



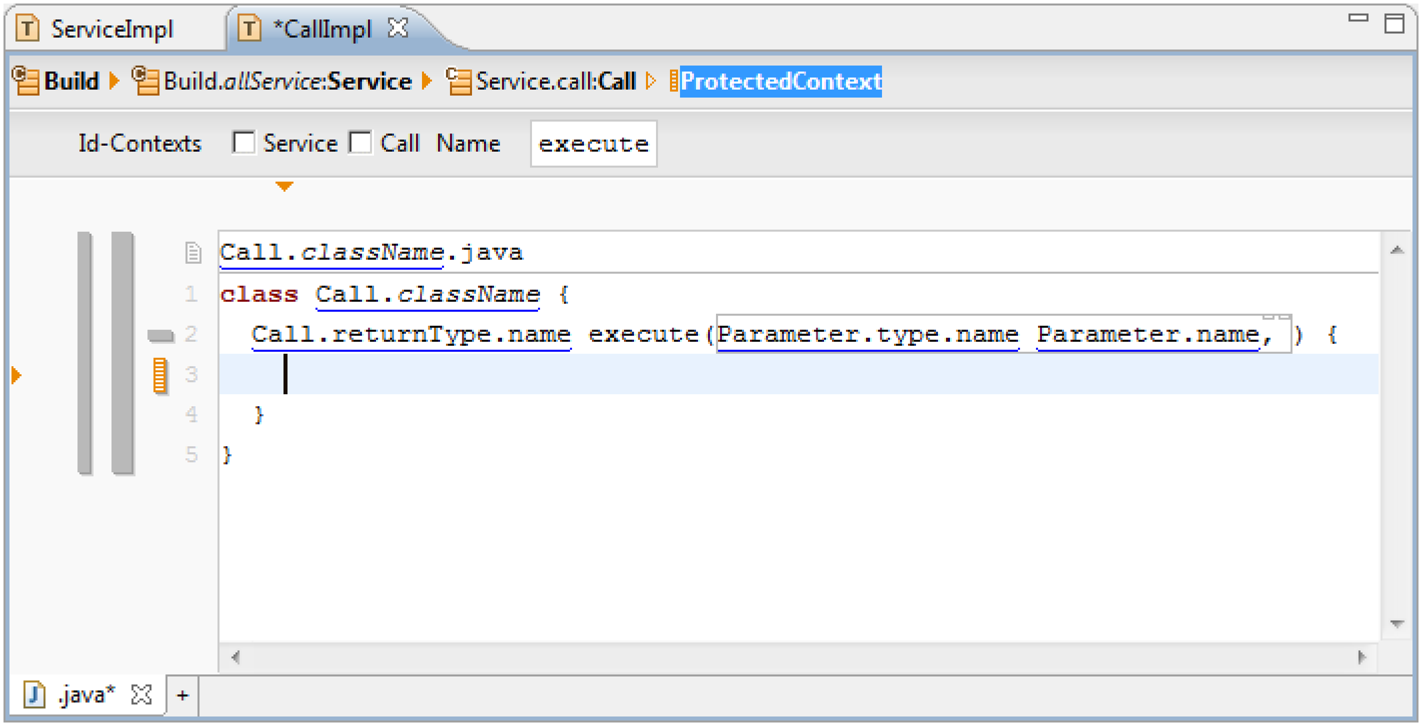
```
Service.call
Call.cla
{ } className CallImpl com.actifsource.complexservice.template StringLiteral
{ } new        Function
Press Ctrl+Space to show Insert Type Name Proposals
```

ⓘ 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

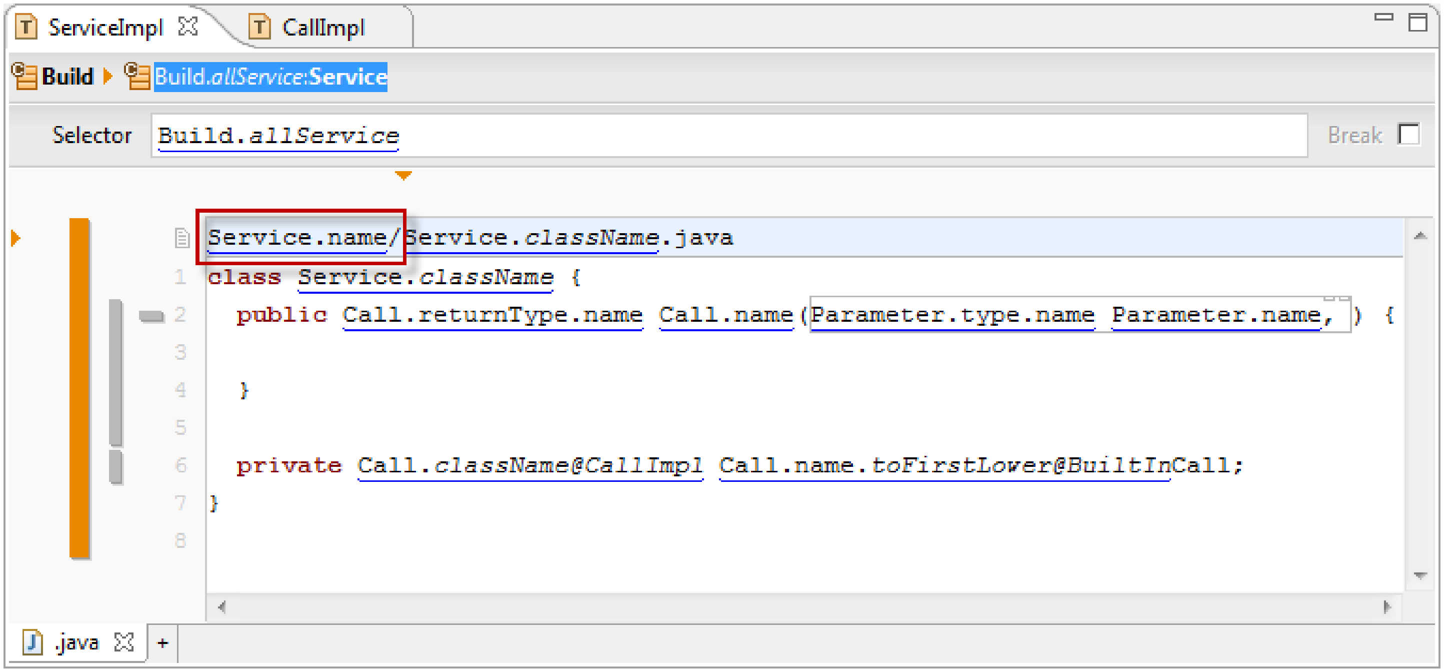


execute

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

- Write a simple class as shown above
- Write a method *execute* with returnType and Parameter
- Place a **Protected Context** in the function body

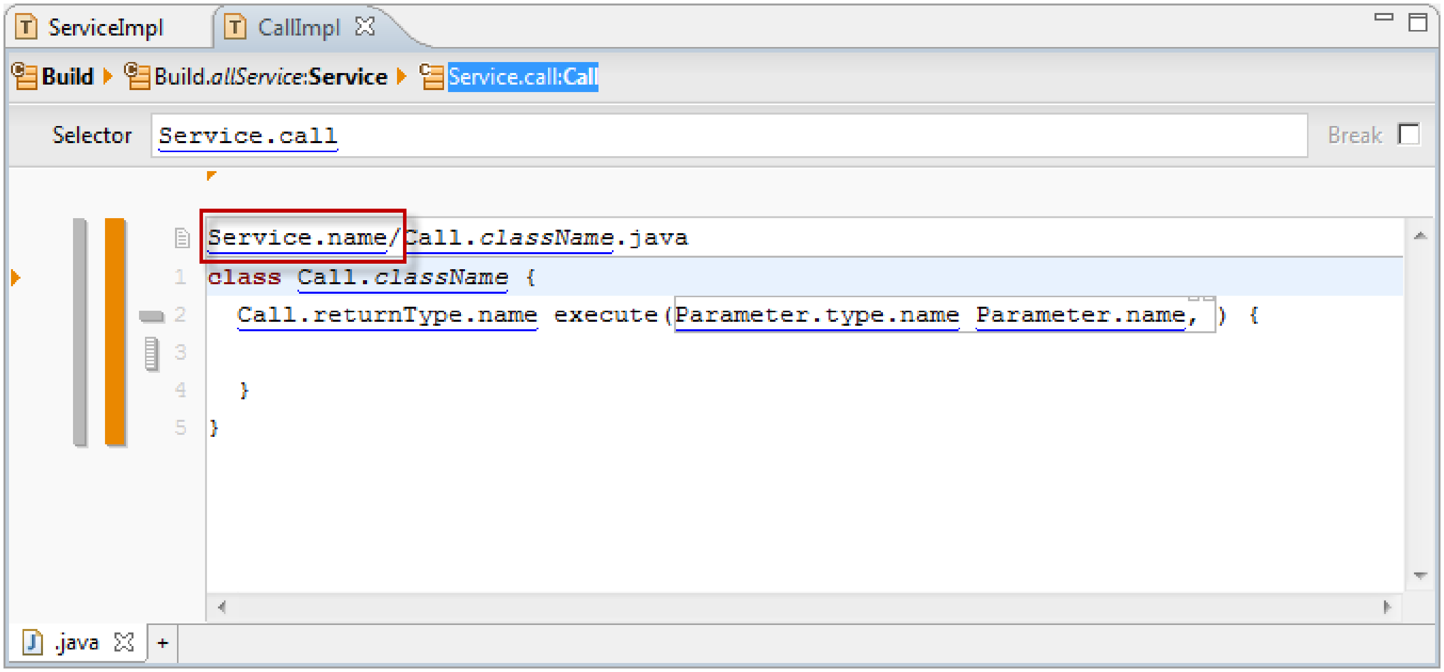
Templates and Folders



```
Build.allService
```

```
Service.name/Service.className.java
class Service.className {
  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



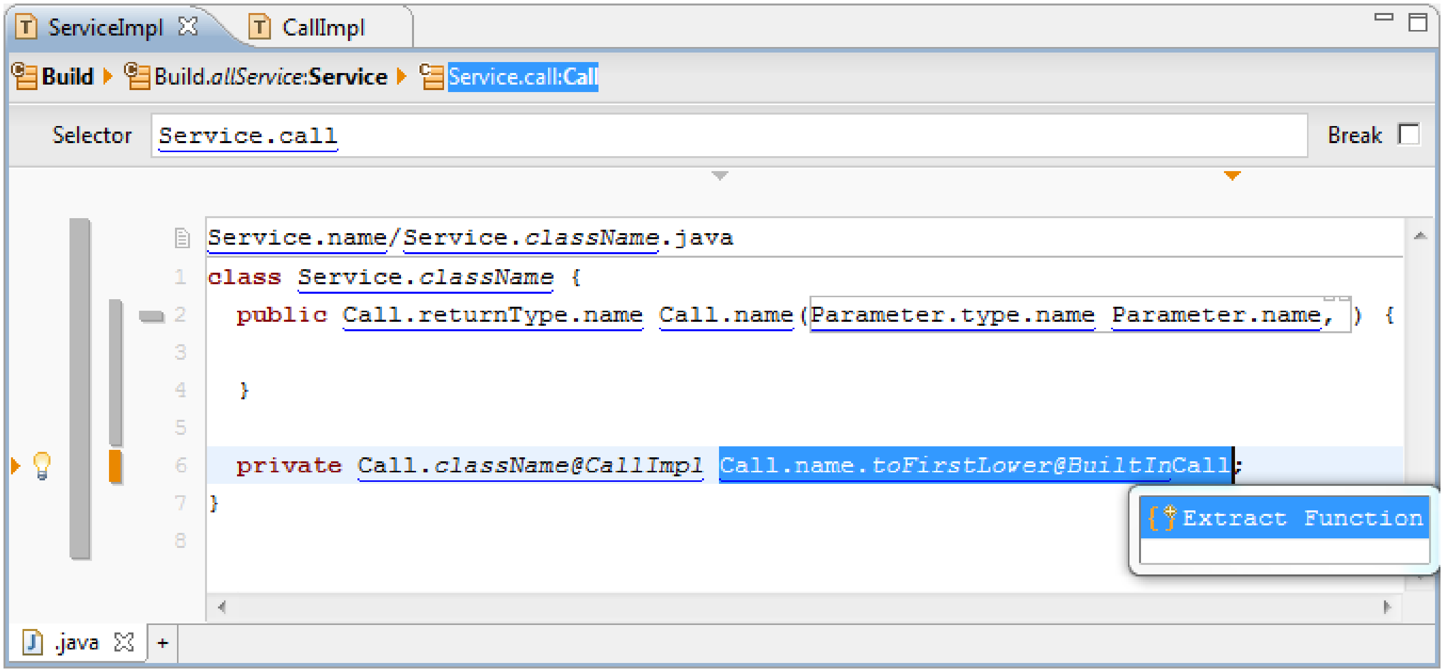
Selector: `Service.call`

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

  }
}
```

- ⓘ 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

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Copy with Context



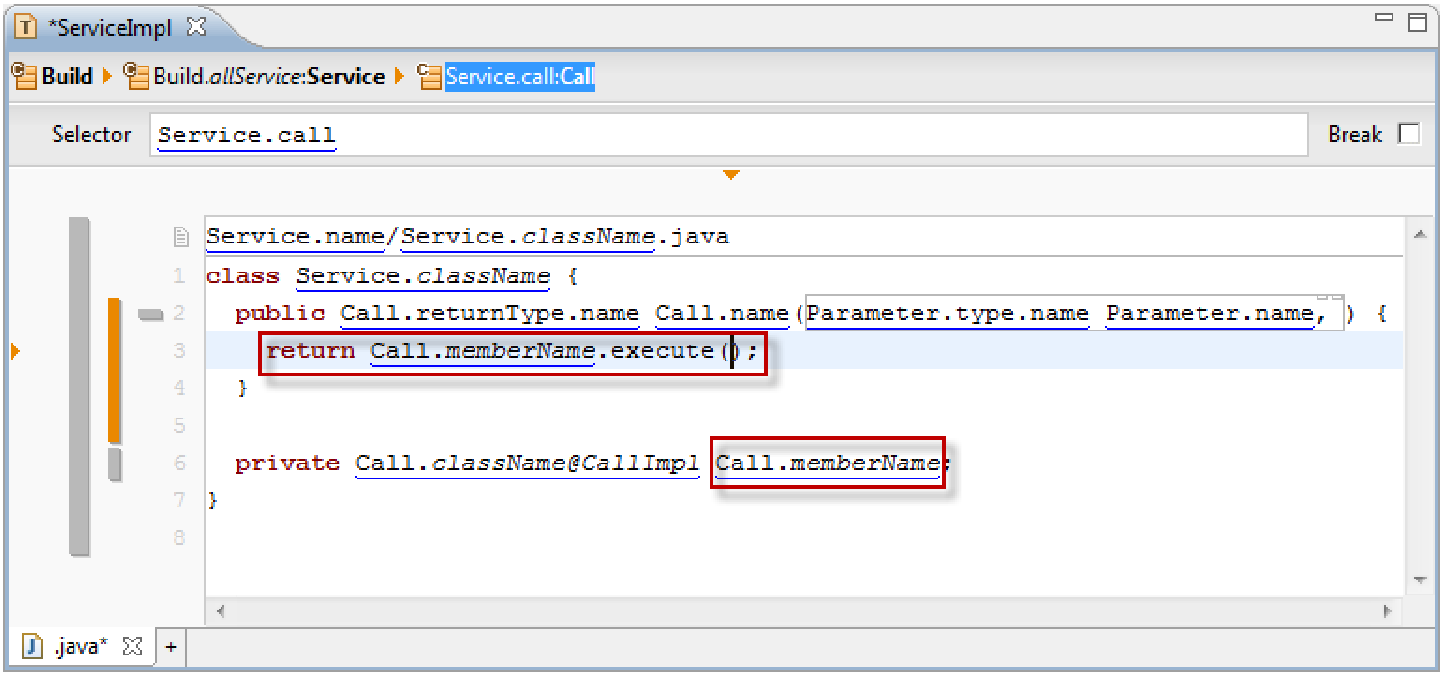
```
Service.call
```

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

  }

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

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

© 2010 actifsource GmbH, Switzerland - all rights reserved.

Copy with Context

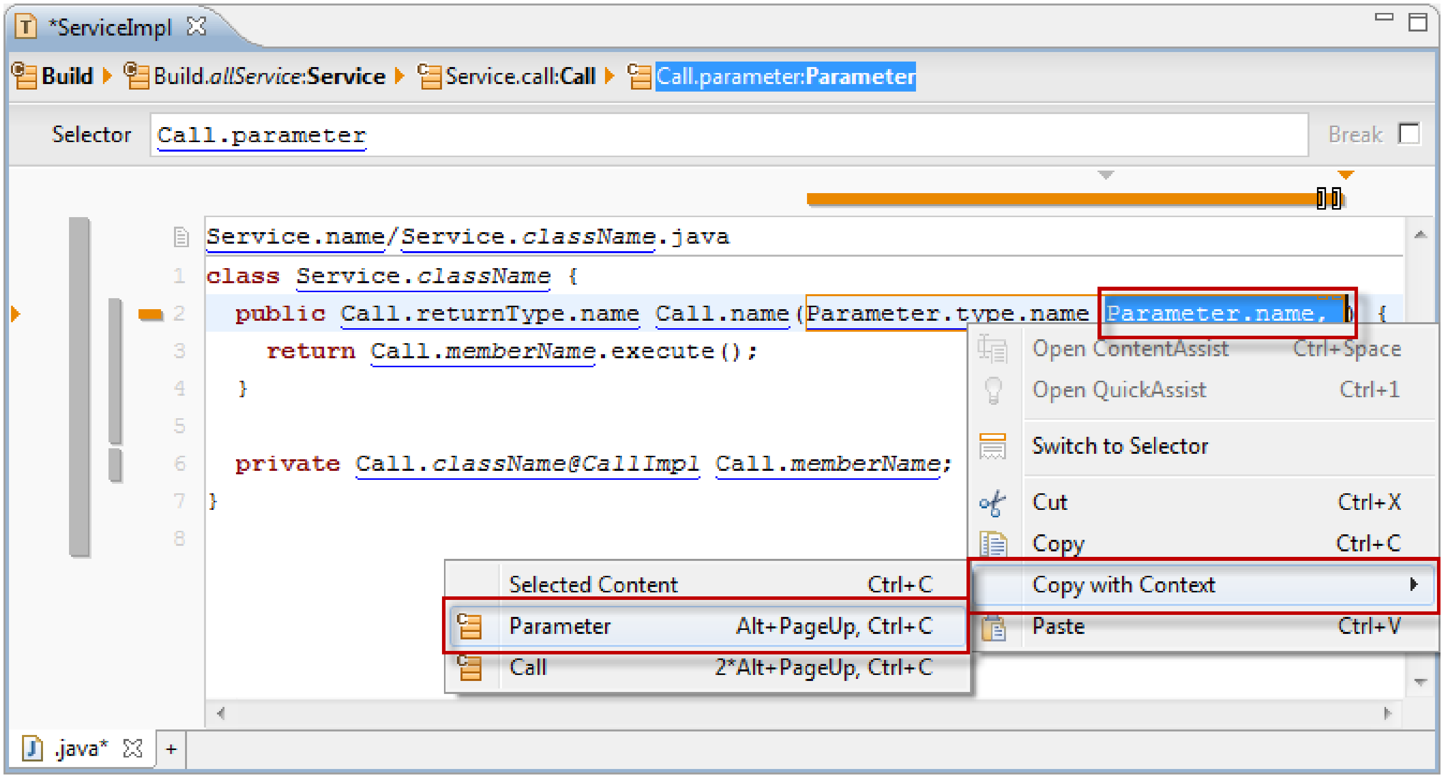


Selector: Service.call

```
Service.name/Service.className.java
class Service.className {
  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

Copy with Context



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

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

*ServiceImpl
Build ▸ Build.allService:Service ▸ Service.call:Call ▸ Call.parameter:Parameter
Selector Call.parameter
Break
Open ContentAssist Ctrl+Space
Open QuickAssist Ctrl+1
Switch to Selector
Cut Ctrl+X
Copy Ctrl+C
Copy with Context
Paste Ctrl+V
Selected Content Ctrl+C
Parameter Alt+PageUp, Ctrl+C
Call 2*Alt+PageUp, Ctrl+C
.java*

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

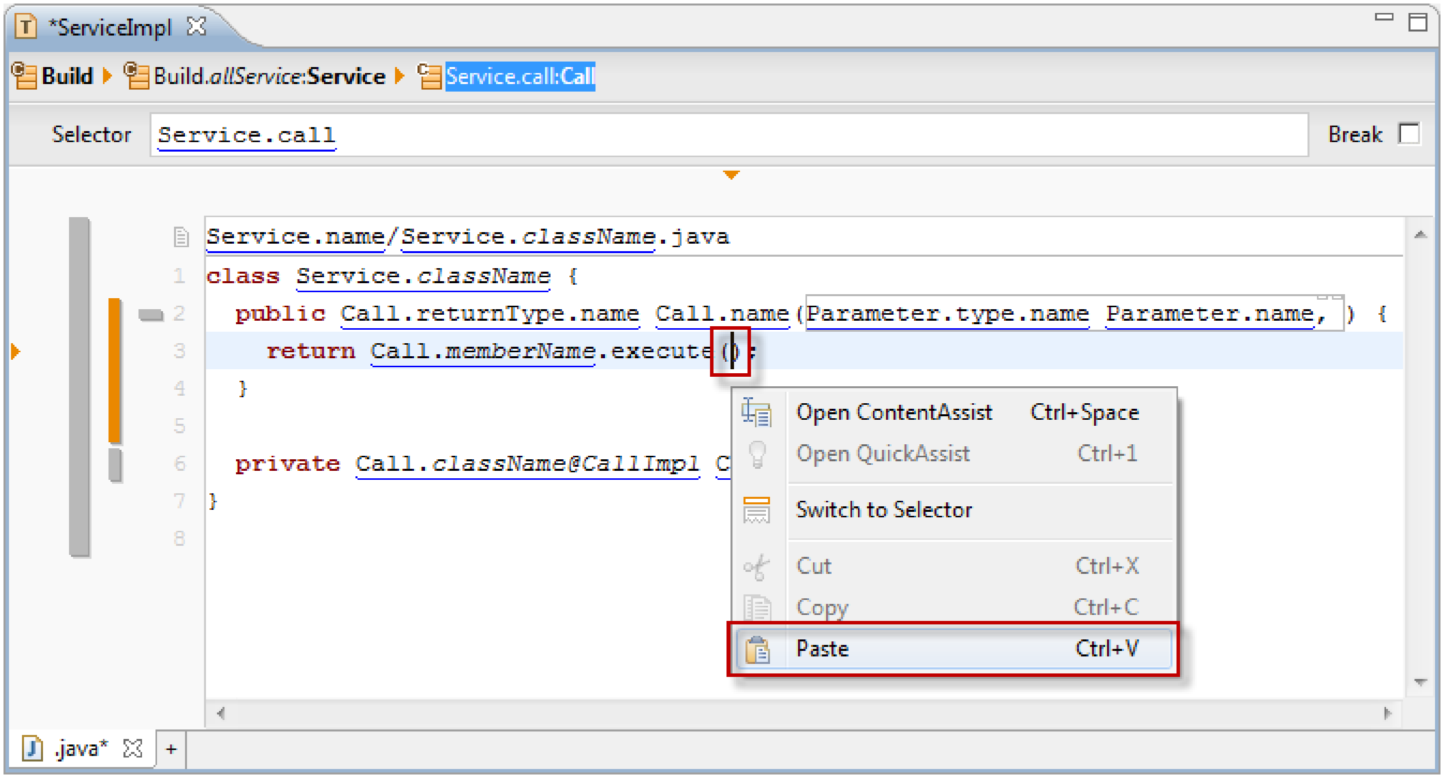
↳ Select the Term "`Parameter.name, `"

↳ 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

© 2010 actifsource GmbH, Switzerland - all rights reserved.

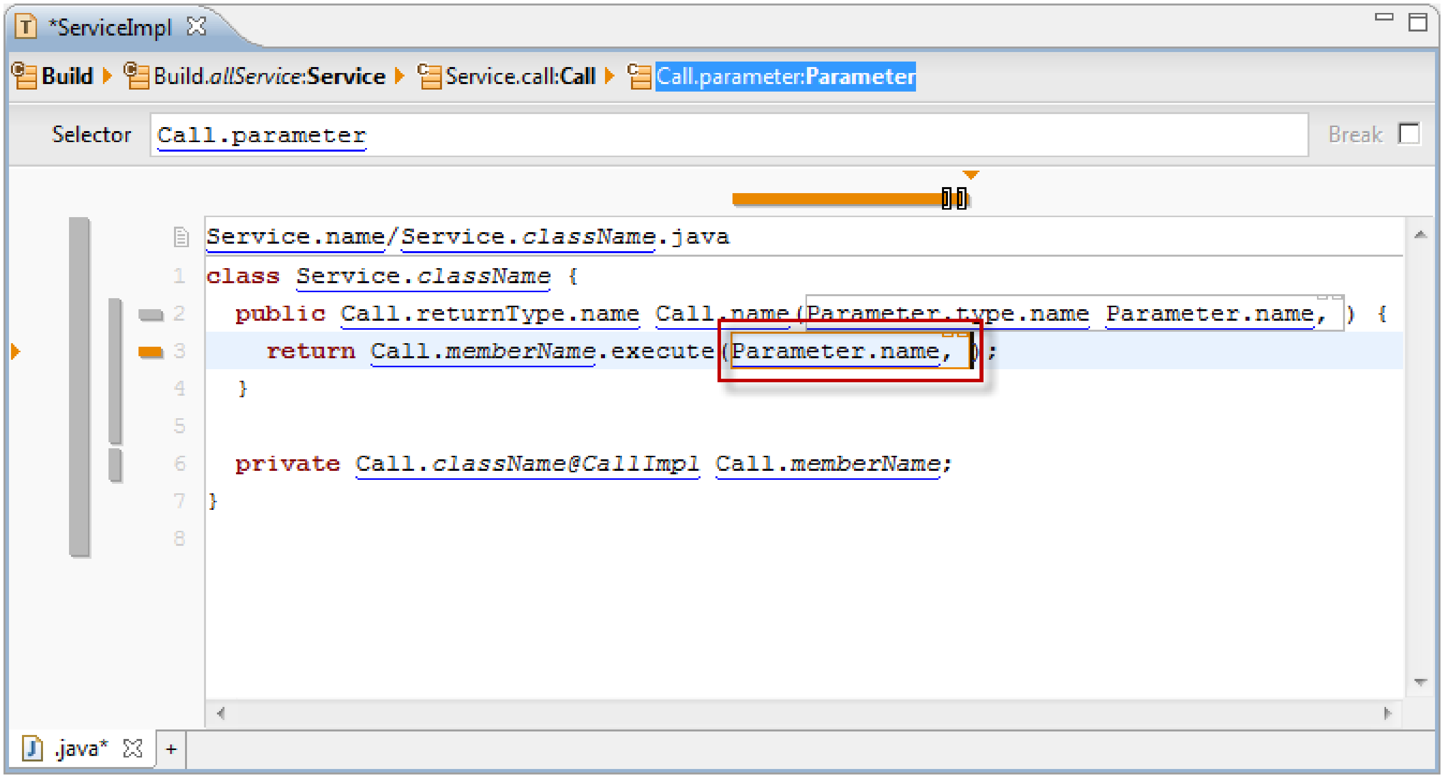


Selector: Service.call

```
Service.name/Service.className.java
class Service.className {
  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 {
  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

© 2010 actifsource GmbH, Switzerland - all rights reserved.

© 2010 actifsource GmbH, Switzerland - all rights reserved.