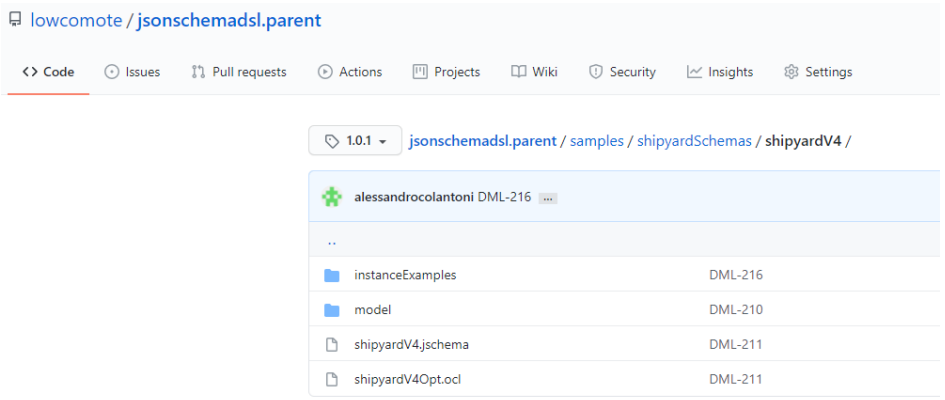
# Language editor generation

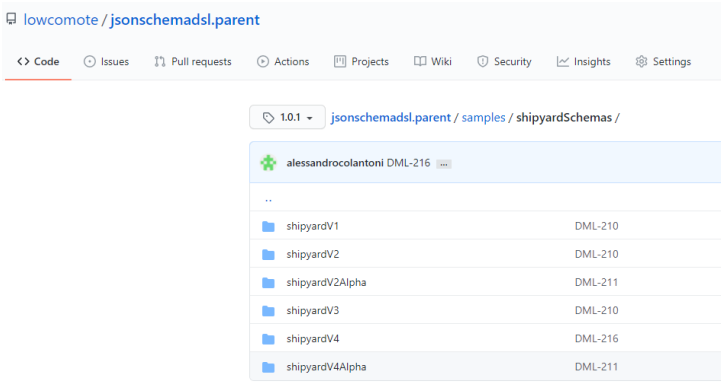
Once installed JsonSchemaDSL and JsonGrammar you can (i) create a json schema, (ii) automatically generate an Ecore metamodel from it, and (iii) generate a concrete json based grammar Xtext, applying the steps described in this tutorial. In other words, in this tutorial we show how to create an editor for the language defined by the json schema created by a user, and how this editor allows the user to manage its artifacts as both json and models.
The screenshots of this tutorial have been taken executing the steps for the shipyardV4.jschema example available in

https://github.com/lowcomote/jsonschemadsl.parent/tree/1.0.1/samples/shipyardSchemas/shipyardV4




You are encouraged to practice trying to reproduce the steps of this tutorial with the same json schema. In the same shipyardV4 folder you can also find the same artifacts that will be generated executing the steps of this tutorial, for a better guide.

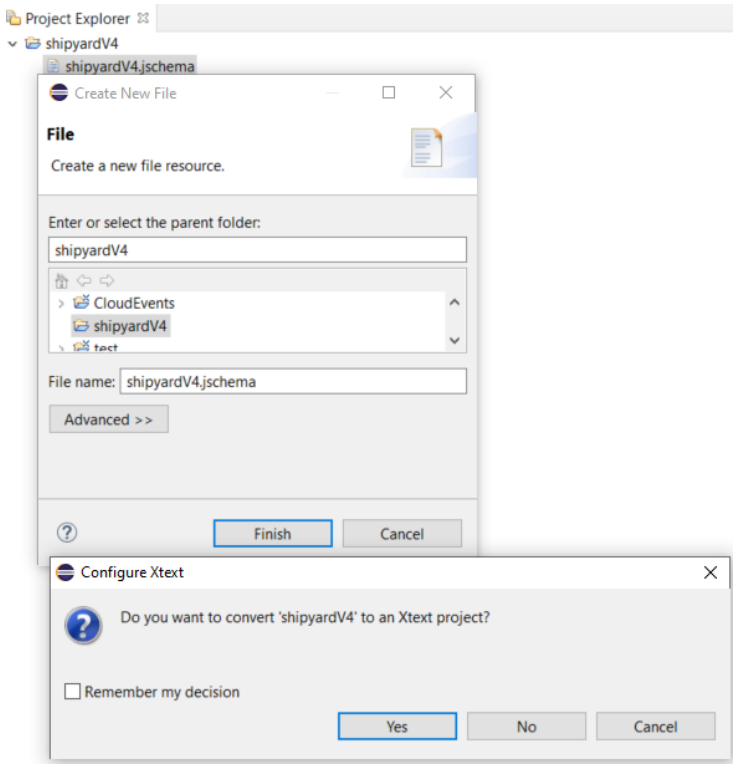
In the link
https://github.com/lowcomote/jsonschemadsl.parent/tree/1.0.1/samples/shipyardSchemas



more versions of the shipyard json schema can be found to practice, but the most of them are examples of not valid json schemas, for the purpose of showing the JsonSchemaDSL capability of detecting them.

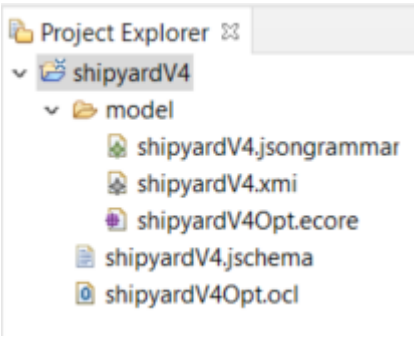
## 1 Create a JsonSchema

- Create a new general project with the New Project wizard (name it shipyardV4)
- Create a file with extension *.jschema* (name it shipyardV4.jschema)
- Click Yes when asked if you want to convert your project to an Xtext project.




## 2 Edit the created Json Schema

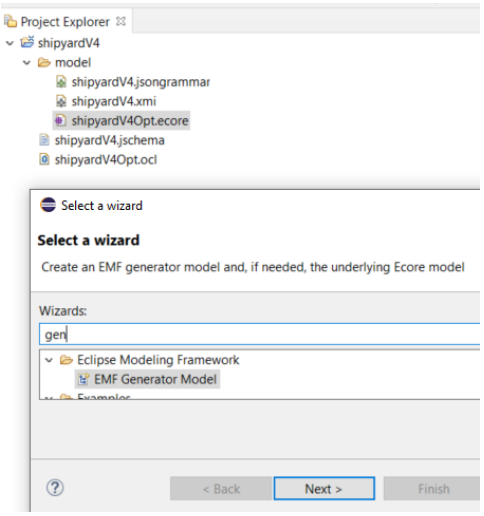
Use the CTRL+Space to use the content assist and code completion. Remember that all the keywords start by double quotes (“).
Open the Problems view and the Properties view.
When you save the modifications, and there are no errors, JsonSchemaDSL will generate the ecore, the jsongrammar and the ocl models, that are the artifacts needed to generate the json based Xtext grammar as described in the steps described in this tutorial.
To properly follow the tutorial, we suggest to copy the content of
https://github.com/lowcomote/jsonschemadsl.parent/blob/1.0.1/samples/shipyardSchemas/shipyardV4/shipyardV4.jschema
and paste it in the file you have created. When you save no error or warnings are detected. You can refresh your project folder, and the model folder inside it to see what it has been created.




If everything has gone well, you should find the same artifact as in https://github.com/lowcomote/jsonschemadsl.parent/tree/1.0.1/samples/shipyardSchemas/shipyardV4

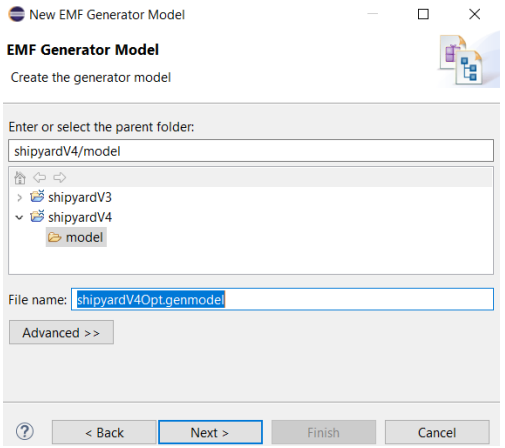
## Json based grammar generation

3. Right click on the generated ecore (e.g., shipyardV4/model/shipyardV4Opt.ecore select *new -> other -> EMF Generator Model*. Click *Next*.



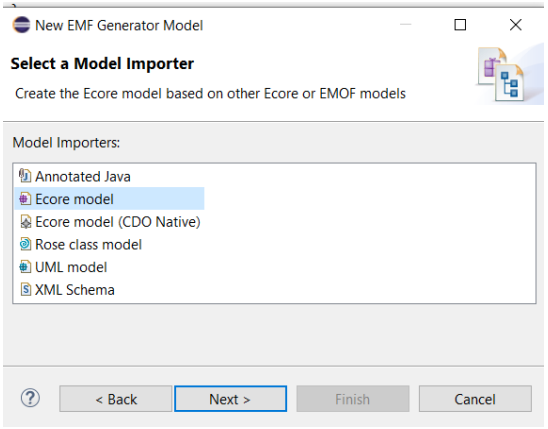

---

4. Fill the *File name* field as suggested. Click *Next*



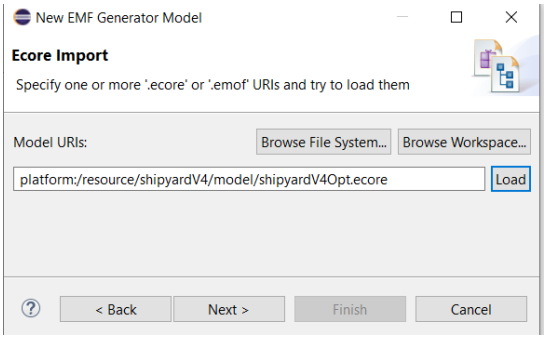
---

5. Choose *Ecore model* and click *Next*



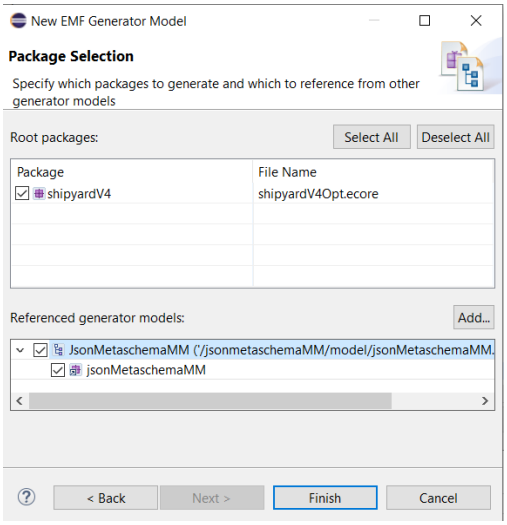

---

6. Click *Load* and then *Next*



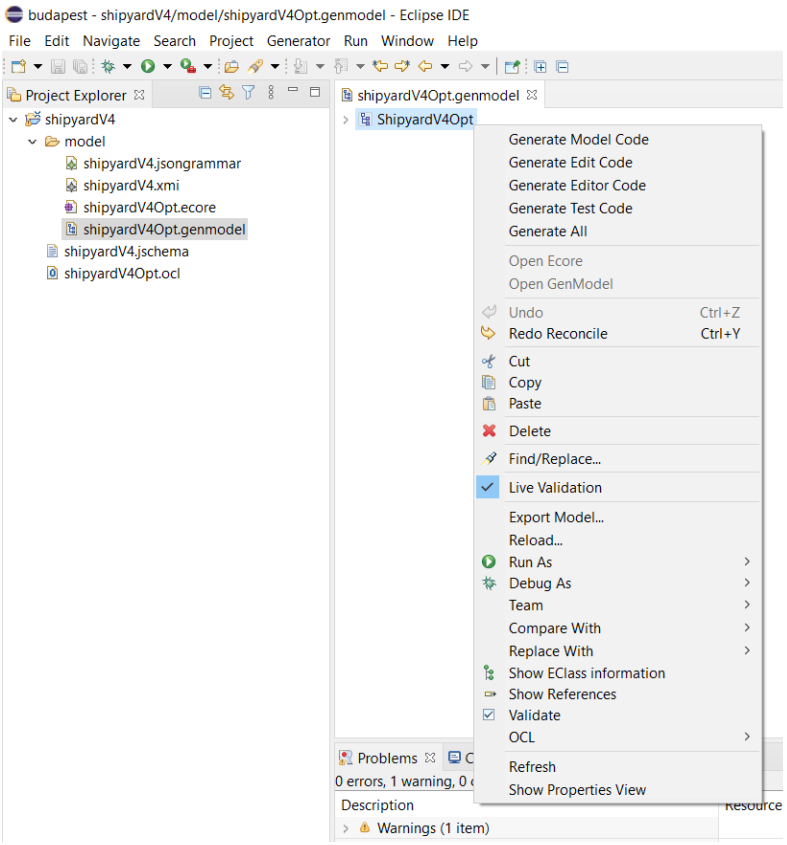

---

7. Select packages (tick checkboxes) as in the figure and click *Finish*



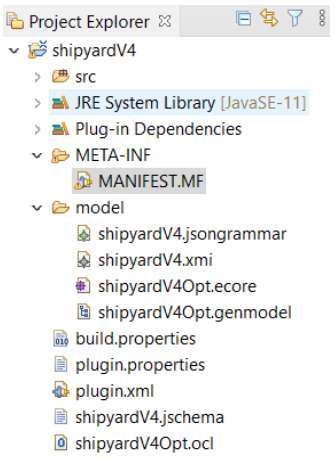

---

8. A file with extension *.genmodel* will be generated under the folder model. Open it. A tree will be displayed. Right click it and then select *Generate Model Code*




---

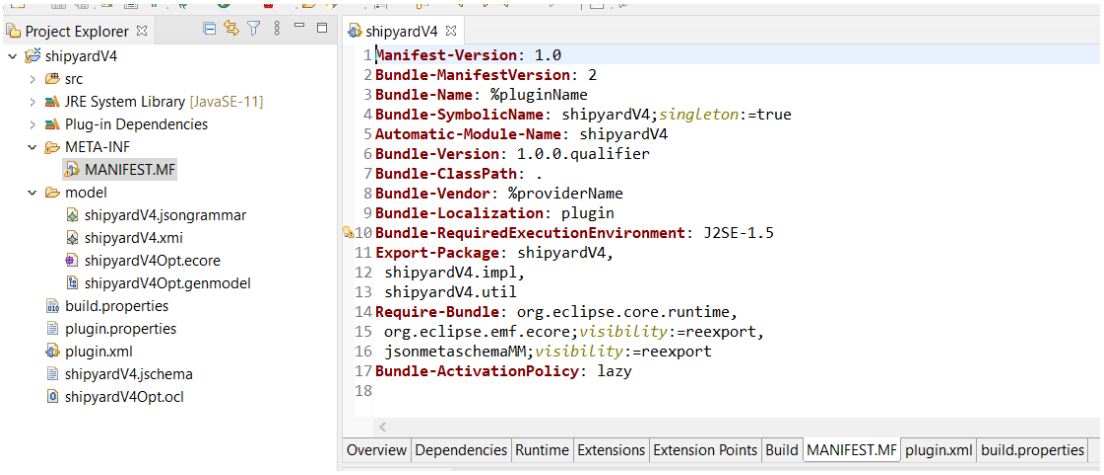
9. A new folder src with code has been generated. The eclipse plugin infrastructure has been built.




---

10. Fix MANIFEST.MF warning

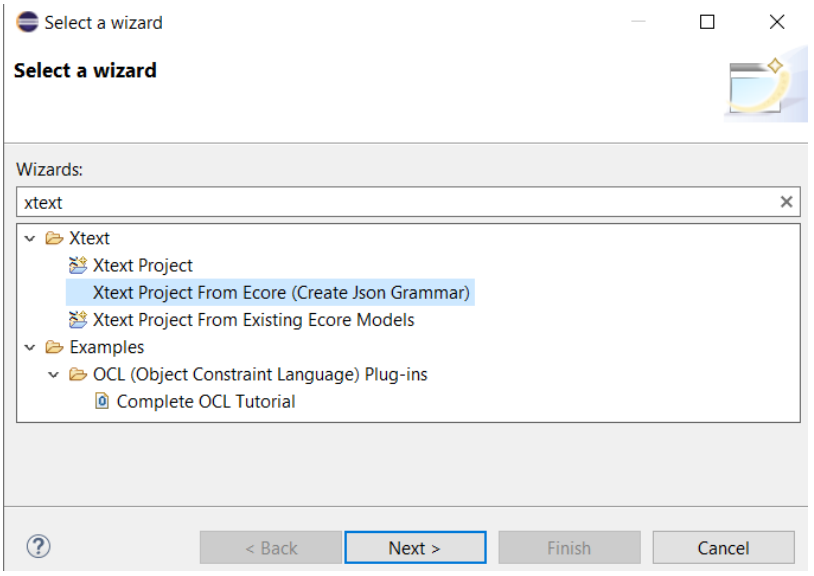
If MANIFEST.MF has been generated with “Bundle-RequiredExecutionEnvironment: J2SE-1.5” as in the picture, it has to be changed to use JavaSE-11. Please refer to In the installation tutorial step 3, to associate JavaSE-11 to jdk 11 in the execution environment.




---

11. **Use JsonGrammar plugin**

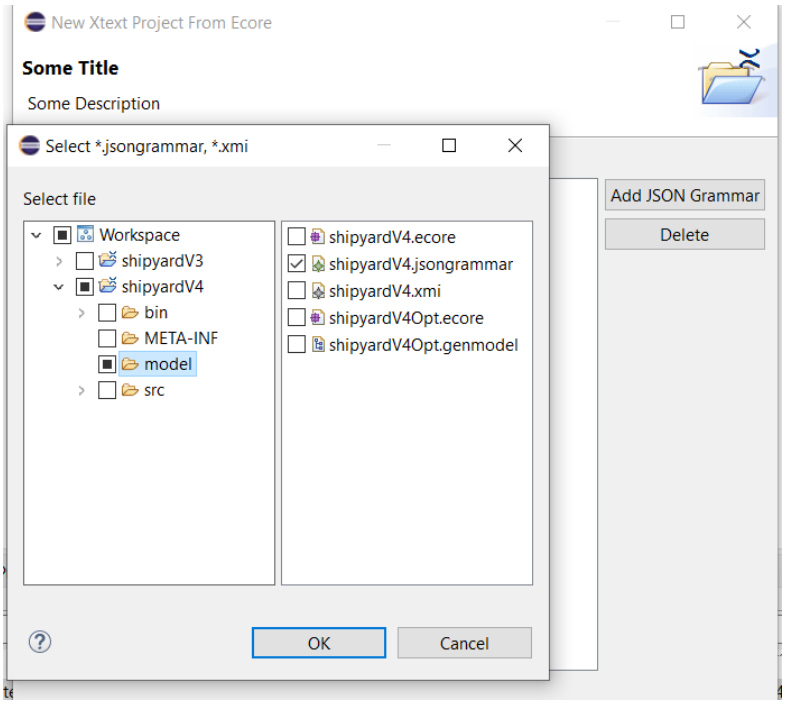
- Click on *File -> New ->Other* in the Eclipse top menu.
- Choose *Xtext Project From Ecore (Create json Grammar)*.
- Click *Next*.




---

12. **JsonGrammar model selection**

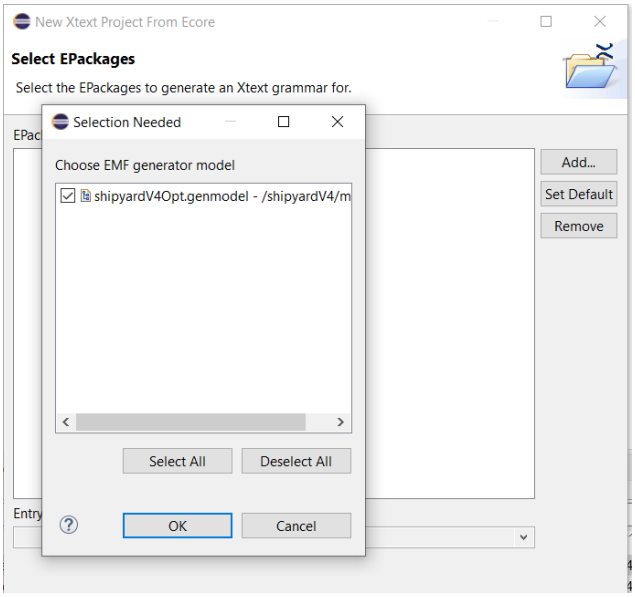
- Click Add JSON Grammar,
- Select the file *.jsongrammar* under the folder model of your project as in the screenshot.
- Click *OK* . Then Click next




---

13. **Genmodel selection**

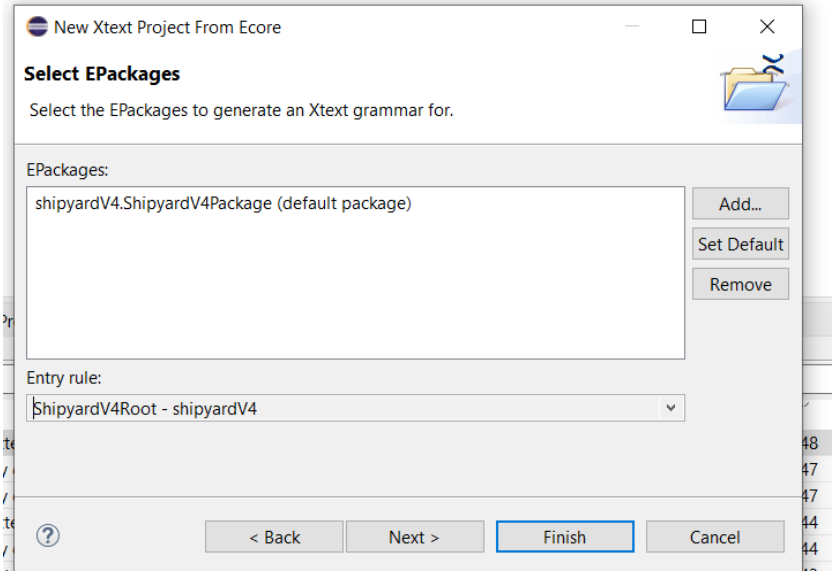
- Click Add
- Select the file with extension *.genmodel* generated in the previous wizard.
- Click *OK*.




---

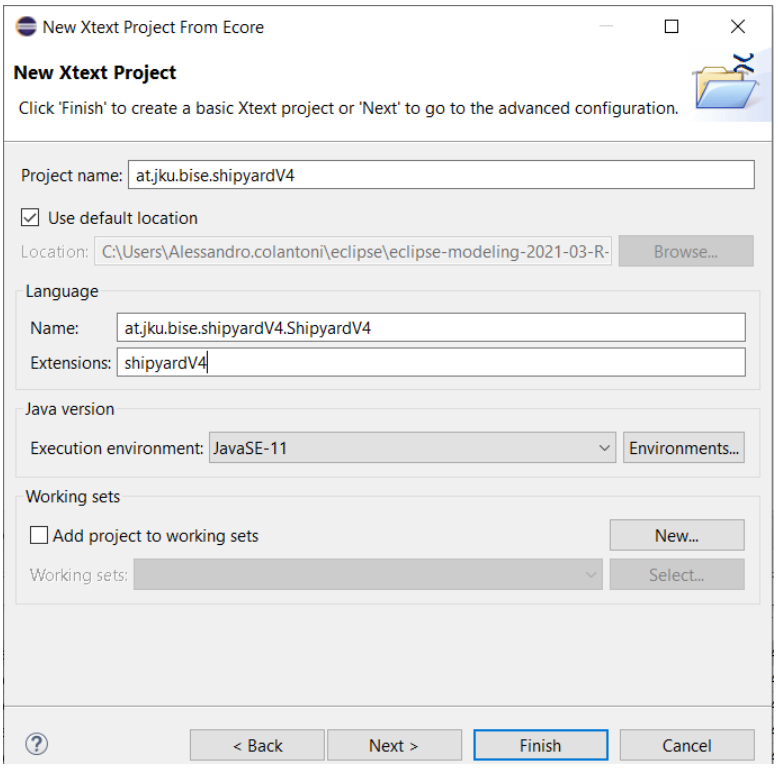
14. **Root selection**

- In the *Entry rule* select the root of your metamodel. It has the name of your project plus *Root* Suffix (e.g. ShipyardV4Root).
- Click *Next*. **DO NOT** click Finish.



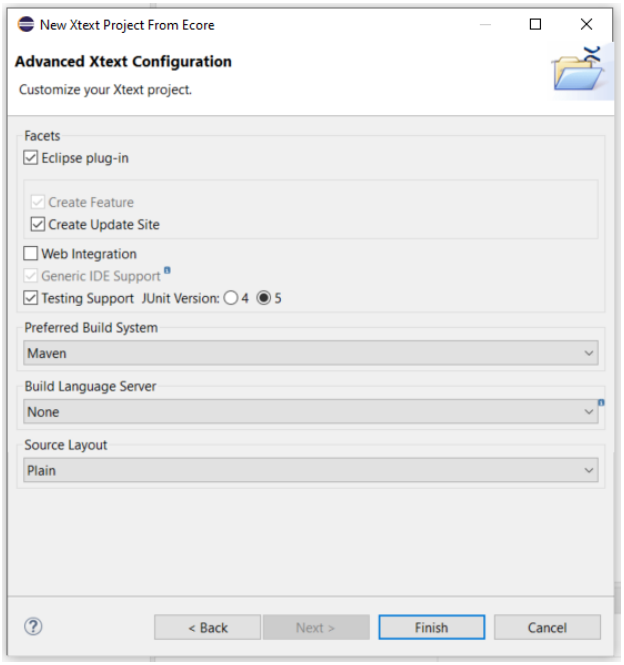

---

15. Choose the proper name for your Editor. Above all take care at the *Extensions* field: This will be the extensions of the files of the language under generation.
Make sure to use JAVA 11.0.10 or newer.
Click *Next*, **DO NOT** click *Finish*.



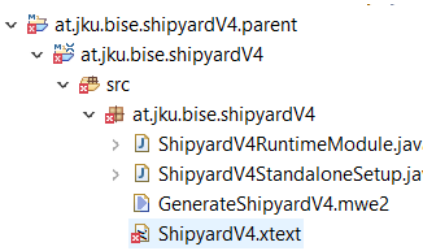
---

16. Fill the Fields as in the picture. Click *Finish*. Wait until Eclipse has finished its job.



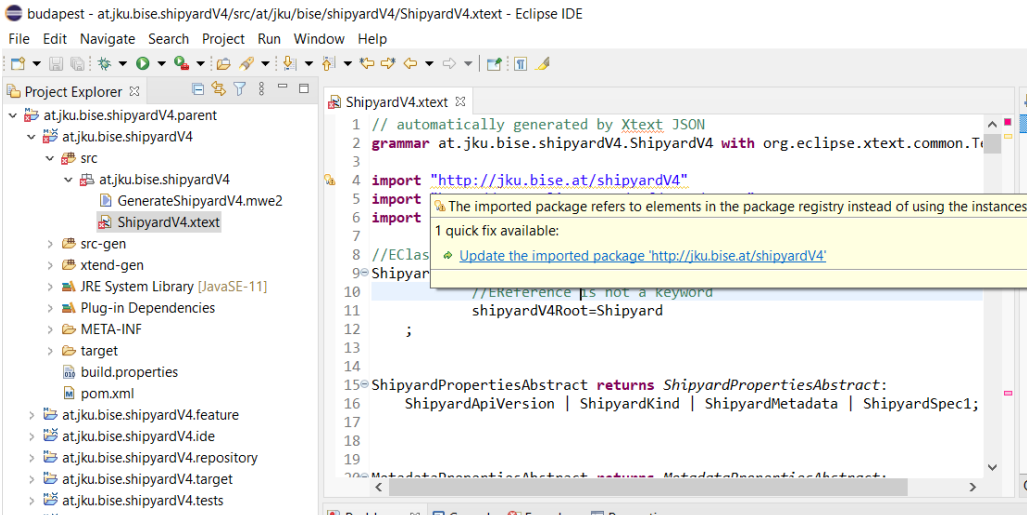

---

17. You will see a new project created. Open its folders as in the picture until you find a file with extension *.xtext*.




---

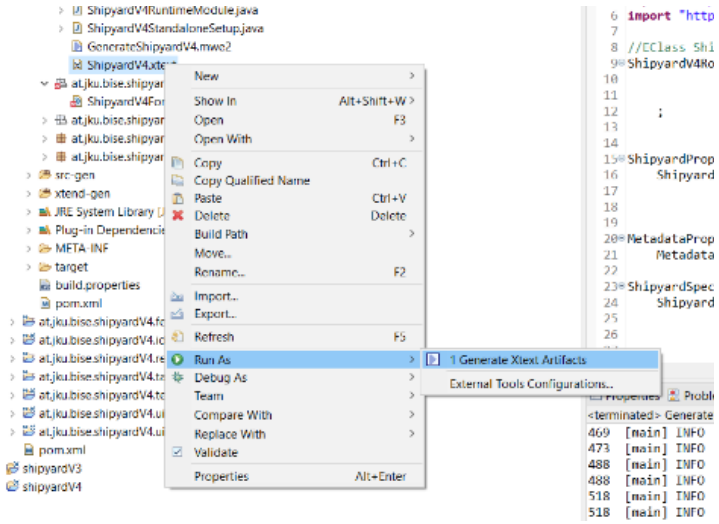
18. Open the file (ShipyardV4.xtext). Probably you will see a warning like in the picture that causes an error in the same file.
Move your mouse over the warning and apply the suggestion. Save the file if required.




---

19. **Generate Xtext artifacts**

- Right click the file with extension *.xtext* (ShipyardV4.xtext)
- Click *Run As*
- Click on : *1. Generate Xext Artifacts*

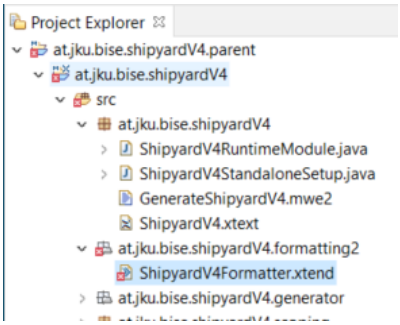



In the console the following INFO will be printed

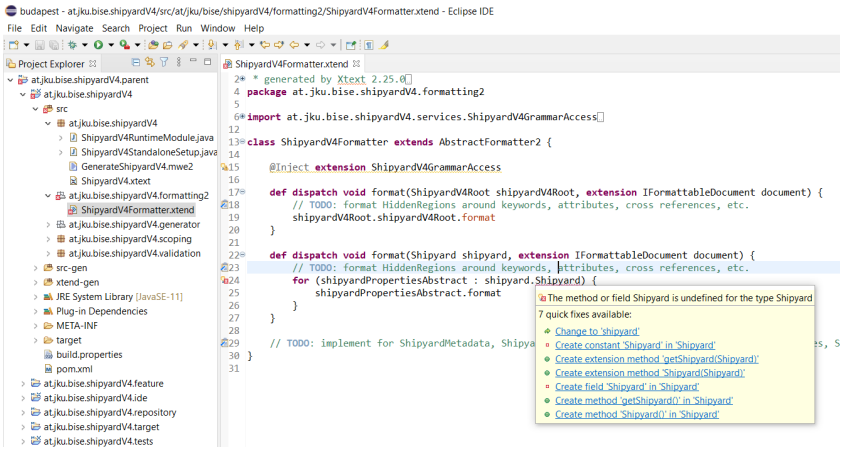
```
Problems  Javadoc  Declaration  Console
<terminated> GenerateShipyardV4.mwe2 [Mwe2 Launch] C:\Program Files\Java\jdk-11.0.1\bin\javaw.exe  (9 may. 2021 22:22:44 – 22:22:53)
0    [main] INFO  text.xtext.generator.XtextGenerator  - Initializing Xtext generator
12   [main] INFO  lipse.emf.mwe.utils.StandaloneSetup  - Adding generated EPackage 'org.eclipse.xtext.common.types.TypesPackage'
261  [main] INFO  lipse.emf.mwe.utils.StandaloneSetup  - Registering project at.jku.bise.shipyardV4 at 'file:/C:/Users/Alessandro.cola
263  [main] INFO  lipse.emf.mwe.utils.StandaloneSetup  - Registering project at.jku.bise.shipyardV4.tests at 'file:/C:/Users/Alessandr
265  [main] INFO  lipse.emf.mwe.utils.StandaloneSetup  - Registering project at.jku.bise.shipyardV4.ide at 'file:/C:/Users/Alessandro.
266  [main] INFO  lipse.emf.mwe.utils.StandaloneSetup  - Registering project at.jku.bise.shipyardV4.ui at 'file:/C:/Users/Alessandro.c
267  [main] INFO  lipse.emf.mwe.utils.StandaloneSetup  - Registering project at.jku.bise.shipyardV4.ui.tests at 'file:/C:/Users/Alessa
276  [main] INFO  lipse.emf.mwe.utils.StandaloneSetup  - Using resourceSet registry. The registered Packages will not be registered in
549  [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'http://www.eclipse.org/Xtext/Xbase/XAnnotations' from 'p
552  [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'http://www.eclipse.org/xtext/xbase/Xtype' from 'platform
564  [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'http://www.eclipse.org/xtext/xbase/Xbase' from 'platform
564  [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'http://www.eclipse.org/xtext/common/JavaVMTypes' from 'p
596  [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'http://jku.bise.at/shipyardV4' from 'platform:/resource/
596  [main] INFO  clipse.emf.mwe.utils.GenModelHelper  - Registered GenModel 'http://at.jku.bise/jsonMetaschemaMM' from 'platform:/res
955  [main] INFO  text.xtext.generator.XtextGenerator  - Generating at.jku.bise.shipyardV4.ShipyardV4
5650 [main] INFO  text.xtext.generator.XtextGenerator  - Generating common infrastructure
5679 [main] INFO  .emf.mwe2.runtime.workflow.Workflow  - Done.
```

---

20. Open the file <*YourProjectName*>*Formatter.xtend* (ShipyardV4Formatter.xtend).



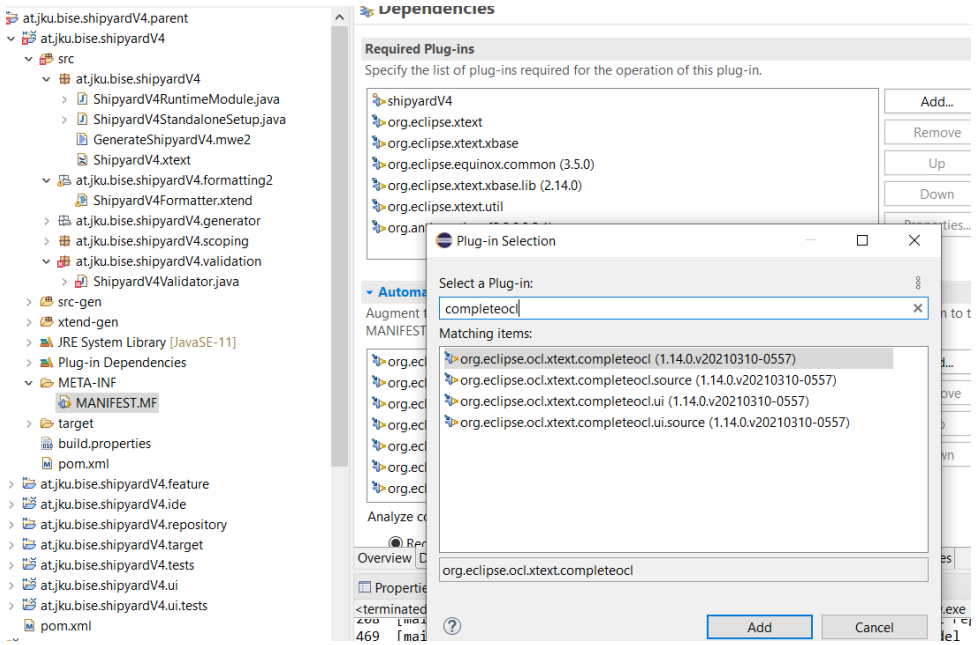

If there is a compilation error like in the screenshot, move your mouse over it and apply the suggestion to change to a name with the first character in lower case, as shown in the screenshot below.




---

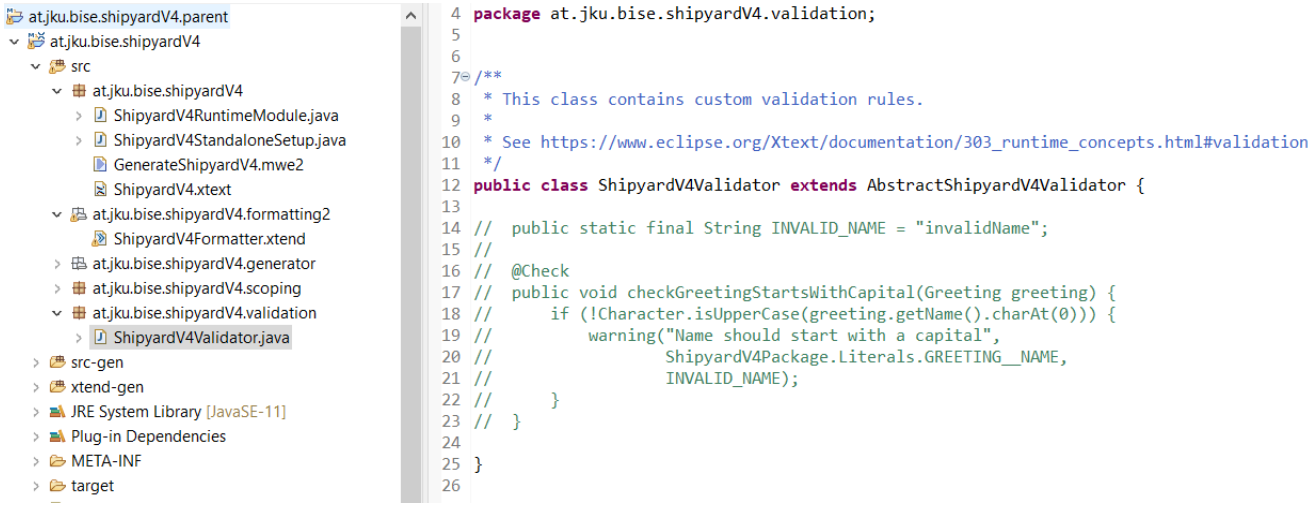
21. **Add OCL dependency**

- Open the file MANIFEST.MF like in the picture
- Select the *Dependencies* tab.
- Click *Add*.
- Select the plugin like in the picture.
- Click *Add*.
- Save the file.




---

22. Open *<YourProjectName>Validator.java* like in the screenshot below (e.g., ShipyardV4Validator.java).




---

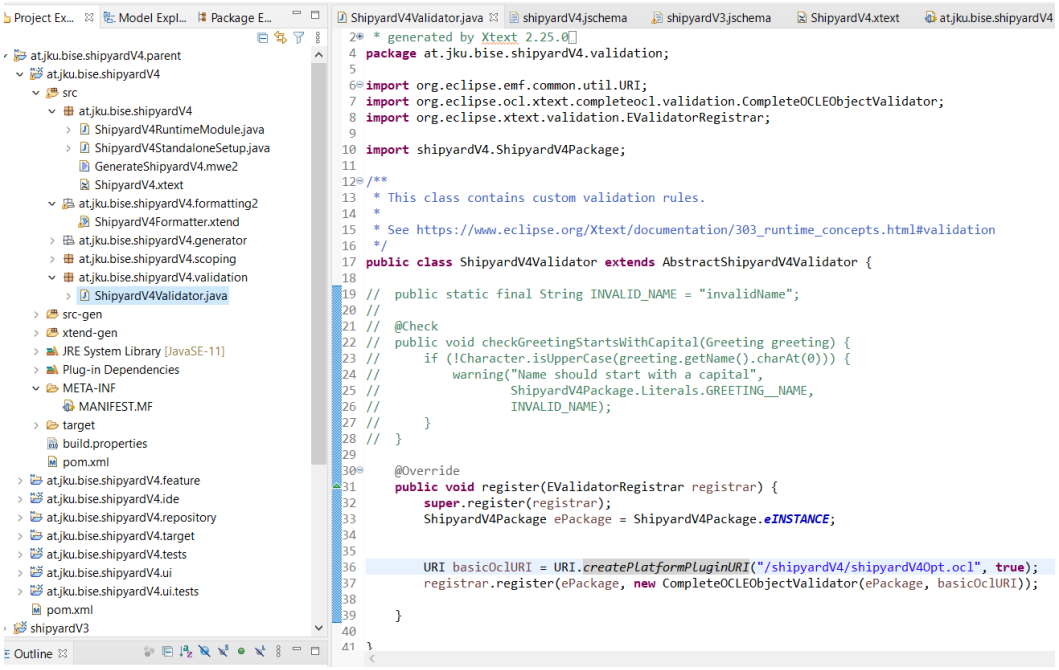
23. Add the following code snippet:

```
public void register(EValidatorRegistrar registrar) {
	super.register(registrar);
	ShipyardV4Package ePackage = ShipyardV4Package.eINSTANCE;


	URI basicOclURI = URI.createPlatformPluginURI("/shipyardV4/shipyardV4Opt.ocl", true);
	registrar.register(ePackage, new CompleteOCLEObjectValidator(ePackage, basicOclURI));

		}
```

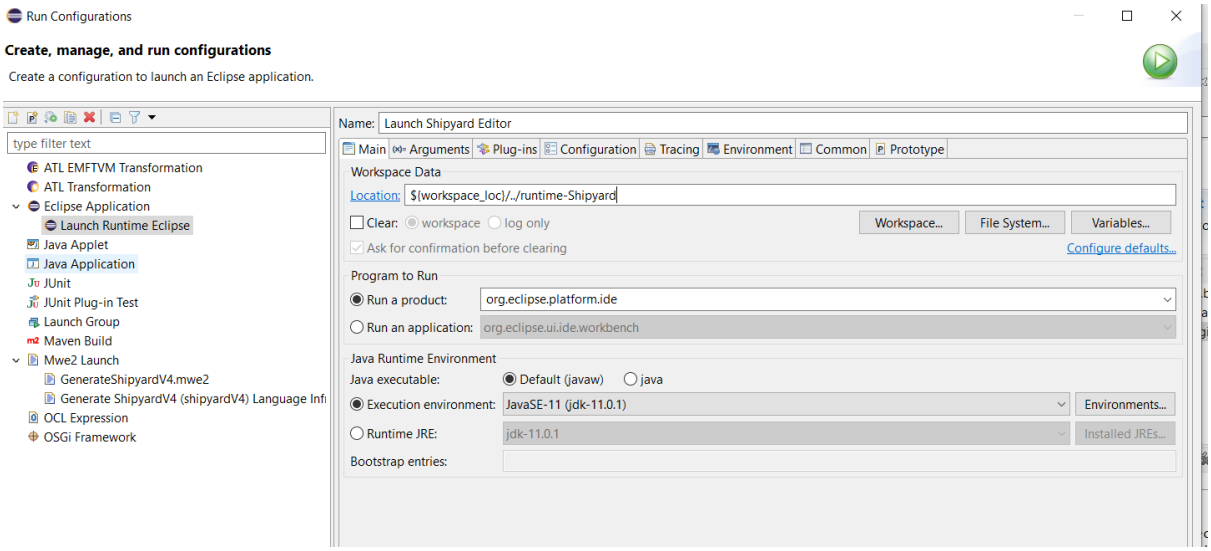
This code is for the case of a project named shipyardV4. If you are not following the shipyardV4 example you must change *"/shipyardV4/shipyardV4Opt.ocl"* for the path to the ocl generated from your *.jschema*.
Take care of the imports. The result of the whole class should be something like the picture below. Above all there are many choices for importing URI class. The right import is: org.eclipse.emf.common.util.URI;




---

24. Execute a project clean build.
25. Regenerate the xtext artifacts
26. Open the Java Perspective.

27. In the eclipse top menu click *Run Configurations*, and create your *language editor launcher* like in the picture.



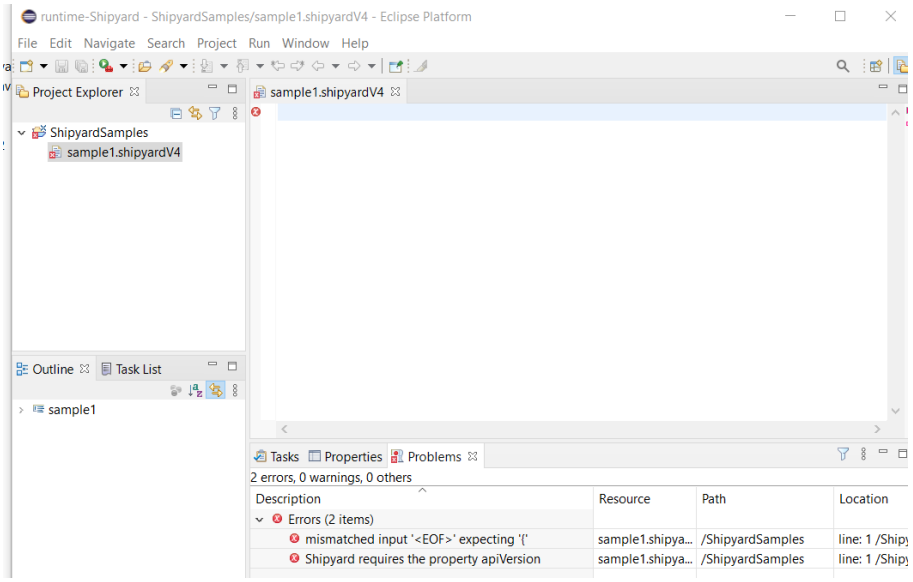

28. Launch the created Runtime eclipse. Wait that a new eclipse is launched

---

## Usage of the newly created json based grammar.

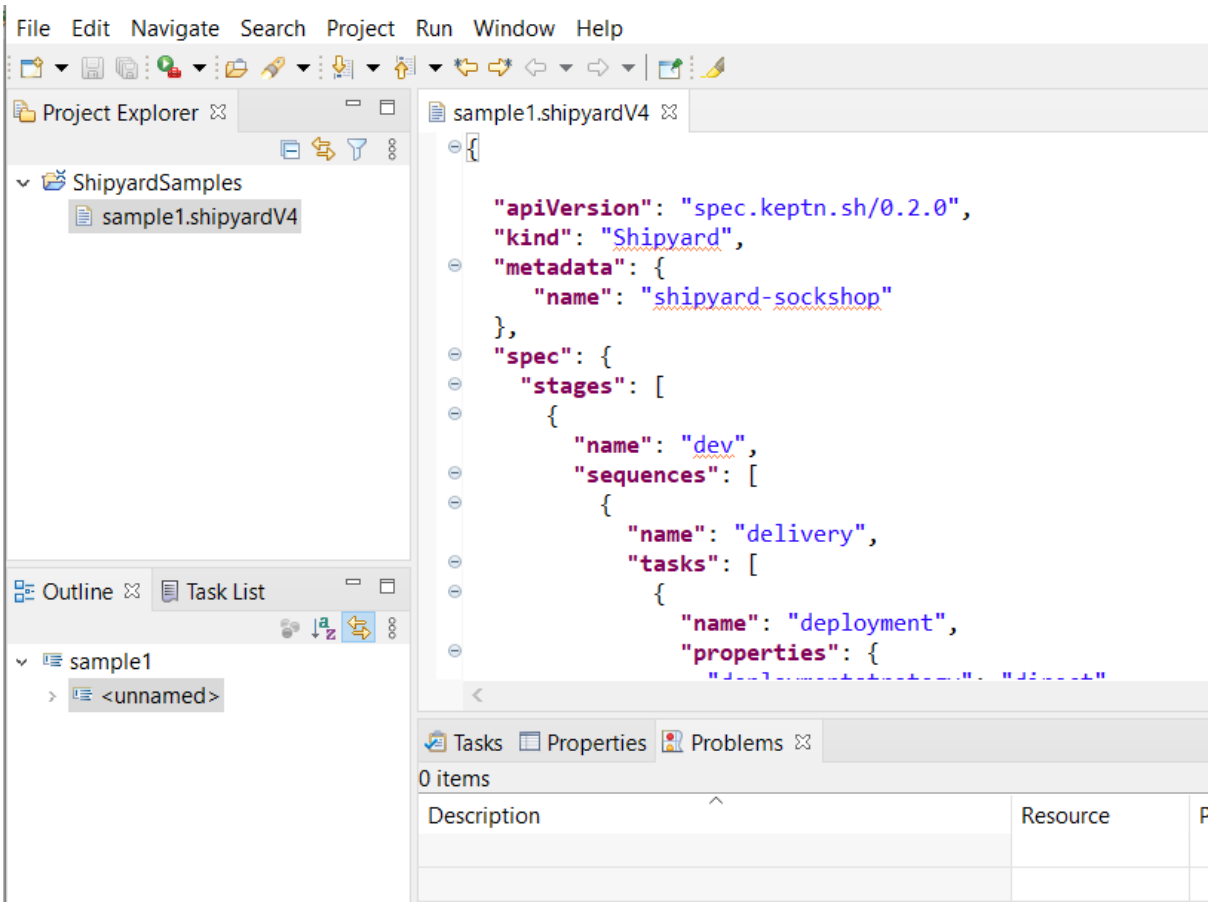
29. In the launched Runtime Eclipse

- create a new general project with the New Project wizard;
- create a file with the extension you chose in the language editor creation(e.g., shipyardV4).
- You will be asked if you want to convert the project to an XText project. Answer YES.
- Open your file
- Open the *Problems* and *Properties* view to see more details.
- Use CTRL+space for the content assist and code completion. Remember that all the keywords are between double quotes (“)



30. **Use our examples**.

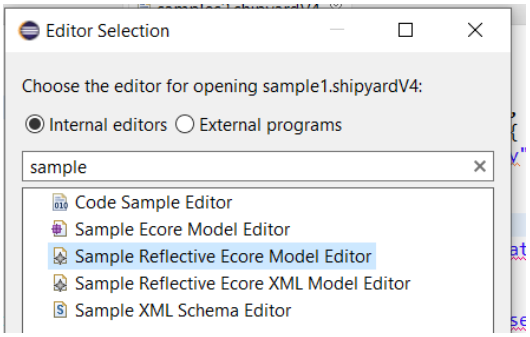
For the shipyardV4.jschema example used in this tutorial you can try json instance examples in
https://github.com/lowcomote/jsonschemadsl.parent/tree/1.0.1/samples/shipyardSchemas/shipyardV4/instanceExamples
where you will find sample1.shipyardV4 and samples2.shipyardV4.
In the screenshot below you can see an excerpt of sample1.shipyardV4.
Open the *Problems* and *Properties* view to see more details.




---

31. **Sample Reflective Ecore Model Editor**

- Right click on the file with the extension that you have created (e.g., sample1.shipyardV4)
- Select *open with -> other*.
- and you will see the tree editor for your file, that is a model.




- Open the view properties, to see more details when you select an element of the tree.

As shown in the screenshot below, you can see sample1.shipyardV4 as a json conforming to the shipyardV4.jschema or as a model conforming to the previously generated shipyardV4Opt.ecore.
Changes applied to the tree are reflected in the file json style and vice versa.