

Eclipse APP4MC 2.2



NEW AND NOTEWORTHY

APP4MC 2.2 (Jul 2022)

General description of platform changes

Model extensions

- ▶ New utility functions (Stimuli event model)

Product

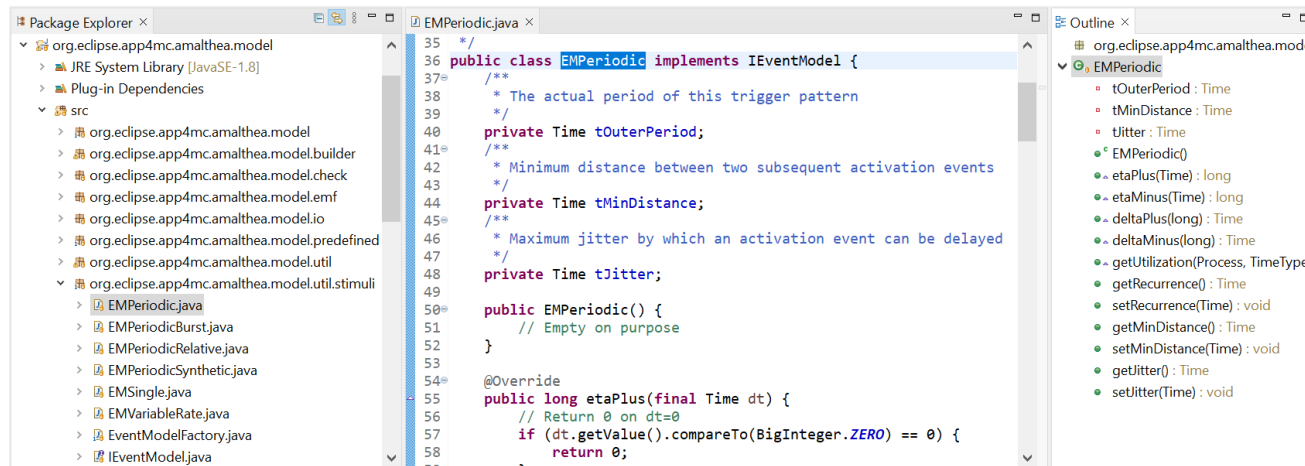
- ▶ Updated Runnable dependencies visualization
- ▶ Validations:
 - New APP4MC.sim validations
 - Duplicate ID check moved to EMF invariants (available as live validation)
- ▶ Extended help: Installable platform extensions

APP4MC 2.2

Utility functions - Stimuli event model

Utility functions - Stimuli event model

The package provides event models corresponding to typical activation patterns in order to enable the application of common performance analysis techniques.



```
35  */
36  public class EMPeriodic implements IEventModel {
37      /**
38       * The actual period of this trigger pattern
39       */
40      private Time tOuterPeriod;
41      /**
42       * Minimum distance between two subsequent activation events
43       */
44      private Time tMinDistance;
45      /**
46       * Maximum jitter by which an activation event can be delayed
47       */
48      private Time tJitter;
49
50      public EMPeriodic() {
51          // Empty on purpose
52      }
53
54      @Override
55      public long etaPlus(final Time dt) {
56          // Return 0 on dt=0
57          if (dt.getValue().compareTo(BigInteger.ZERO) == 0) {
58              return 0;
59          }
60      }
61  }
```

The screenshot shows the Eclipse IDE interface. On the left is the Package Explorer showing the project structure for 'org.eclipse.app4mc.amalthea.model'. The central editor displays the source code for 'EMPeriodic.java', which implements the 'IEventModel' interface. The code defines private fields for 'tOuterPeriod', 'tMinDistance', and 'tJitter', and includes a 'public EMPeriodic()' constructor and an '@Override' method 'etaPlus'. On the right, the Outline view shows the class hierarchy and the methods defined in 'EMPeriodic'.

Further information

- Publication: Complex event models for automotive embedded systems
<https://doi.org/10.1016/j.sysarc.2021.102343>
- Presentation: Timing Analysis support for complex trigger patterns
<https://panorama-research.org/news/2021/12/22/timing-analysis-support-for-complex-trigger-patterns/>

New IDE extension examples

IDE extension examples

introduced with
APP4MC 2.1

New examples for **developers** who want to extend the IDE

- Editor actions
- Validations
- Visualizations

Steps to use the examples

- Start the APP4MC product
- Install additional plugin from APP4MC update site
- Create new examples in the current workspace
- Run an Eclipse Application (runtime instance) to see the effects

IDE extensions documentation

added with
APP4MC 2.2

- APP4MC Documentation
 - Introduction to APP4MC
 - User Guide
 - Data Models
 - Developer Guide
 - Platform Extensions**
 - Overview
 - EMF Model Viewers
 - Additional Examples
 - Model Transformation
 - SystemC Timing Simulation
 - Release Notes
 - Roadmap

[APP4MC Documentation](#) > [Platform Extensions](#)

Overview

There are additional features of the APP4MC platform that are not included in the standard product. They can be installed in the current environment via "Install new software...".

Available Extensions

EMF Viewers

Additional examples how to extend the IDE

- Custom visualizations
- Custom validations
- Custom editor actions

Model Transformation

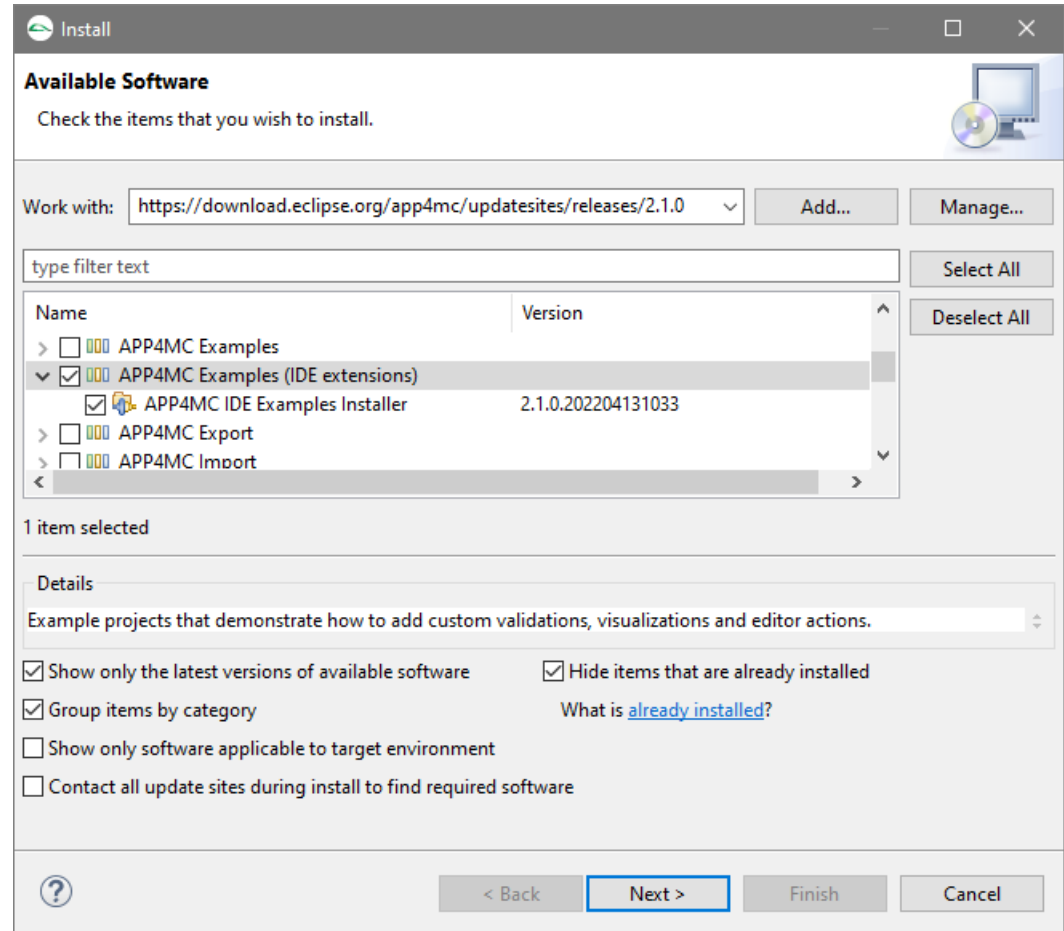
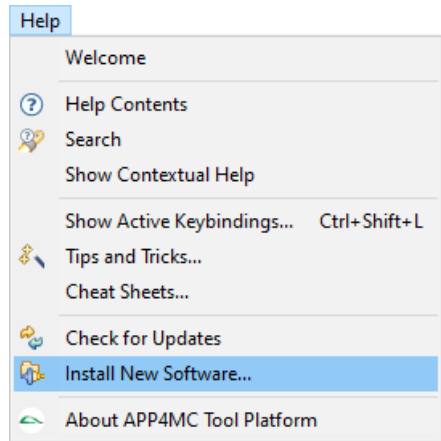
- Framework and examples
- Transformation of AMALTHEA to Linux code
- Transformation of AMALTHEA to ROS2 code
- Transformation of AMALTHEA to SystemC code

SystemC Timing Simulation

<https://www.eclipse.org/app4mc/help/latest/index.html>

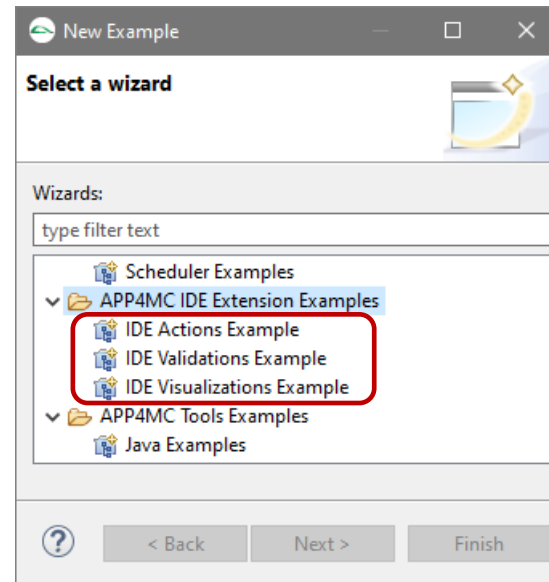
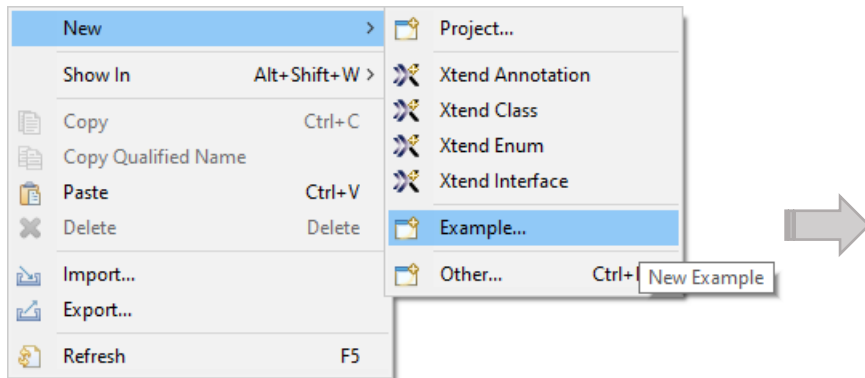
IDE extension examples

Add IDE examples installer



IDE extension examples

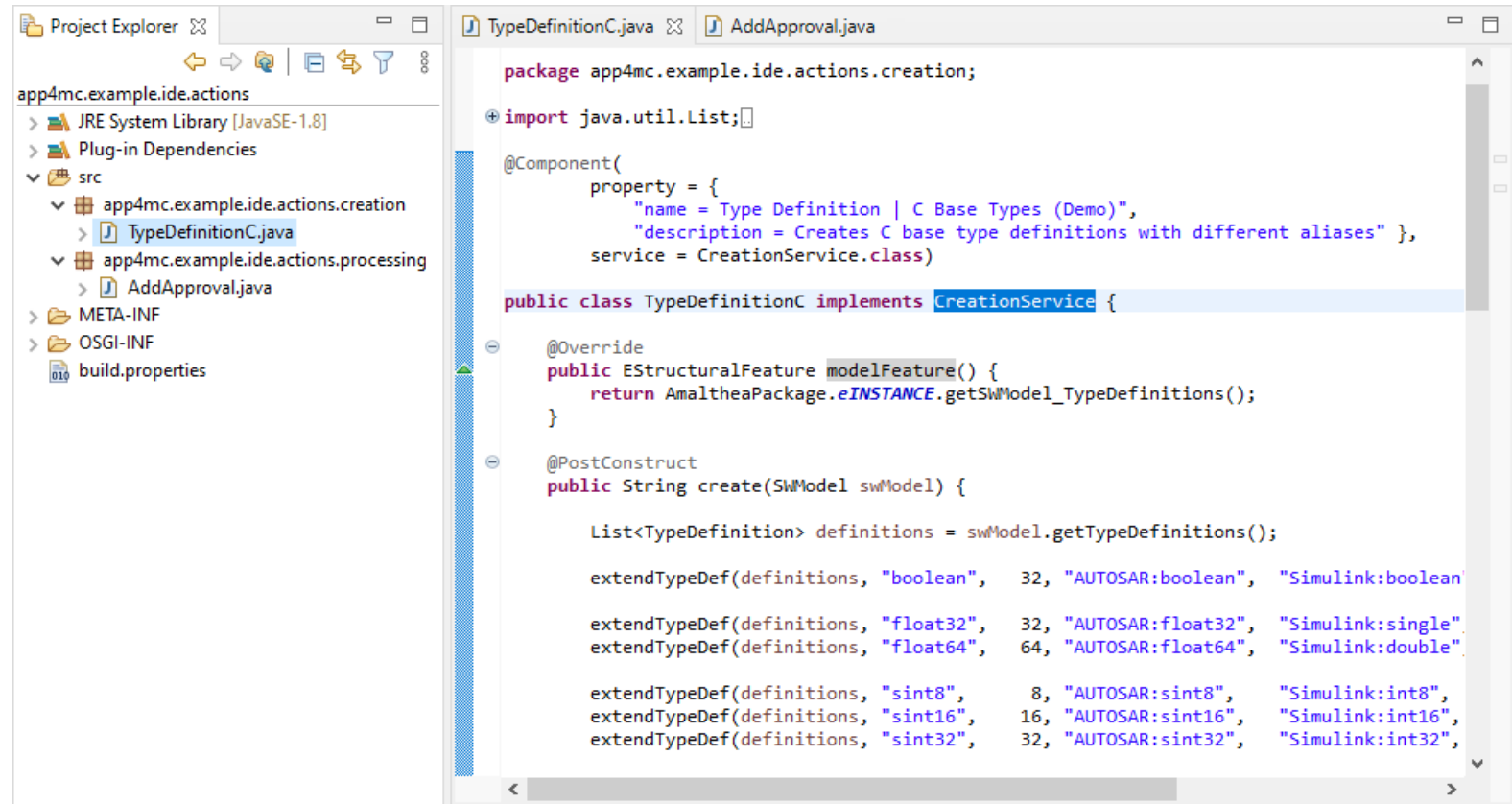
Create IDE examples



IDE extension examples

Editor action example

- Creation



The screenshot displays the Eclipse IDE interface. On the left, the Project Explorer shows a project named 'app4mc.example.ide.actions' with a source folder 'src'. Inside 'src', there are two packages: 'app4mc.example.ide.actions.creation' and 'app4mc.example.ide.actions.processing'. The 'creation' package contains two files: 'TypeDefinitionC.java' and 'AddApproval.java'. The 'processing' package contains 'AddApproval.java'. The main editor window shows the code for 'TypeDefinitionC.java'. The code includes a package declaration, an import for 'java.util.List', an '@Component' annotation with a 'property' block, and a 'public class TypeDefinitionC implements CreationService' declaration. The class contains an '@Override' method 'modelFeature()' and a '@PostConstruct' method 'create(SwModel swModel)'. The 'create' method uses 'swModel.getTypeDefinitions()' to get a list of definitions and then calls 'extendTypeDef' multiple times to register different data types with their corresponding AUTOSAR and Simulink aliases.

```
package app4mc.example.ide.actions.creation;

import java.util.List;

@Component(
    property = {
        "name = Type Definition | C Base Types (Demo)",
        "description = Creates C base type definitions with different aliases" },
    service = CreationService.class)

public class TypeDefinitionC implements CreationService {

    @Override
    public EStructuralFeature modelFeature() {
        return AmaltheaPackage.eINSTANCE.getSwModel_TypeDefinitions();
    }

    @PostConstruct
    public String create(SwModel swModel) {

        List<TypeDefinition> definitions = swModel.getTypeDefinitions();

        extendTypeDef(definitions, "boolean", 32, "AUTOSAR:boolean", "Simulink:boolean");

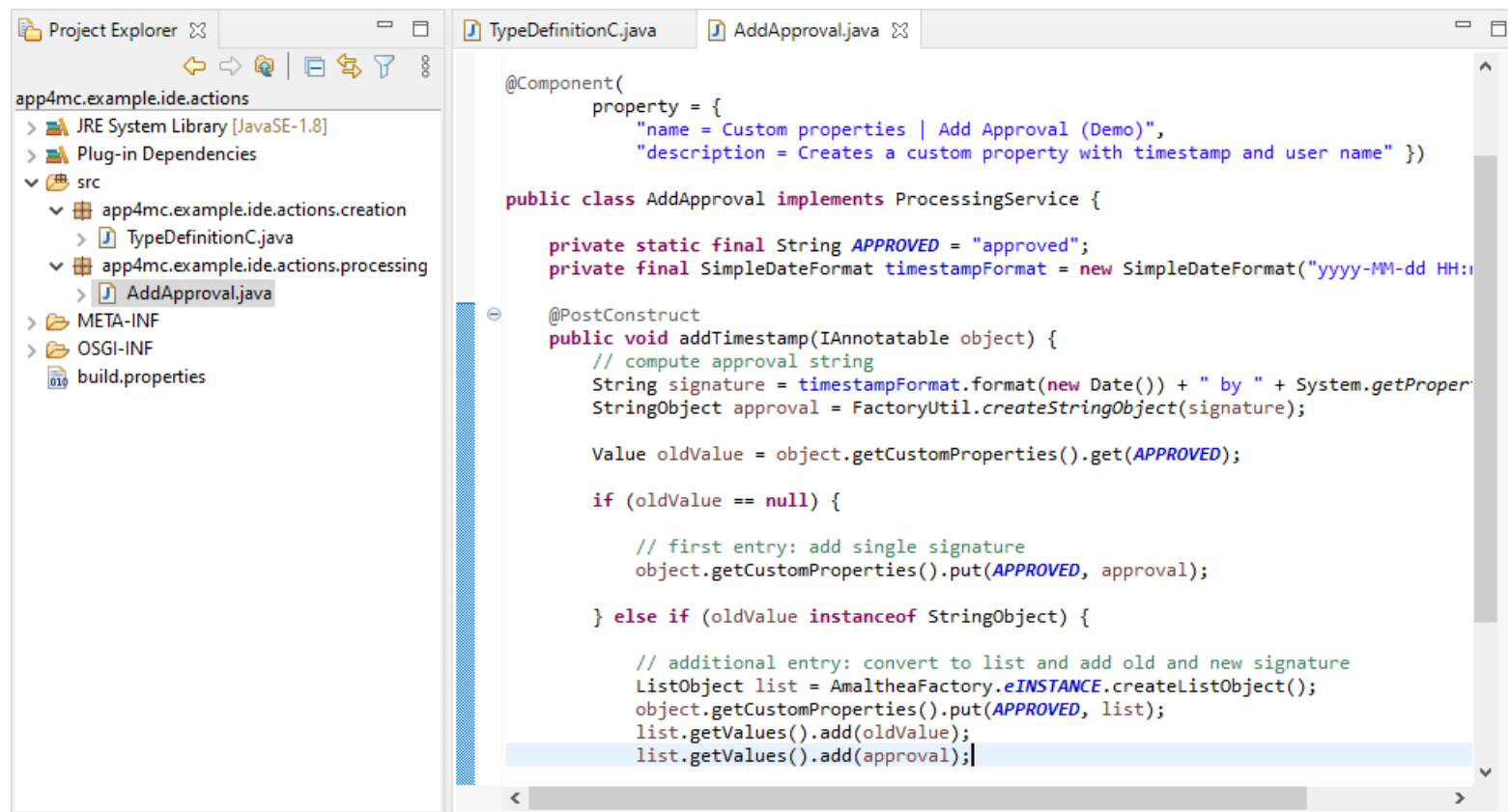
        extendTypeDef(definitions, "float32", 32, "AUTOSAR:float32", "Simulink:single");
        extendTypeDef(definitions, "float64", 64, "AUTOSAR:float64", "Simulink:double");

        extendTypeDef(definitions, "sint8", 8, "AUTOSAR:sint8", "Simulink:int8");
        extendTypeDef(definitions, "sint16", 16, "AUTOSAR:sint16", "Simulink:int16");
        extendTypeDef(definitions, "sint32", 32, "AUTOSAR:sint32", "Simulink:int32");
    }
}
```

IDE extension examples

Editor action example

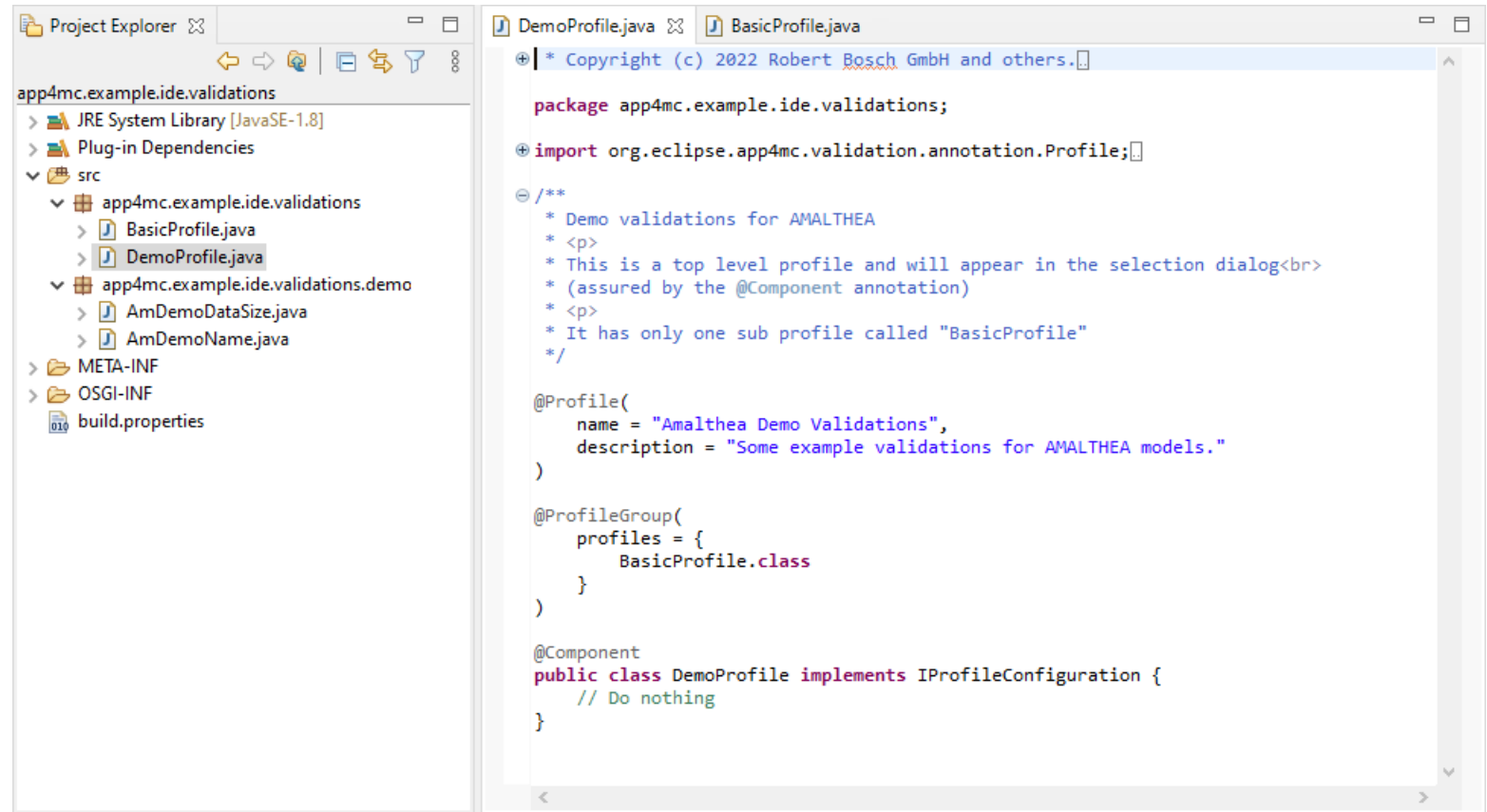
- Processing



IDE extension examples

Validation example

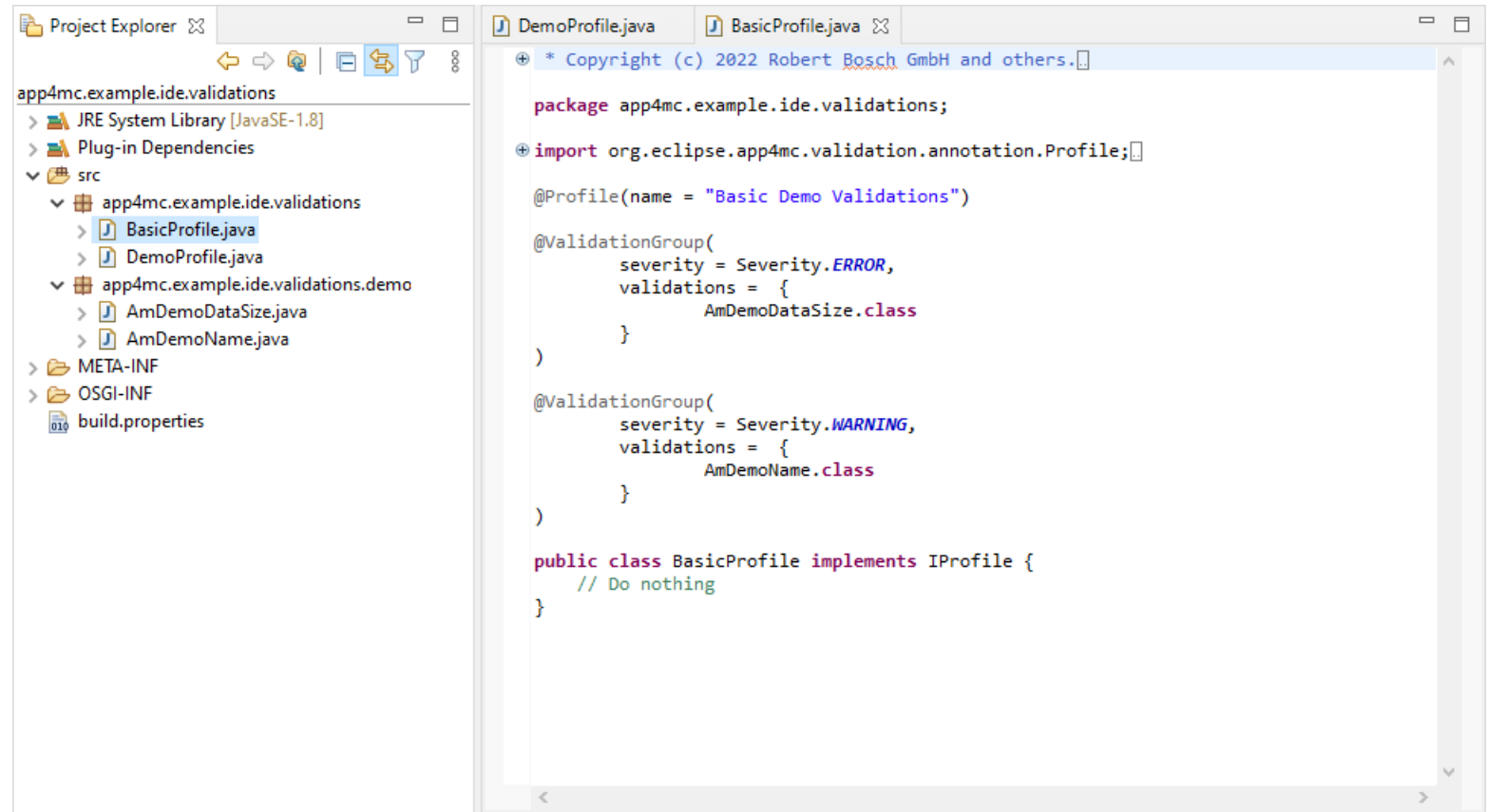
- Top level profile



IDE extension examples

Validation example

- Profile



IDE extension examples

Validation example

- Validation

```
Project Explorer
app4mc.example.ide.validations
  JRE System Library [JavaSE-1.8]
  Plug-in Dependencies
  src
    app4mc.example.ide.validations
      BasicProfile.java
      DemoProfile.java
    app4mc.example.ide.validations.de
      AmDemoDataSize.java
      AmDemoName.java
  META-INF
  OSGI-INF
  build.properties

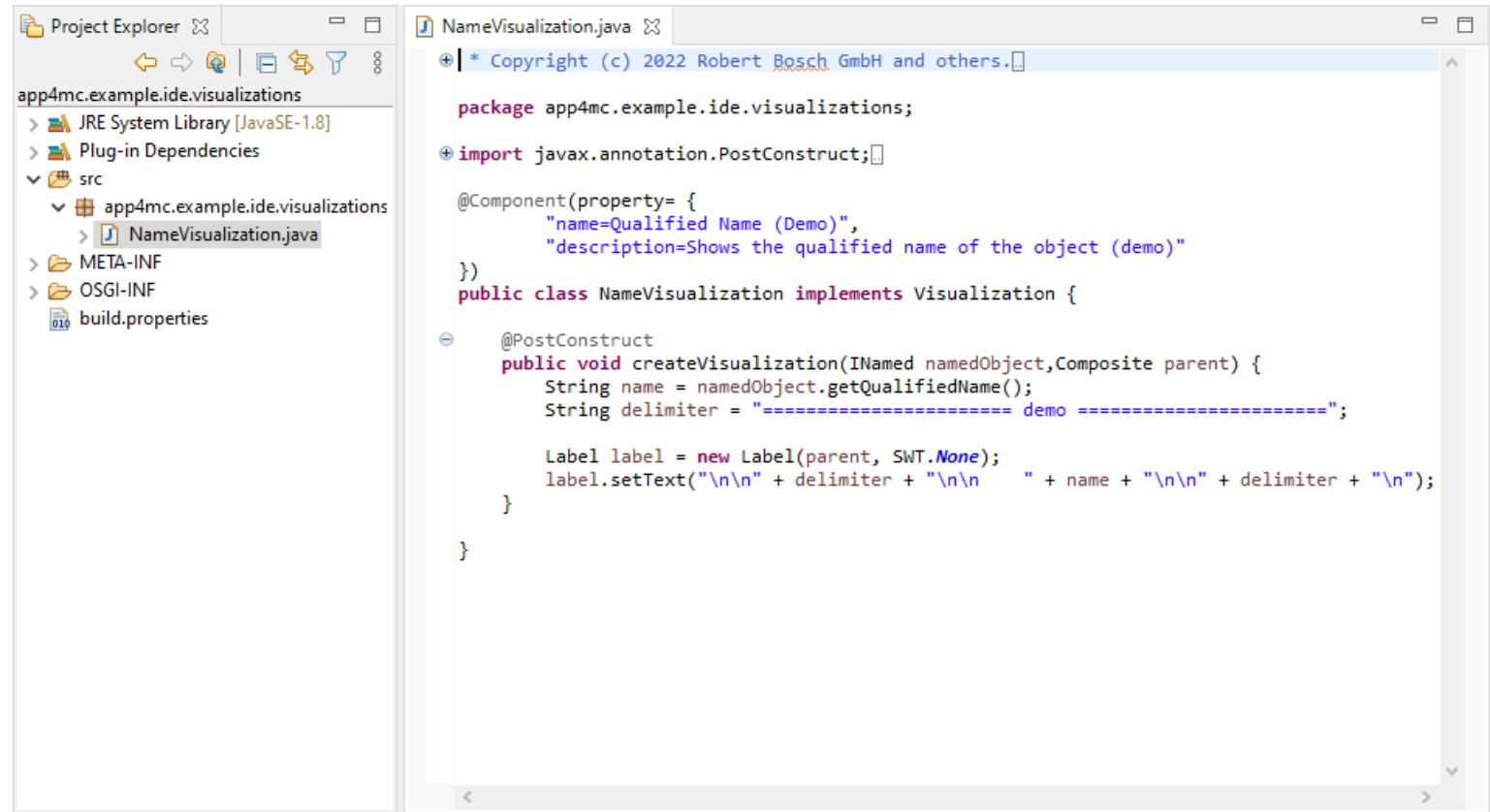
NameVisualization.java | AmDemoDataSize.java | AmDemoName.java
/**
 * Checks the correctness of names
 * <ul>
 * <li>Some names appear far down the alphabetical lists ;-)</li>
 * </ul>
 */
@Validation(
    id = "AM-Demo-Name",
    checks = { "Some names appear far down the alphabetical lists ;-)" })
public class AmDemoName extends AmaltheaValidation {
    @Override
    public EClassifier getEClassifier() {
        return ePackage.getINamed();
    }
    @Override
    public void validate(EObject eObject, List<ValidationDiagnostic> results) {
        if (eObject instanceof INamed) {
            final INamed namedObject = (INamed) eObject;
            String name = namedObject.getName();

            // name is null or empty -> checked by another validation
            if (name == null || name.isEmpty()) {
                return;
            }

            char firstChar = name.charAt(0);
```

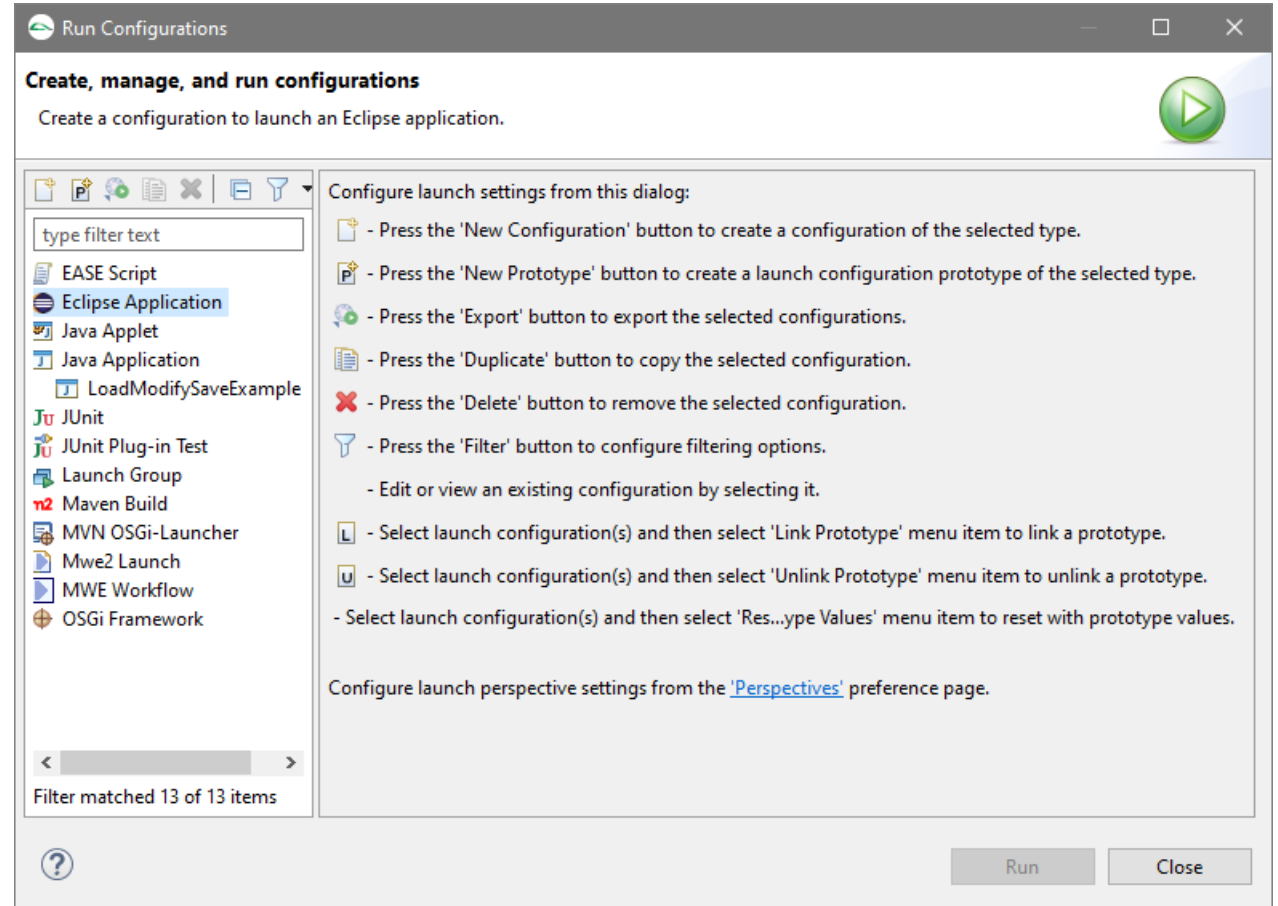
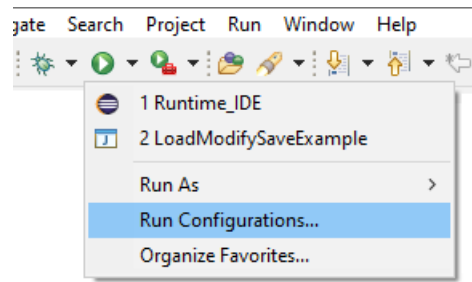
IDE extension examples

Visualization example



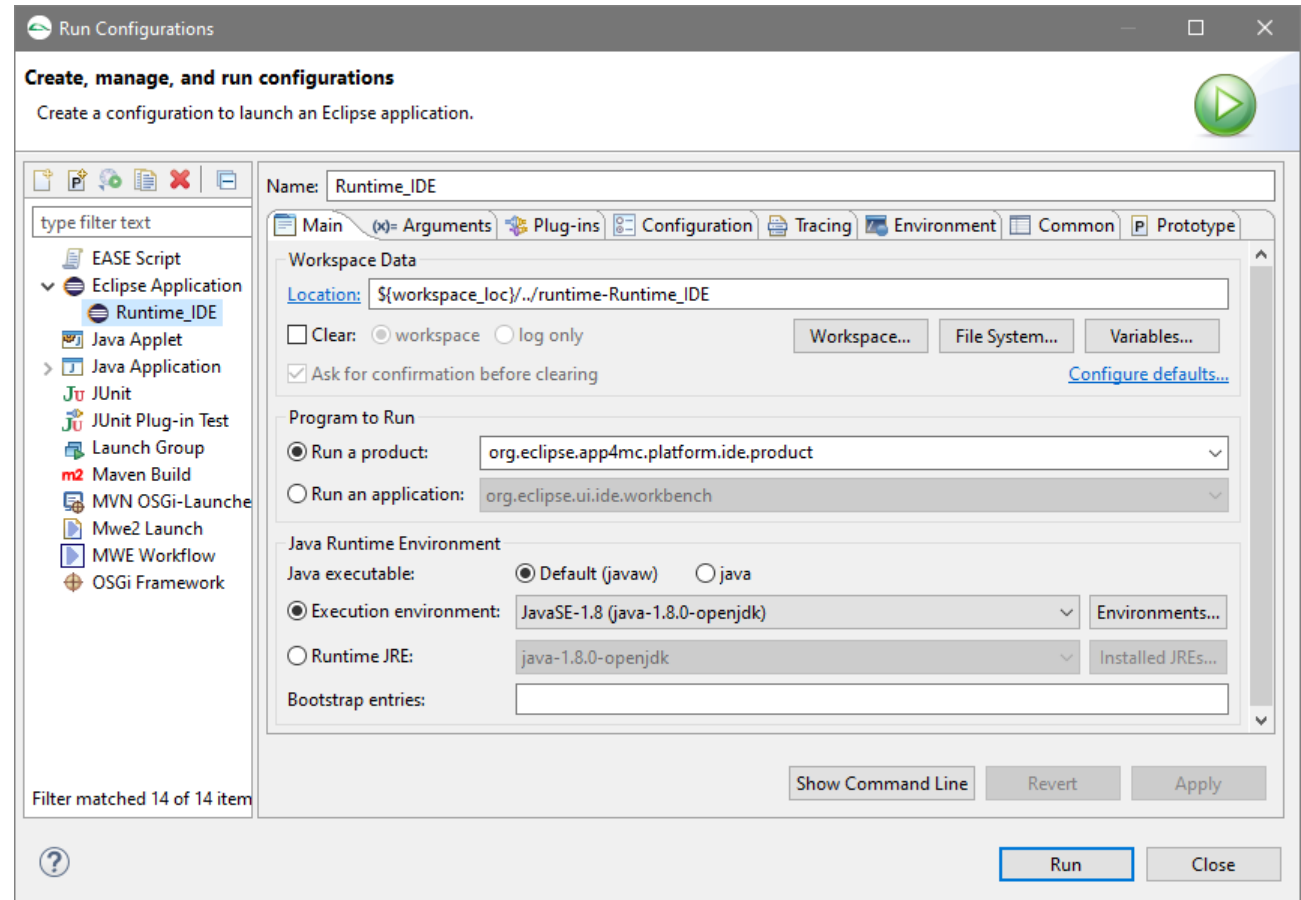
IDE extension examples

Run an Eclipse Application



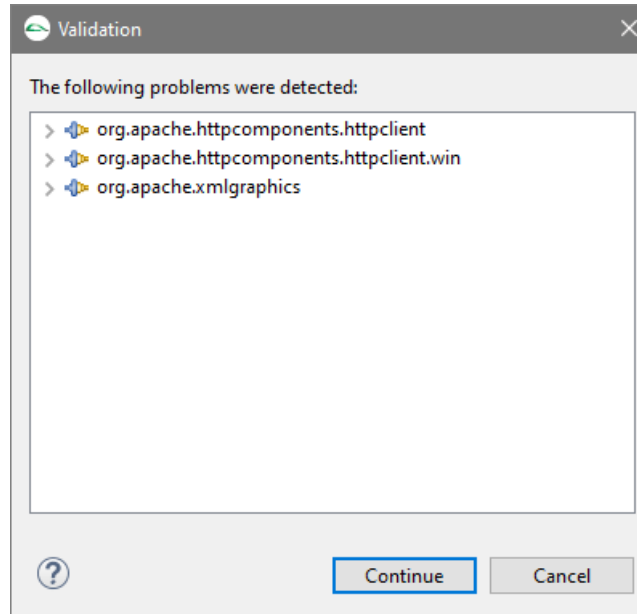
IDE extension examples

Run an Eclipse Application



IDE extension examples

Run an Eclipse Application



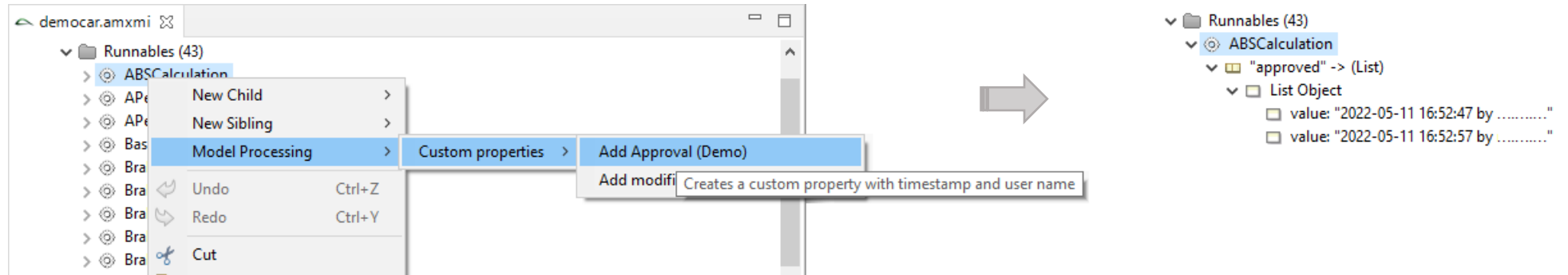
Ignore this validation

-> Continue

IDE extension examples - Runtime

Additional features:

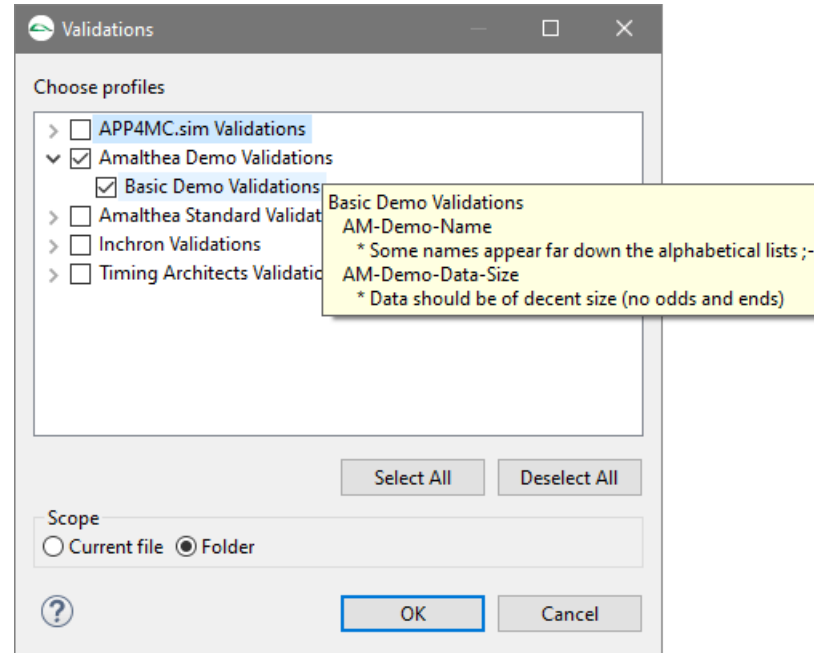
- New Processing Action



IDE extension examples - Runtime

Additional features:

- Basic Demo Validation



IDE extension examples - Runtime

Additional features:

- New Visualization

