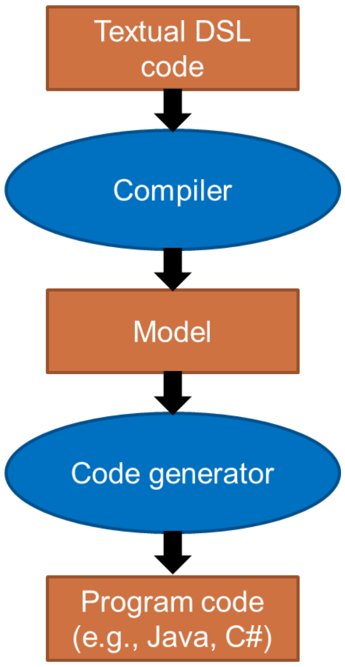
Practice 3 – Preparing the projects

Dr. Balázs Simon, 2024

1 Introduction

This guideline shows how to create the projects needed to solve the practice.

The processing of a textual DSL consist of the following steps:



The compiler builds a model from the textual DSL code. In order to be able to do this, the elements of the model and their relationships must be specified. This is the purpose of the meta-model. When defining the compiler, the mapping of the DSL source code elements to the meta-model elements must also be specified. If this mapping is accurate enough, the model can be built automatically by the compiler. By traversing the model, a code generator can produce some useful output code.

To follow this guideline, you will need the **Eclipse IDE for Java and DSL Developers** tool from the following link:

https://www.eclipse.org/downloads/packages/release/2023-06/r/eclipse-ide-java-and-dsl-developers

Important: Use exactly this version and this edition of Eclipse, otherwise, you may run into problems due to version conflicts in the referenced libraries.

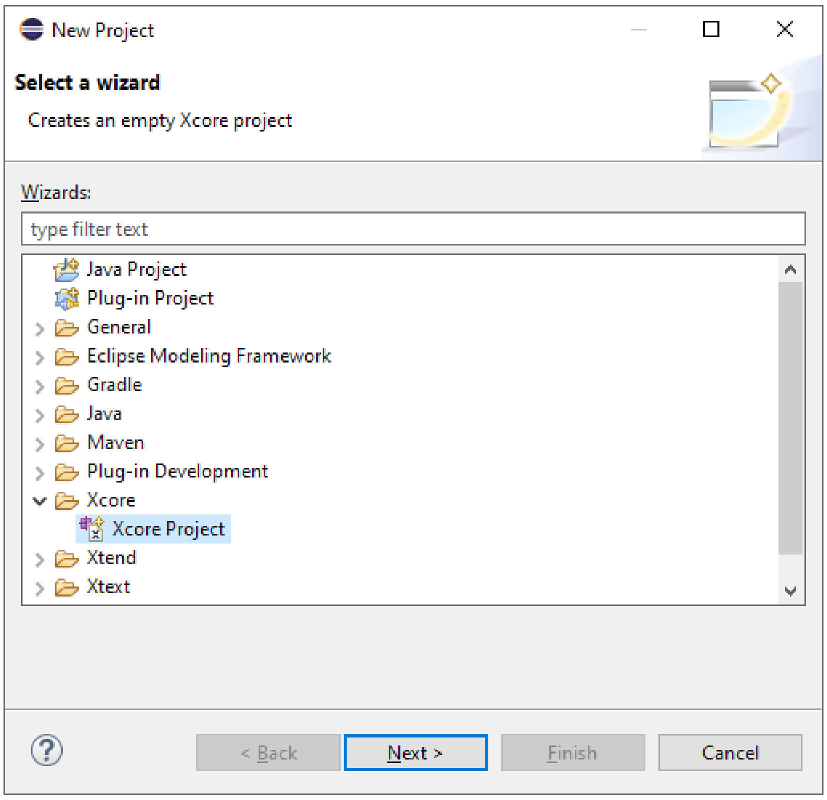
Remarks: the **MANIFEST.MF** files appearing in this guideline are taken from a previous semester, so the version numbers in them may be different than the version numbers used by the Eclipse version downloaded from the link above. Ignore the version numbers used in this guideline, and use the version numbers produced by your own Eclipse. Your own **MANIFEST.MF** files should only be modified by adding the parts marked with yellow background.

2 Goal

The goal is to create a compiler and tools support for a simple state machine description language. To prepare for this, let’s create the projects as described in the following chapters!

3 Project for the meta-model: Xcore

Create a new **Xcore** project by clicking on **File > New > Project…**:



New Project

Select a wizard

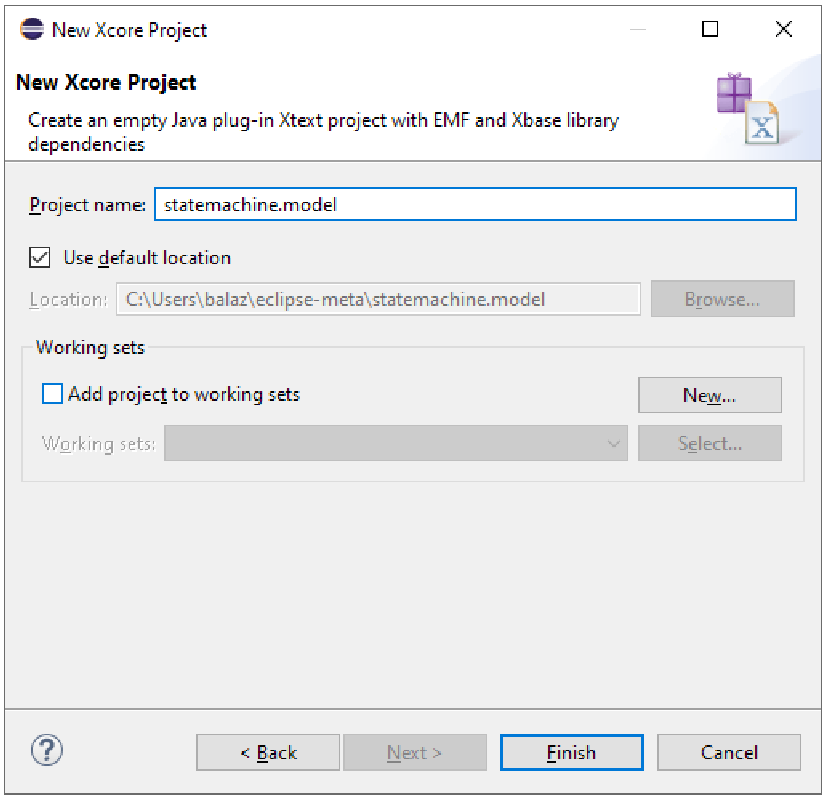
Creates an empty Xcore project

Wizards:

type filter text

- Java Project
- Plug-in Project
- General
- Eclipse Modeling Framework
- Gradle
- Java
- Maven
- Plug-in Development
- Xcore
  - Xcore Project
- Xtend
- Xtext

Click **Next**. The name of the project should be **statemachine.model**:


New Xcore Project

New Xcore Project

Create an empty Java plug-in Xtext project with EMF and Xbase library dependencies

Project name: statemachine.model

☑ Use default location

Location: C:\Users\balaz\eclipse-meta\statemachine.model

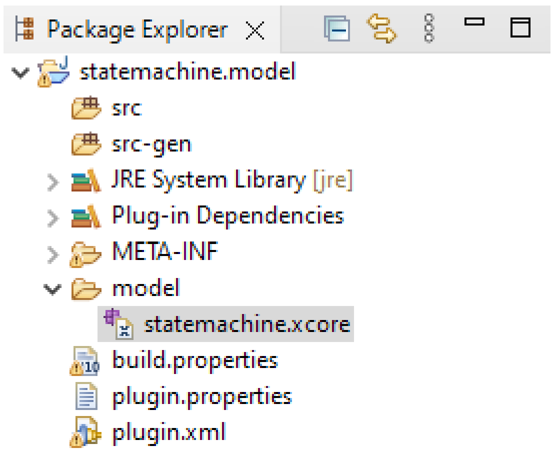
Working sets

☐ Add project to working sets

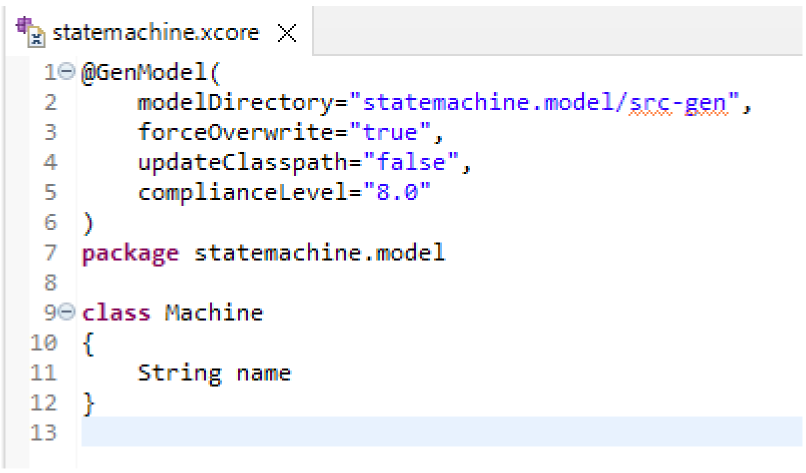
Working sets:

Click **Finish**.

Under the **model** folder, create a file called **statemachine.xcore**:



The contents of the file should be the following:



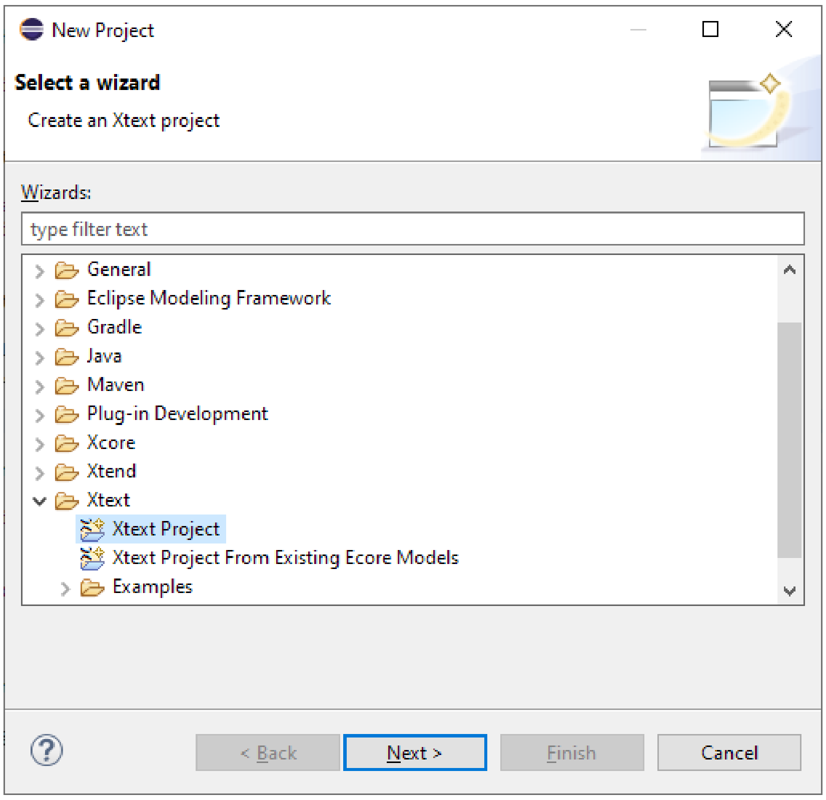
```
@GenModel(
	modelDirectory="statemachine.model/src-gen",
	forceOverwrite="true",
	updateClasspath="false",
	complianceLevel="8.0"
)
package statemachine.model

class Machine
{
	String name
}
```

Extend the **META-INF/MANIFEST.MF** file with the following lines:

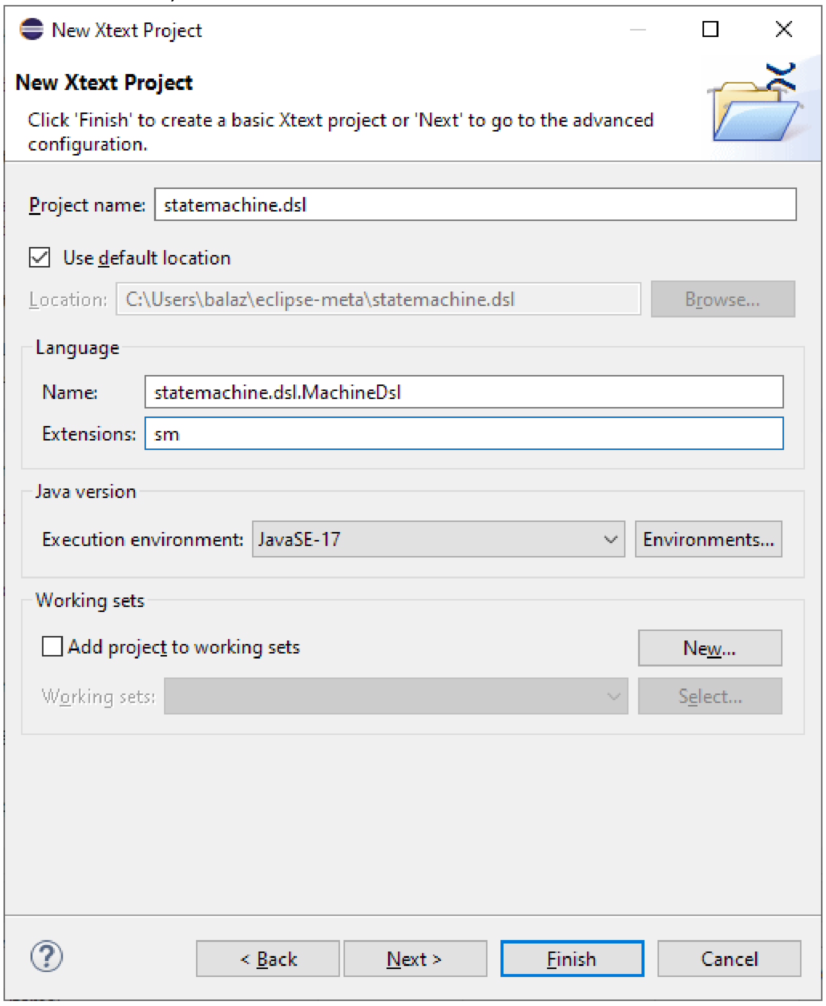
```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: %pluginName
Bundle-SymbolicName: statemachine.model;singleton:=true
Automatic-Module-Name: statemachine.model
Bundle-Version: 0.1.0.qualifier
Bundle-ClassPath: .
Bundle-Vendor: %providerName
Bundle-Localization: plugin
Bundle-RequiredExecutionEnvironment: JavaSE-17
Require-Bundle: org.eclipse.core.runtime,
 org.eclipse.emf.ecore;visibility:=reexport,
 org.eclipse.xtext.xbase.lib,
 org.eclipse.emf.ecore.xcore.lib
Bundle-ActivationPolicy: lazy
Export-Package: statemachine.model
```

4 Project for the compiler: Xtext

Create a new **Xtext** project by clicking on **File > New > Project…**:



Click **Next**. The name of the project should be **statemachine.dsl**, the name of the language should be **statemachine.dsl.MachineDsl**, and the extension of our state machine files should be **sm**:



New Xtext Project

New Xtext Project

Click 'Finish' to create a basic Xtext project or 'Next' to go to the advanced configuration.

Project name: statemachine.dsl

☑ Use default location

Location: C:\Users\balaz\eclipse-meta\statemachine.dsl

Language

Name: statemachine.dsl.MachineDsl

Extensions: sm

Java version

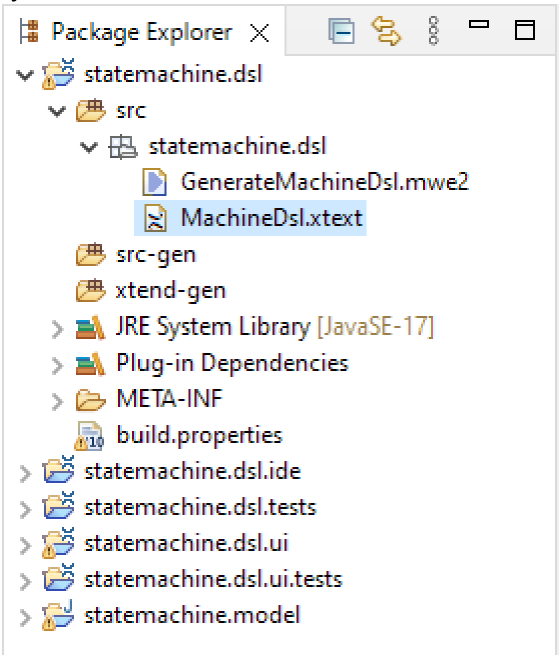
Execution environment: JavaSE-17

Working sets

☐ Add project to working sets

Working sets:

Click **Finish**. Some other projects are also created besides the **statmachine.dsl** project:

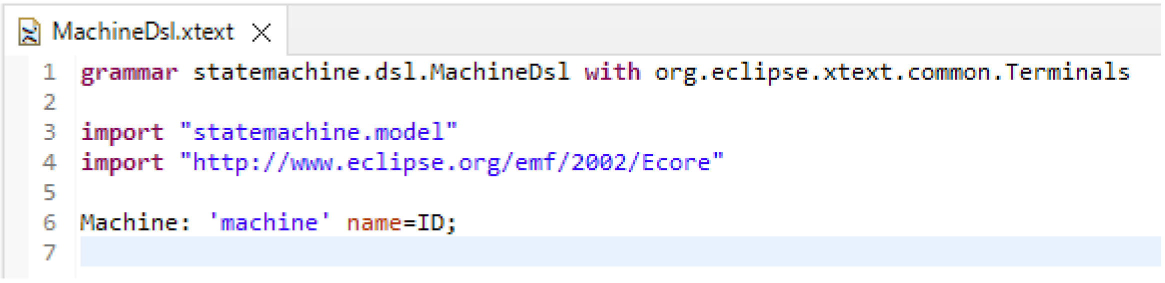


```
Package Explorer
statemachine.dsl
  src
    statemachine.dsl
      GenerateMachineDsl.mwe2
      MachineDsl.xtext
  src-gen
  xtend-gen
  JRE System Library [JavaSE-17]
  Plug-in Dependencies
  META-INF
  build.properties
statemachine.dsl.ide
statemachine.dsl.tests
statemachine.dsl.ui
statemachine.dsl.ui.tests
statemachine.model
```

For now, we are only interested in the **statemachine.dsl** project. The other project is useful for the Eclipse IDE support and testing:

- **statemachine.dsl.ide**: main IDE functions which are independent of the graphical IDE interface
- **statemachine.dsl.ui**: IDE functions dependent on the graphical IDE interface
- **statemachine.dsl.test**: unit tests for our DSL
- **statemachine.dsl.ui.tests**: unit tests for the IDE support of our DSL

Modify the contents of **MachineDsl.xtext** as follows:



```
grammar statemachine.dsl.MachineDsl with org.eclipse.xtext.common.Terminals

import "statemachine.model"
import "http://www.eclipse.org/emf/2002/Ecore"

Machine: 'machine' name=ID;

```

For the compiler to be able to see our meta-model, we have to add the following line with yellow background to the **GenerateMachineDsl.mwe2** file:

```
module statemachine.dsl.GenerateMachineDsl

import org.eclipse.xtext.xtext.generator.*
import org.eclipse.xtext.xtext.generator.model.project.*

var rootPath = ".."

Workflow {

  component = XtextGenerator {
    configuration = {
      project = StandardProjectConfig {
        baseName = "statemachine.dsl"
        rootPath = rootPath
        runtimeTest = {
          enabled = true
        }
        eclipsePlugin = {
          enabled = true
        }
        eclipsePluginTest = {
          enabled = true
        }
        createEclipseMetaData = true
      }
      code = {
        encoding = "UTF-8"
        lineDelimiter = "\r\n"
        fileHeader = "/*\n * generated by Xtext \${version}\n */"
        preferXtendStubs = false
      }
    }
    language = StandardLanguage {
      name = "statemachine.dsl.MachineDsl"
      fileExtensions = "sm"
      referencedResource      =      "platform:/resource/statemachine.model/model/
statemachine.xcore"

      serializer = {
        generateStub = false
      }
      validator = {
        // composedCheck = "org.eclipse.xtext.validation.NamesAreUniqueValidator"
        // Generates checks for @Deprecated grammar annotations, an IssueProvider
and a corresponding PropertyPage
        generateDeprecationValidation = true
      }
      generator = {
        generateXtendStub = true
      }
      junitSupport = {
        junitVersion = "5"
      }
    }
  }
}
```

In addition, the **META-INF/MANIFEST.MF** file must be extended with the following lines:

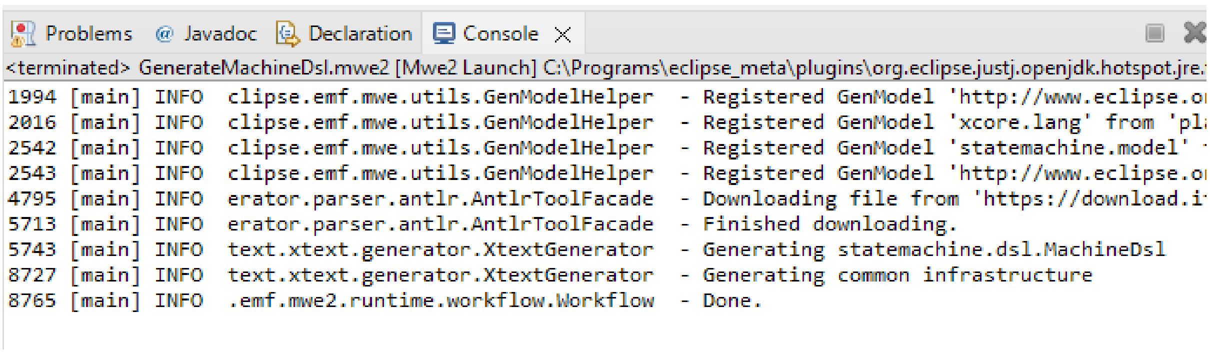
```
Manifest-Version: 1.0
Automatic-Module-Name: statemachine.dsl
Bundle-ManifestVersion: 2
Bundle-Name: statemachine.dsl
Bundle-Vendor: My Company
Bundle-Version: 1.0.0.qualifier
Bundle-SymbolicName: statemachine.dsl; singleton:=true
Bundle-ActivationPolicy: lazy
Require-Bundle: org.eclipse.xtext,
 org.eclipse.xtext.xbase,
 org.eclipse.equinox.common;bundle-version="3.16.0",
 org.eclipse.emf.ecore.xcore.lib,
 org.eclipse.emf.ecore.xcore,
 statemachine.model
Bundle-RequiredExecutionEnvironment: JavaSE-17
```

Also, in the **statemachine.dsl.tests** project, the **META-INF/MANIFEST.MF** file must be extended:

```
Manifest-Version: 1.0
Automatic-Module-Name: statemachine.dsl.tests
Bundle-ManifestVersion: 2
Bundle-Name: statemachine.dsl.tests
Bundle-Vendor: My Company
Bundle-Version: 1.0.0.qualifier
Bundle-SymbolicName: statemachine.dsl.tests; singleton:=true
Bundle-ActivationPolicy: lazy
Require-Bundle: statemachine.dsl,
 org.eclipse.xtext.testing,
 org.eclipse.xtext.xbase.testing,
 statemachine.model
Import-Package: org.junit.jupiter.api;version="[5.1.0,6.0.0)",
 org.junit.jupiter.api.extension;version="[5.1.0,6.0.0)"
```

```
Bundle-RequiredExecutionEnvironment: JavaSE-17
```

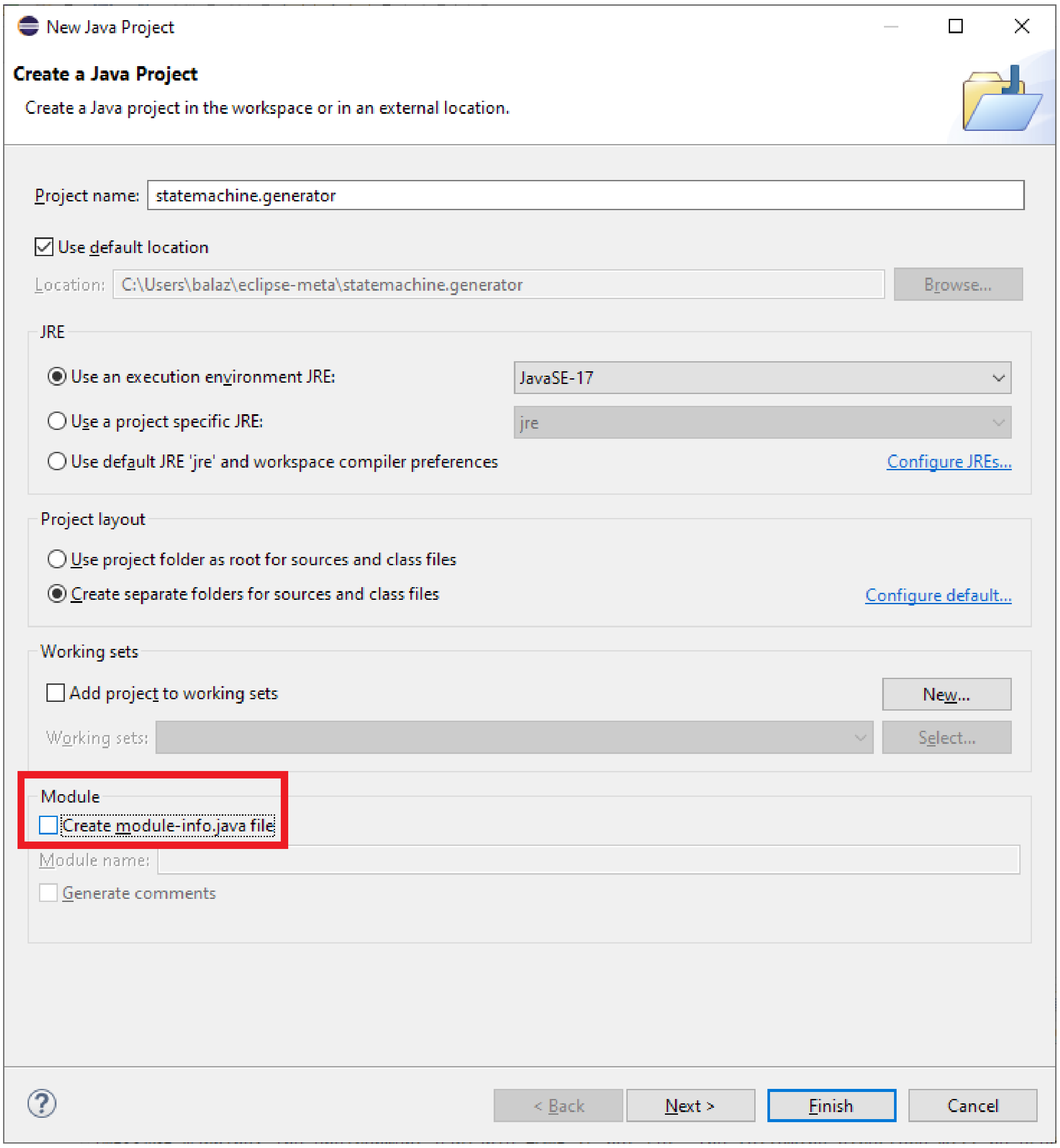
When you are ready with the steps above, click on **GenerateMachineDsl.mwe2** with the right mouse button, and select **Run As > MWE2 Workflow**. If everything was done correctly, the process should end without any errors:



```
<terminated> GenerateMachineDsl.mwe2 [Mwe2 Launch] C:\Programs\eclipse_meta\plugins\org.eclipse.justj.openjdk.hotspot.jre.
1994 [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'http://www.eclipse.o
2016 [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'xcore.lang' from 'pl
2542 [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'statemachine.model' 
2543 [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'http://www.eclipse.o
4795 [main] INFO  erator.parser.antlr.AntlrToolFacade  - Downloading file from 'https://download.i
5713 [main] INFO  erator.parser.antlr.AntlrToolFacade  - Finished downloading.
5743 [main] INFO  text.xtext.generator.XtextGenerator  - Generating statemachine.dsl.MachineDsl
8727 [main] INFO  text.xtext.generator.XtextGenerator  - Generating common infrastructure
8765 [main] INFO  .emf.mwe2.runtime.workflow.Workflow  - Done.
```

5 Project for the code generator: Xtend

Click **File > New > Java Project** to create a new Java project called **statemachine.generator**! Remove the checkmark before the „Create module-info.java file" option:


New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: statemachine.generator

[✓] Use default location

Location: C:\Users\balaz\eclipse-meta\statemachine.generator

JRE

[●] Use an execution environment JRE: JavaSE-17

○ Use a project specific JRE: jre

○ Use default JRE 'jre' and workspace compiler preferences

Project layout

○ Use project folder as root for sources and class files

[●] Create separate folders for sources and class files

Working sets

[ ] Add project to working sets

Working sets:

Module

[ ] Create module-info.java file

Module name:

[ ] Generate comments

Click **Finish** to create the project!

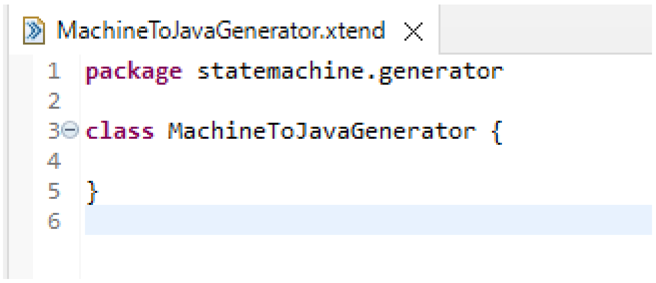
Right click on the **statemachine.generator** project, and select **Configure > Convert to Xtext Project**!

Right click again on the **statemachine.generator** project, and select **Configure > Convert to Plug-in Projects**! Close the appearing window with the **Finish** button.

Add the following lines to the **META-INF/MANIFEST.MF** file:

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Generator
Bundle-SymbolicName: statemachine.generator
Bundle-Version: 1.0.0.qualifier
Bundle-RequiredExecutionEnvironment: JavaSE-17
Automatic-Module-Name: statemachine.generator
Require-Bundle: org.eclipse.xtext,
 org.eclipse.xtext.xbase,
 statemachine.model
Export-Package: statemachine.generator
```

Under the **src** folder inside the **statemachine.generator** package create a file **MachineToJavaGenerator.xtend** with the following content:



```
package statemachine.generator

class MachineToJavaGenerator {

}
```

Inside the **statemachine.dsl** project modify the **META-INF/MANIFEST.MF** file as follows:

```
Manifest-Version: 1.0
Automatic-Module-Name: statemachine.dsl
Bundle-ManifestVersion: 2
Bundle-Name: statemachine.dsl
Bundle-Vendor: My Company
Bundle-Version: 1.0.0.qualifier
Bundle-SymbolicName: statemachine.dsl; singleton:=true
Bundle-ActivationPolicy: lazy
Require-Bundle: org.eclipse.xtext,
 org.eclipse.xtext.xbase,
 org.eclipse.equinox.common;bundle-version="3.16.0",
 org.eclipse.emf.ecore.xcore.lib,
 org.eclipse.emf.ecore.xcore,
 statemachine.model,
 statemachine.generator,
 org.eclipse.xtext.xbase.lib;bundle-version="2.14.0",
 org.eclipse.xtext.util,
 org.antlr.runtime;bundle-version="[3.2.0,3.2.1)"
Bundle-RequiredExecutionEnvironment: JavaSE-17
Export-Package: statemachine.dsl.serializer,
 statemachine.dsl.parser.antlr,
 statemachine.dsl.generator,
 statemachine.dsl.validation,
 statemachine.dsl,
 statemachine.dsl.services,
 statemachine.dsl.scoping,
 statemachine.dsl.parser.antlr.internal
Import-Package: org.apache.log4j
```