

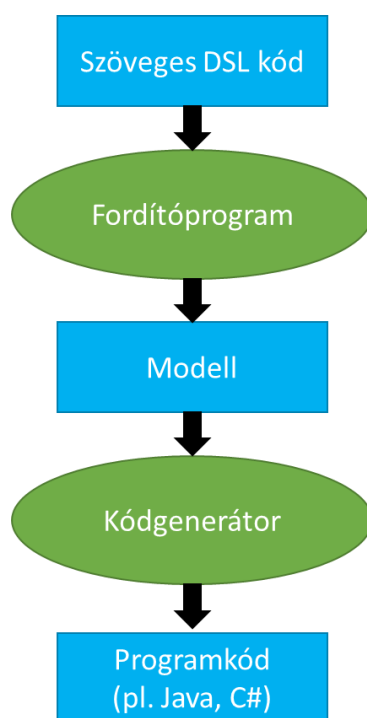
3. Gyakorlat – Projektek létrehozása

Dr. Simon Balázs, 2024.

1 Bevezetés

Ez az útmutató a gyakorlat megoldásához szükséges projektek előkészítését mutatja be.

Egy szöveges DSL feldolgozása az alábbi lépésekből áll:



A szöveges DSL kódból a fordító egy modellt épít. Ehhez meg kell adni, hogy a modell milyen elemekből áll, és azok hogyan kapcsolódnak egymáshoz. Erre szolgál a metamodel. A fordító definiálásánál meg kell adni azt is, hogy a kód elemei hogyan képezhetők le a metamodel egyes elemeire. Ha ez a leképezés elég pontos, a modellt a fordító automatikusan fel tudja építeni. A felépített modell bejárásával egy kódgenerátor készíthet valamilyen kimeneti kódot.

A feladat megoldásához **Eclipse IDE for Java and DSL Developers** változatra lesz szükség az alábbi weboldalról:

<https://www.eclipse.org/downloads/packages/release/2023-12/r/eclipse-ide-java-and-dsl-developers>

Pontosan ezt a verziót és ezt a változatot használjuk, különben problémák lehetnek a felhasznált könyvtárak verziószámaival!

Megjegyzés: az ebben az útmutatóban szereplő **MANIFEST.MF** fájlok egy korábbi féléből lettek átvéve, így az azokban olvasható verziószámok eltérhetnek a fenti Eclipse verzió által használt

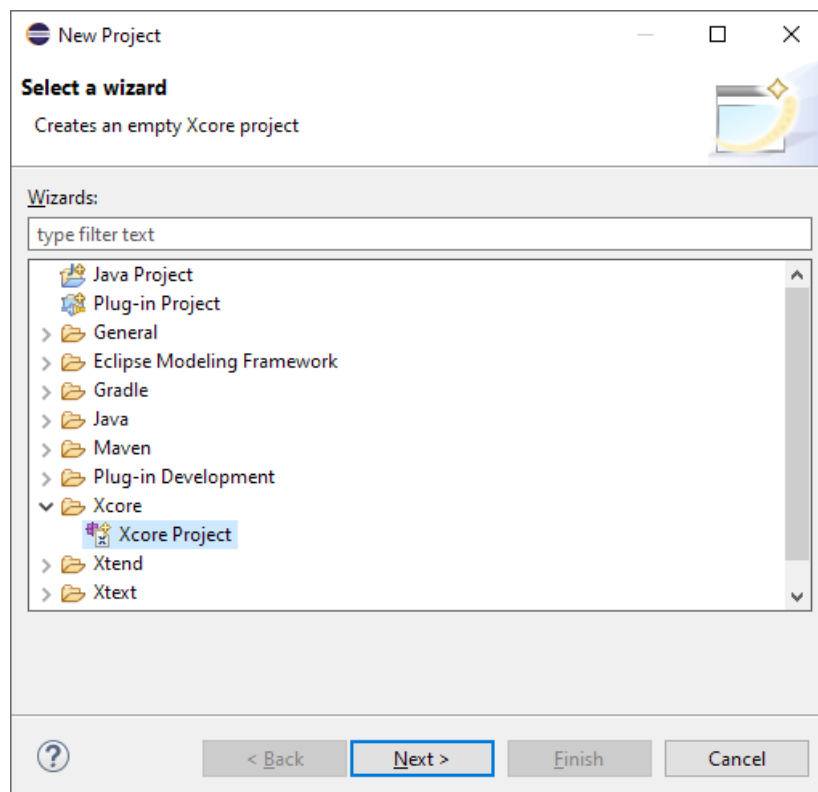
verziószámoktól. Az ebben az útmutatóban szereplő verziószámokat hagyjuk figyelmen kívül, maradjunk a saját Eclipse-ünk által használt verziószámoknál, és a saját **MANIFEST.MF** fájljainkhoz csak a sárga háttérű részeket adjuk hozzá!

2 Cél

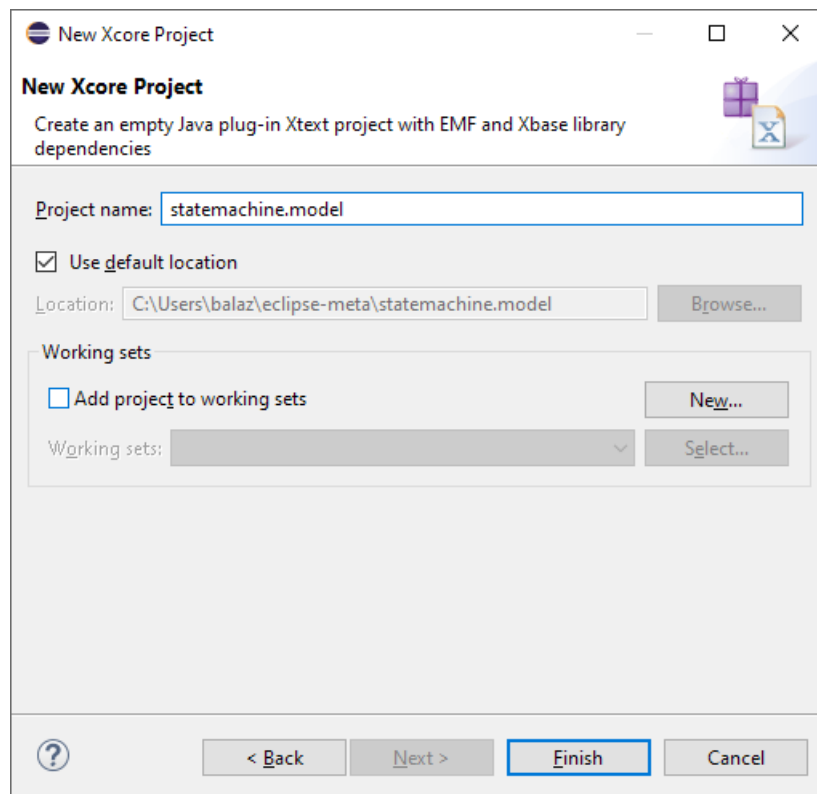
A cél, hogy egy egyszerű állapotgép-leíró nyelvhez készítsünk fordítót és eszköztámogatást. Ennek előkészítéséhez hozzuk létre a projekteket a következő fejezetek által leírt módon!

3 Metamodell projekt: Xcore

A **File > New > Project...** menüpontra kattintva hozzunk létre egy új **Xcore** projektet:

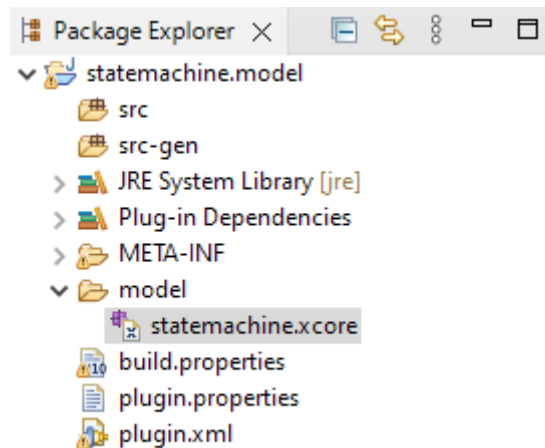


Kattintsunk a **Next**-re. A projekt neve legyen **statemachine.model**:

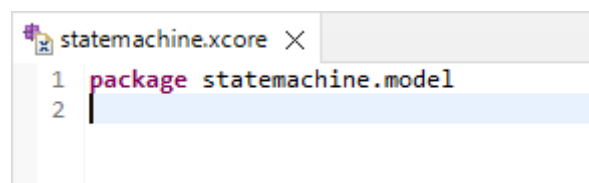


Kattintsunk a **Finish**-re.

A **model** könyvtár alatt hozunk létre egy **statemachine.xcore** fájlt:

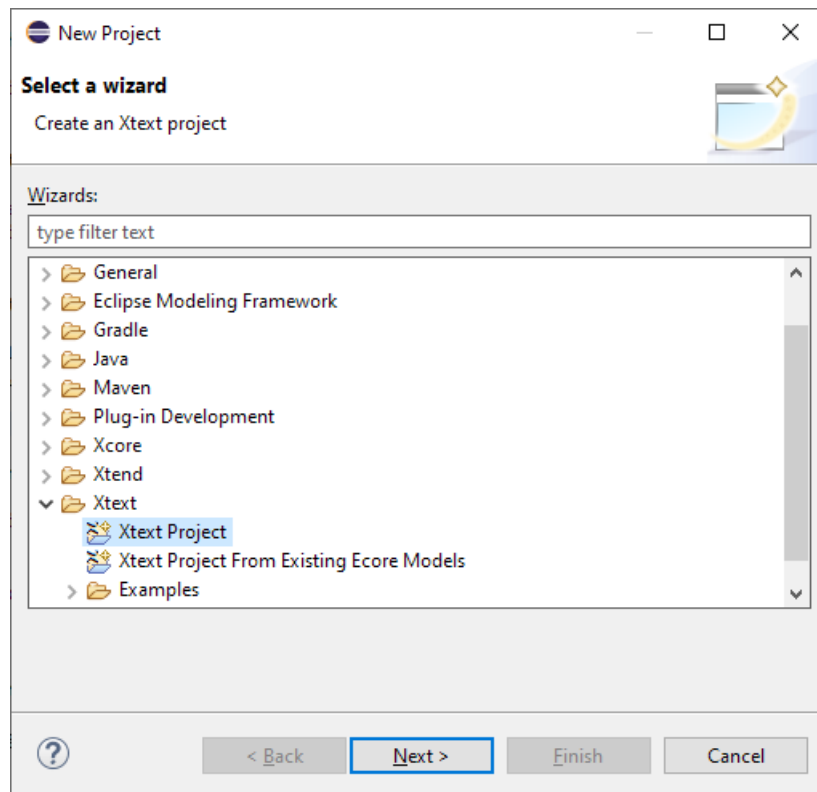


A fájl tartalma az alábbi legyen:

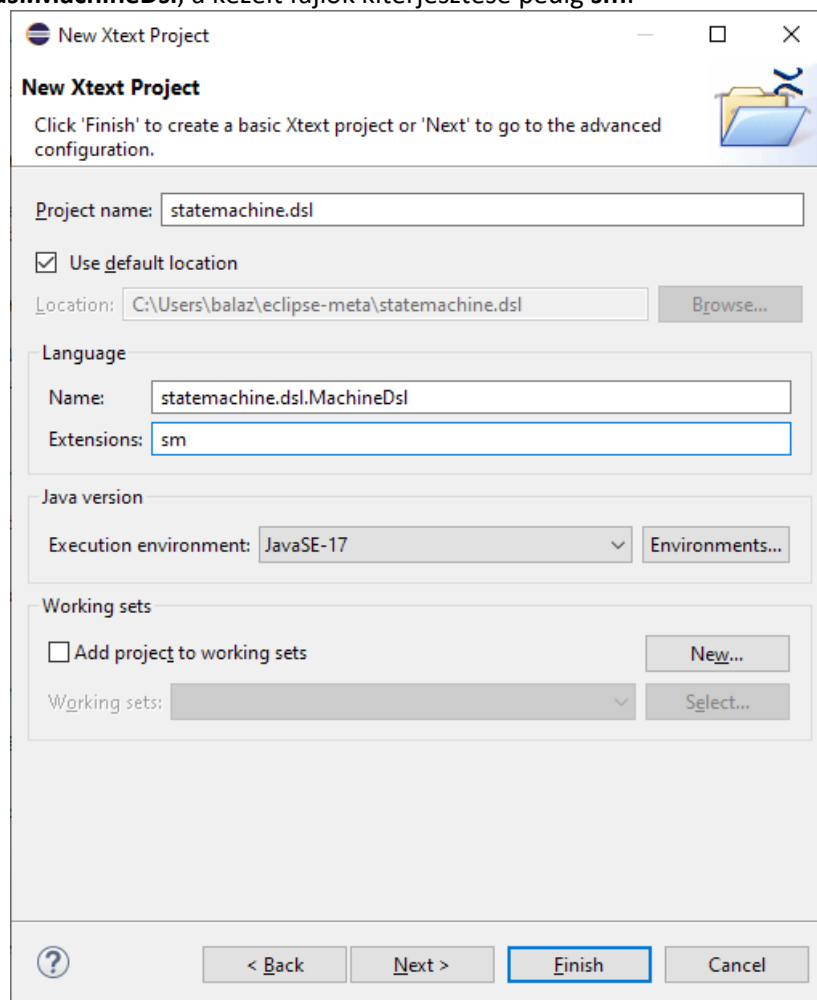


4 Fordító projekt: Xtext

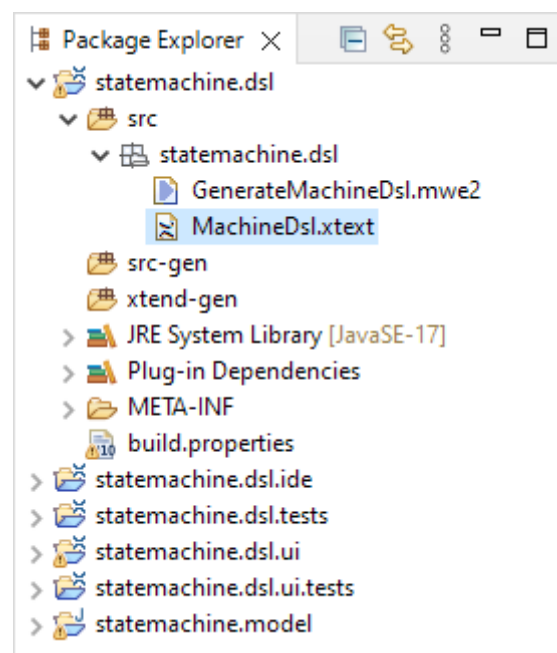
A **File > New > Project...** menüpontra kattintva hozzunk létre egy új **Xtext** projektet:



Kattintsunk a **Next**-re. A projekt neve legyen **statemachine.dsl**, a nyelv neve **statemachine.dsl.MachineDsl**, a kezelt fájlok kiterjesztése pedig **sm**:



Kattintsunk a **Finish**-re. Ekkor a **statemachine.dsl** projekt mellett még létrejön jónéhány másik projekt is:



Első körben számunkra az **entities.dsl** projekt érdekes, a többi a DSL Eclipse támogatását implementálja, illetve teszteket futtat:

- **statemachine.dsl.ide**: legfontosabb IDE funkciók, amelyek függetlenek a grafikus IDE felülettől
- **statemachine.dsl.ui**: a grafikus IDE felülettel kapcsolatos IDE funkciók
- **statemachine.dsl.test**: a DSL-ünk unit tesztelését lehet benne elvégezni
- **statemachine.dsl.ui.tests**: a grafikus IDE felülettel kapcsolatos IDE funkciók unit tesztelését lehet benne elvégezni

Ahhoz, hogy a fordító lássa a metamodelünket, ki kell egészíteni a **GenerateEntityDsl.mwe2** fájlt a sárga háttérű sorral:

```
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"
    }
}
}
}
}

```

Továbbá, a **META-INF/MANIFEST.MF** fájlt az alábbi sorokkal:

```

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

```

Az **statemachine.dsl.tests** projekt **META-INF/MANIFEST.MF** fájlját is egészítsük ki:

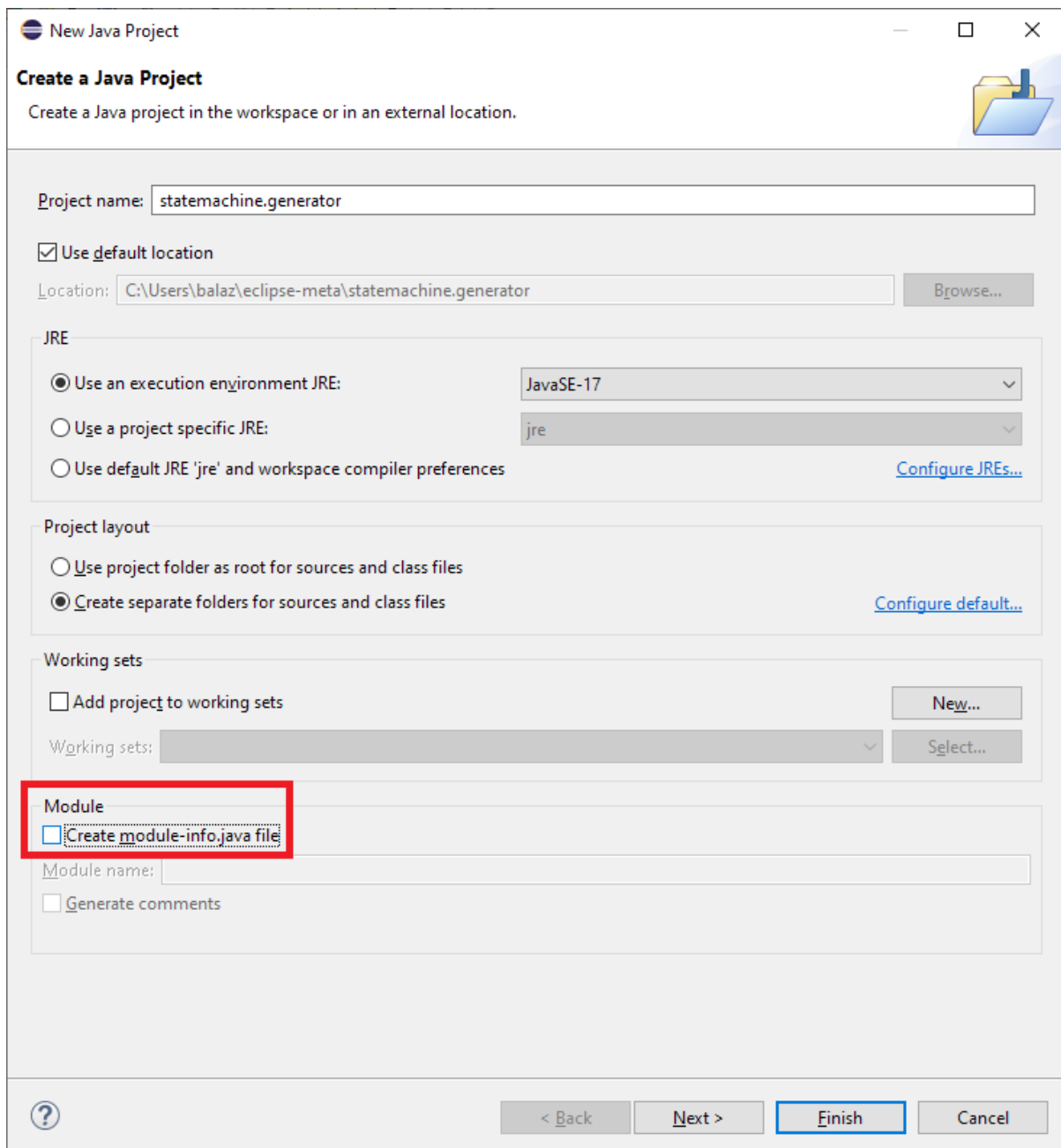
```

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

```

5 Kódgenerátor projekt: Xtend

A **File > New > Java Project** menüpontra kattintva hozzunk létre egy új Java projektet! A projekt neve legyen **statemachine.generator**! A „Create module-info.java file” opció előtt szedjük ki a pipát:



New Java Project

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

Project name:

☒ Use default location

Location: [Browse...](#)

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE 'jre' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

Module

☐ Create module-info.java file

Module name:

☐ Generate comments

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

A **Finish** gombbal hozzuk létre a projektet! Az **src** könyvtár alatt egy **statemachine.generator** nevű csomagban készítünk egy **MachineToJavaGenerator.xtend** nevű fájlt az alábbi tartalommal:

```
MachineToJavaGenerator.xtend ×
1 package statemachine.generator
2
3 class MachineToJavaGenerator {
4
5 }
6
```


Kattintsunk jobb gombbal a **statemachine.generator** projektünkön, és válasszuk a **Configure > Convert to Xtext Project** menüpontot!

Kattintsunk még egyszer jobb gombbal az **statemachine.generator** projektünkön, és válasszuk a **Configure > Convert to Plug-in Projects** menüpontot! A megjelenő ablakot **Finish** gombbal zárjuk le.

Egészítsük ki a **META-INF/MANIFEST.MF** fájlt az alábbi sorokkal:

```
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
```

Egészítsük ki a **statemachine.dsl** projektben a **META-INF/MANIFEST.MF** fájlt az alábbi sorral:

```
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,
    organtlr.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
```