Current state of Bean Support

Last update 5-JUL-2017

Overview

Though we support many in-route elements within Camel routes in the graphical editor, we don't support many of the global configuration options that appear out-of-route yet. This support is gradually gaining functionality as we add new capabilities to the Configurations tab.

In order to better support mixed XML- and Java-configured Camel configurations and up-and-coming Camel components such as type-safe transformers, we need Bean support to exist in the editor for both Spring- and Blueprint-based configurations.

Basic Case

At its core, essentially a “bean” reference is a pointer to a Java bean class that is given a unique ID. This seems to be the most common use of Java beans defined in XML.

Examples

Here’s one from Blueprint:

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
   <bean id="accountOne" class="org.apache.aries.simple.Account" />
</blueprint>
```

And another from Blueprint, specifying a constructor argument.:

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
   <bean id="accountOne" class="org.apache.aries.simple.Account">
     <argument value="1"/>
   </bean>
</blueprint>
```

And another for Blueprint, specifying both a constructor argument and a property.

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  <bean id="accountOne" class="org.apache.aries.simple.Account">
       <argument value="1"/>
       <property name="description" value="#1 account"/>
   </bean>
</blueprint>
```

Here’s one for Spring:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
           http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
           http://camel.apache.org/schema/spring
http://camel.apache.org/schema/spring/camel-spring.xsd">
```

```
  <camelContext id="camel5"
xmlns="http://camel.apache.org/schema/spring">
      <routeBuilder ref="myBuilder" />
  </camelContext>
  <bean id="myBuilder"
class="org.apache.camel.spring.example.test1.MyRouteBuilder"/>
</beans>
```

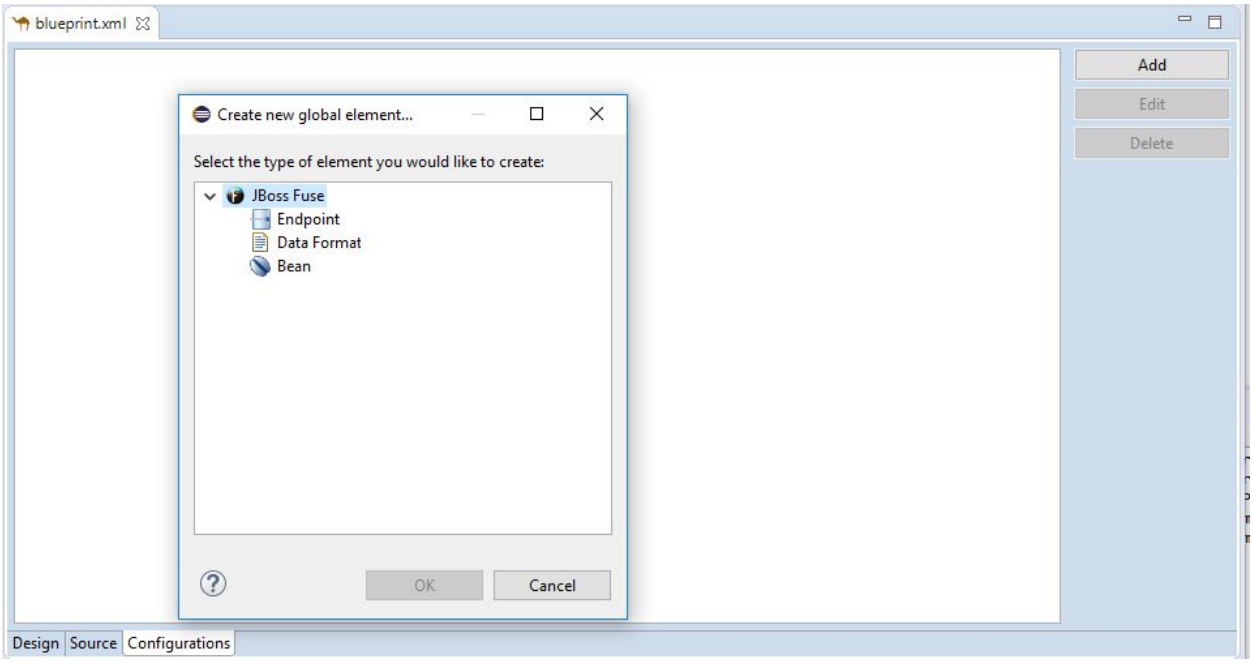
There are a few other cases I've called out in this document: https://docs.google.com/a/redhat.com/document/d/1oPoeNF1Qilbo7yBiVYATI4_v1ca0qAENY5puv5kQcnQ/edit?usp=sharing

Functionality Included

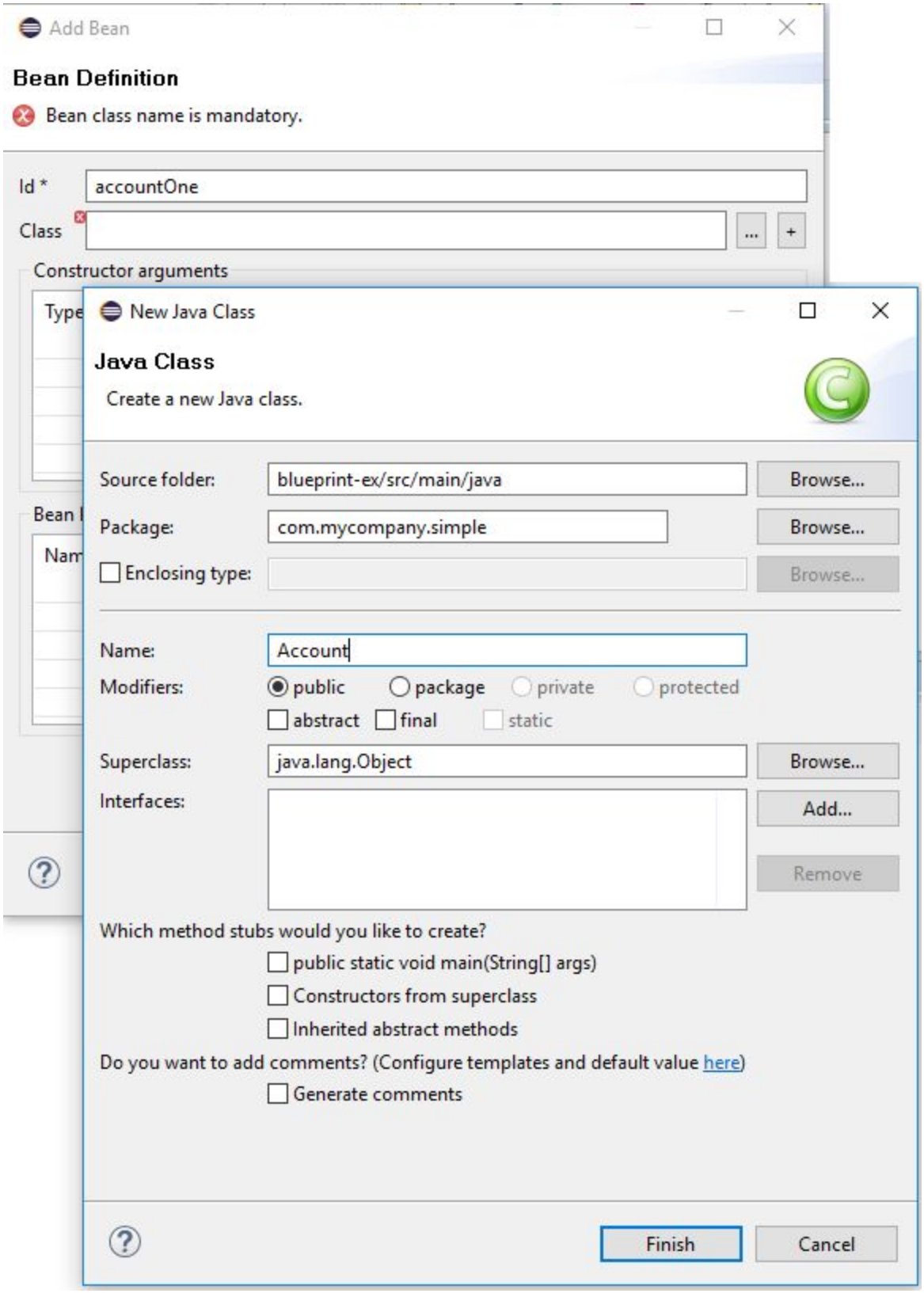
- Support for both Blueprint and Spring XML DSL.
- Create a new Bean on the Configurations tab that references an existing class in the project classpath.
- Create a new Bean on the Configurations tab with a new Java Class on the project classpath.
- Add, remove, and edit simple constructor arguments.
- Add, remove, and edit name/value properties for injection.
- Edit Bean configuration elements via the property page.
- Configure depends-on, scope, factory-method, init-method, and destroy-method via the property page.
- Configure the scope (singleton or prototype) via drop-down

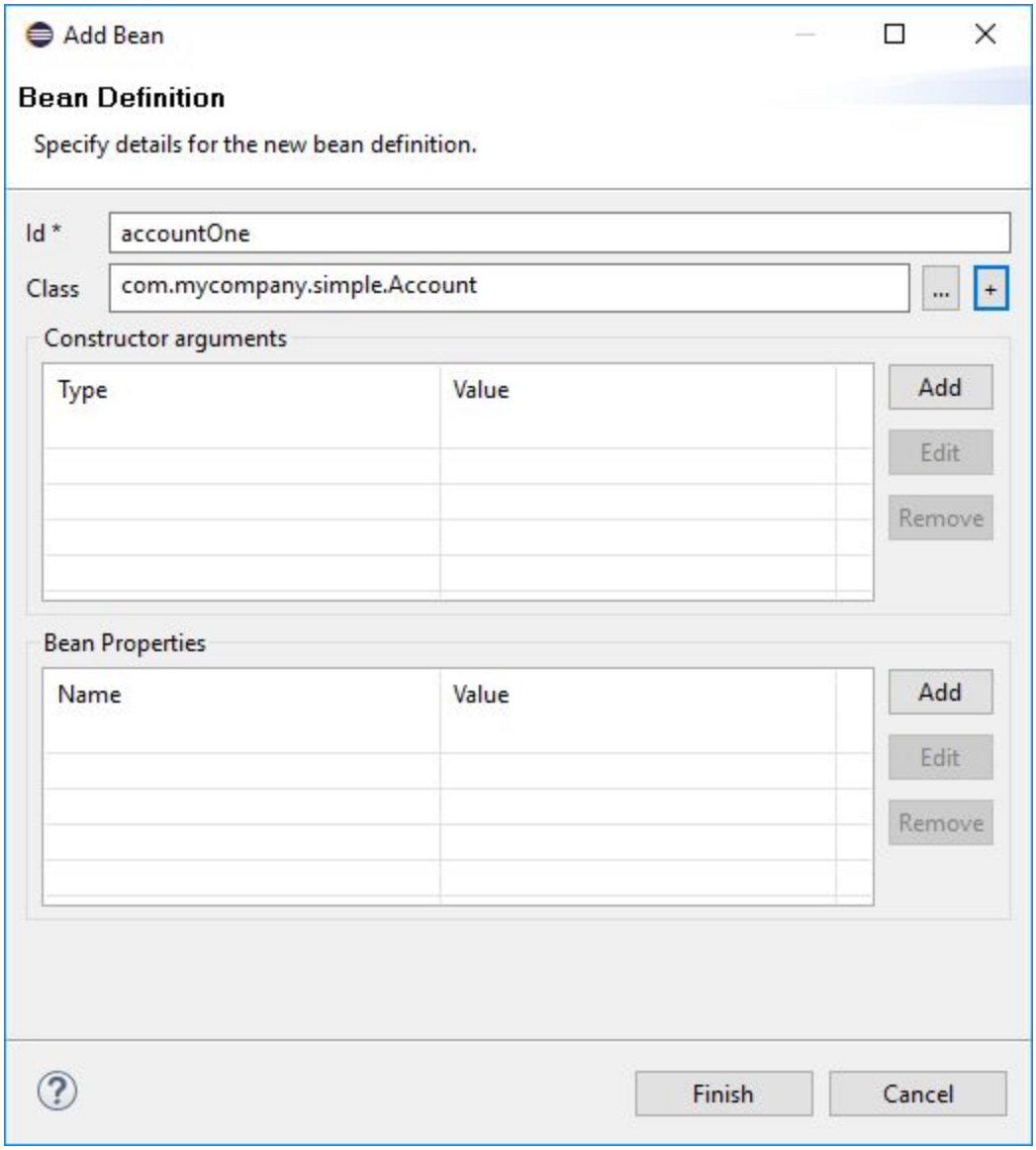
Steps

1. Open your Spring- or Blueprint XML configuration in the Camel Editor.
2. Select the Configurations tab.
3. Click the Add button to bring up the “Create new global element…” dialog.

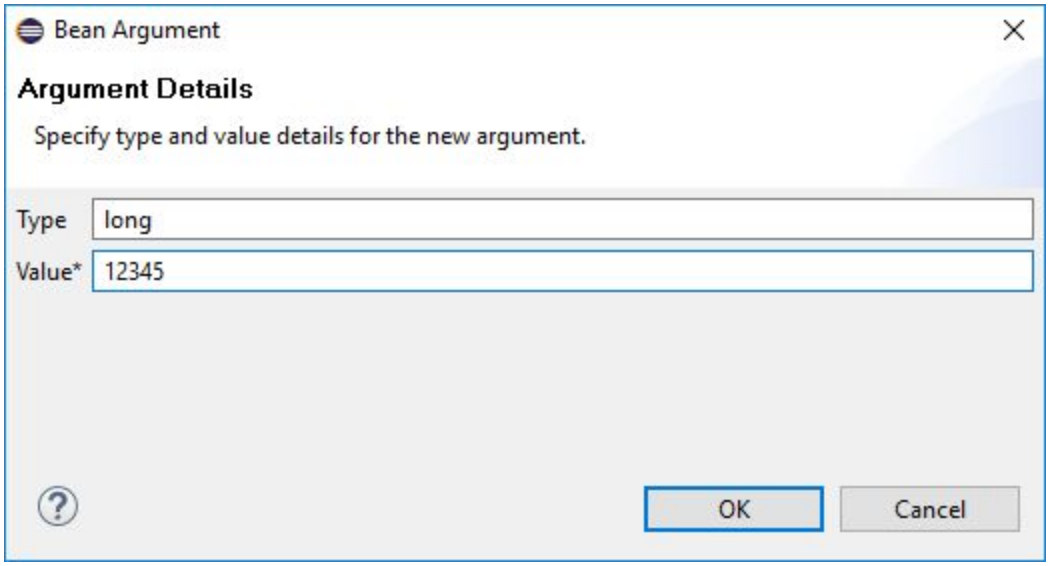


4. Select Bean and click OK.
5. On the “Add Bean” dialog, specify a unique ID for the new bean.
6. Then provide the full Class with package name. You can type it directly and it will validate to ensure that the class is on the project classpath once you stop.
   Or select the “...” button to browse for a class on the project classpath.
   Or select the “+” button to use the New Java Class wizard to create a new class in the project.

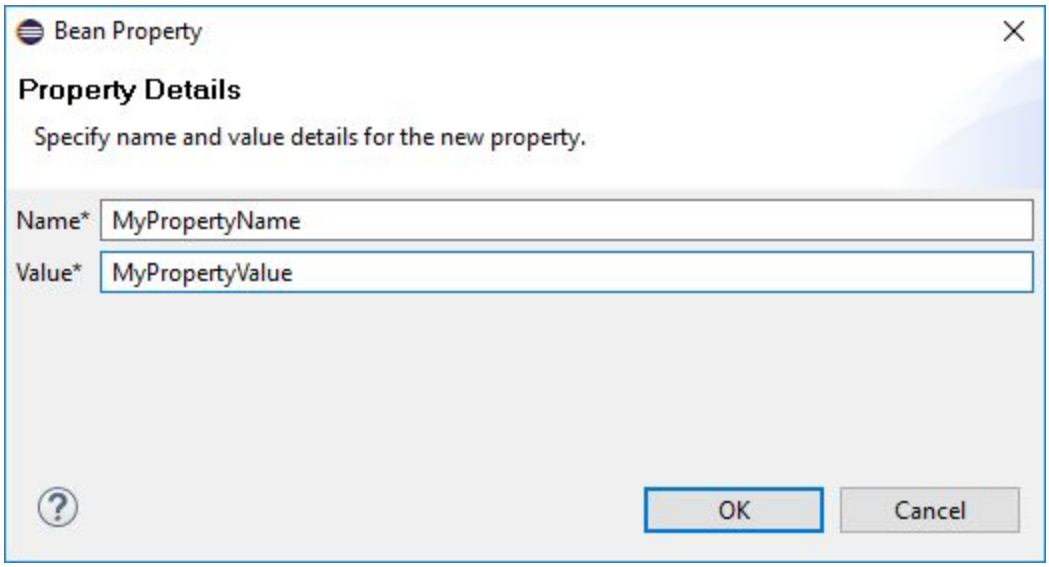

Add Bean
Bean Definition
Bean class name is mandatory.
Id *
accountOne
Class
Constructor arguments
New Java Class
Java Class
Create a new Java class.
Source folder:
blueprint-ex/src/main/java
Browse...
Package:
com.mycompany.simple
Browse...
Enclosing type:
Browse...
Name:
Account
Modifiers:
public
package
private
protected
abstract
final
static
Superclass:
java.lang.Object
Browse...
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel



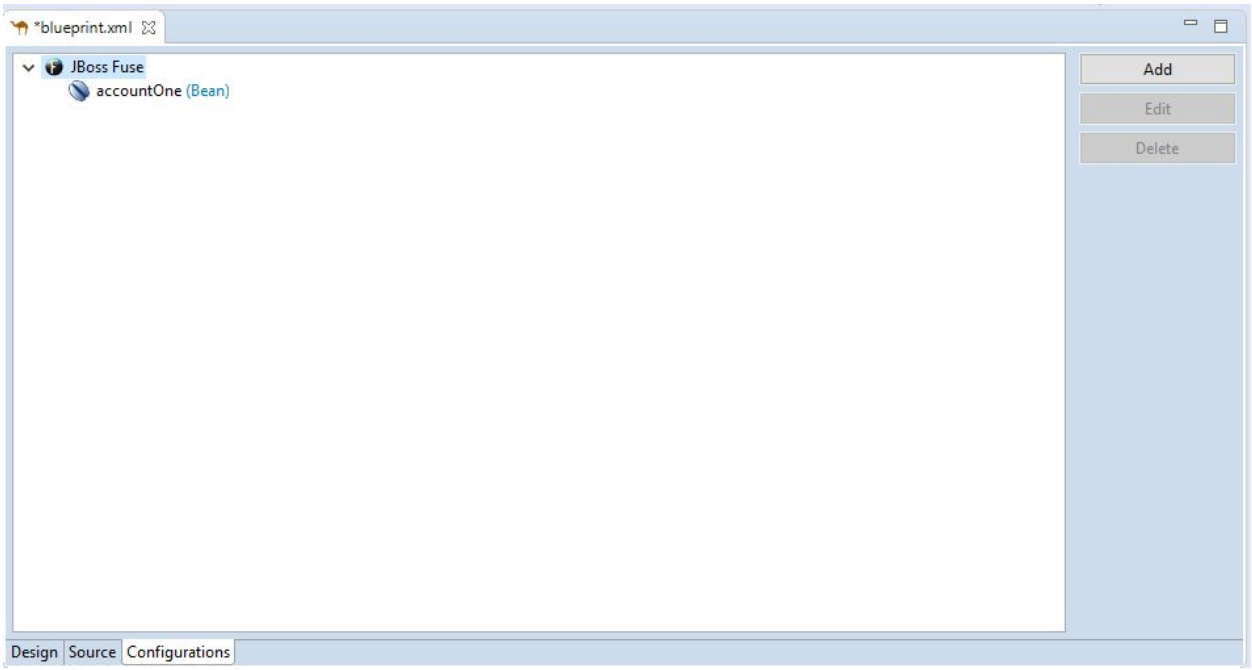
7. (Optional) If the class constructor has any constructor arguments, you can add them in the Constructor arguments table. Click Add to open the “Bean Argument” dialog and specify a type (optional, java.lang.String is default) and a value (required).



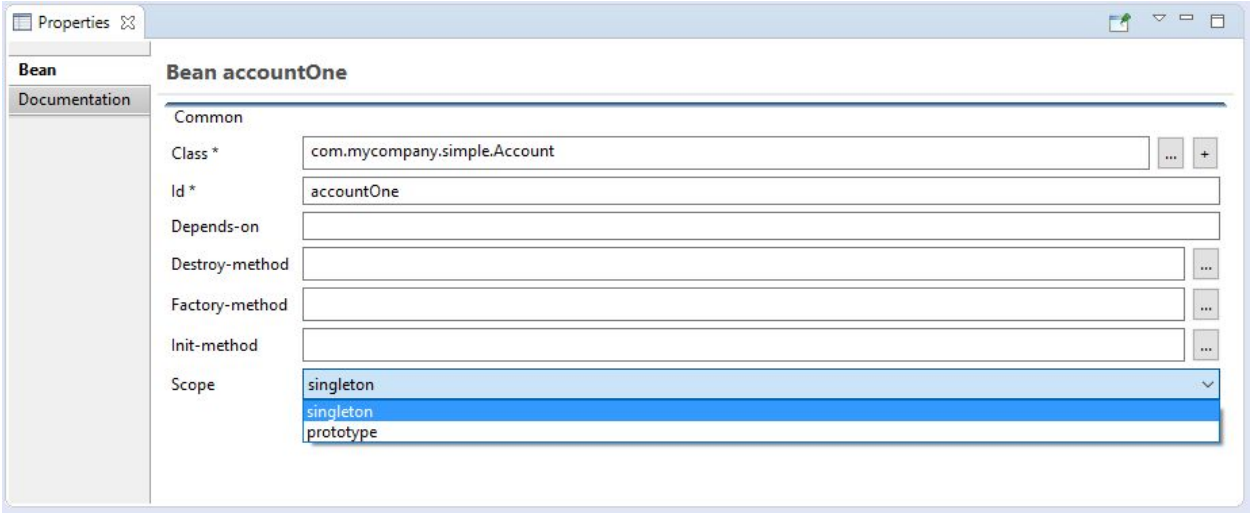
8. (Optional) If the bean requires any properties for injection, you can specify those as well. Click Add in the Bean properties table to open the “Bean Property” dialog and specify a Name and Value (both required). If you specify multiple properties for the bean, all properties must have unique names.



9. Click Finish in the Add Bean wizard to finish creating your new bean.



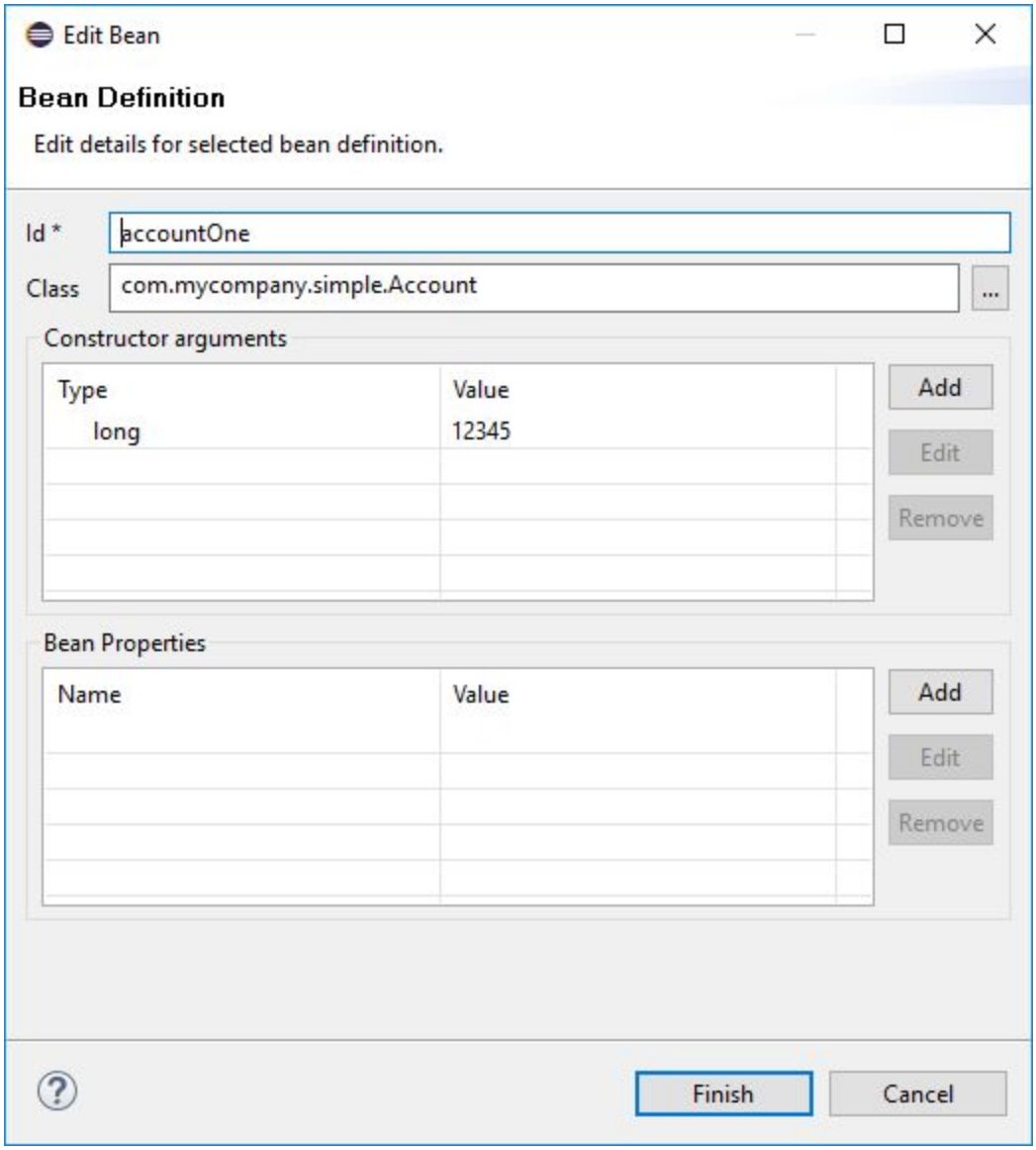
To edit the properties for your new bean, you can select it in the Configurations tab to see the Properties sheet:



From here you can change the referenced class and ID as well as modify several other advanced properties:

- Depends-on - simple string field
- Destroy-method, Init-method, and Factory-method - specify the method name or use the "..." button to open a browse dialog and select one
- Scope - read-only combo box with singleton & prototype (singleton is default)

To edit the constructor arguments or Property elements for the bean, you must select the bean on the Configurations tab and click the Edit button. This opens the “Edit Bean” dialog, which allows you to edit the id, class, arguments, and properties as you did when you created the bean.



Keep in mind that when you edit the class, you may be invalidating any arguments, methods, or properties specified in the bean configuration, so we advise caution.