zeligsoft

Exploring UML in Eclipse

zeligsoft

What is the UML2 project?

- An EMF based implementation of the UML 2.x specification
- A base for modeling tools to build upon
- With support for UML Profiles

zeligsoft

EMF Implementation of UML2 Spec

- What many consider the reference implementation for the UML 2 specification
- Metamodel completely specified as an EMF model
- Support for many of the UML techniques
  - Redefinition
  - Subsetting

zeligsoft

Base for Modeling Tools

- Tools share a common foundation improving
  - Model interchange
  - Add-on tools, for example model analysis
  - Tool independent transformations
- Shared interpretation of the UML 2.x specification
- Models serialized to common XMI format

zeligsoft

Support for UML Profiles

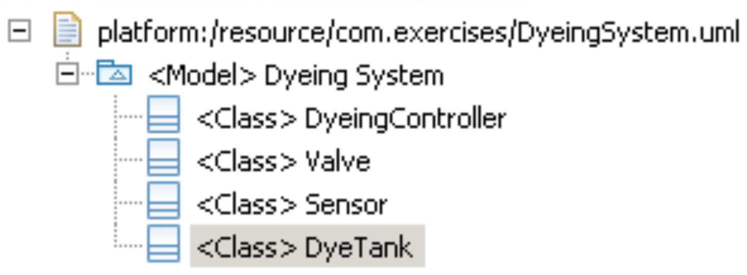
- Profiles are UML 2.x extensibility and customization mechanism
  - Use domain concepts
  - Refining semantics
  - Customize presentation
  - Tagging model elements
  - Add domain specific information
- Enables the definition of domain specific languages
  - For example Rose Real-Time in RSA

Creating a UML model

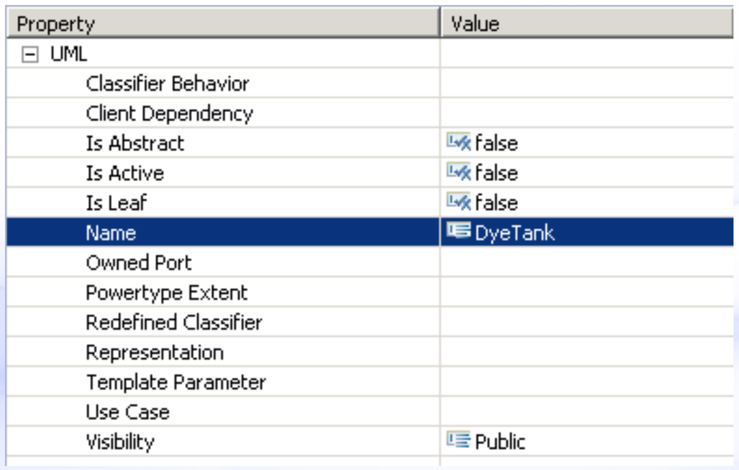
- Default editor
  - Tree based editor
- Rational Modeling Platform
  - Visual modeling
- Programmatically
  - Use the Factory for the UML 2 model

UML Model Editor

- **File → New...** from the main menu
- Select UML Model
  - Under Example EMF Model Creation Wizards
- Provide a name
- Set Model Object to Model
- Open with UML Model Editor
- Use Create Child in the context menu to create model elements



platform:/resource/com.exercises/DyeingSystem.uml
- <Model> Dyeing System
  - <Class> DyeingController
  - <Class> Valve
  - <Class> Sensor
  - <Class> DyeTank

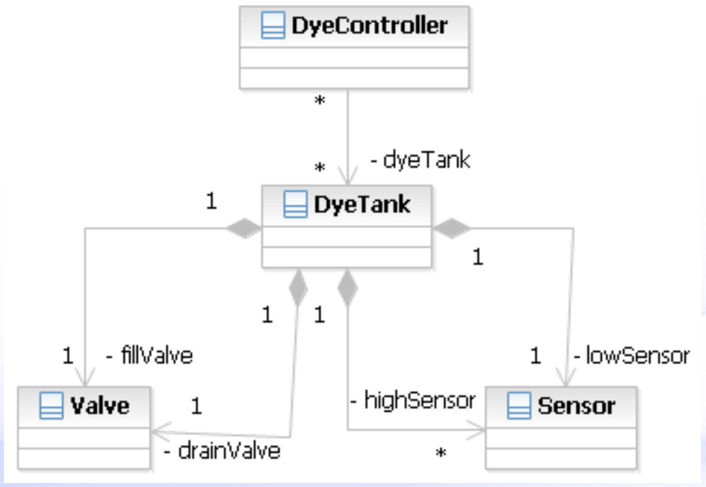


Property	Value
UML	
Classifier Behavior	
Client Dependency	
Is Abstract	false
Is Active	false
Is Leaf	false
Name	DyeTank
Owned Port	
Powertype Extent	
Redefined Classifier	
Representation	
Template Parameter	
Use Case	
Visibility	Public

zeligsoft

Rational Modeling Platform

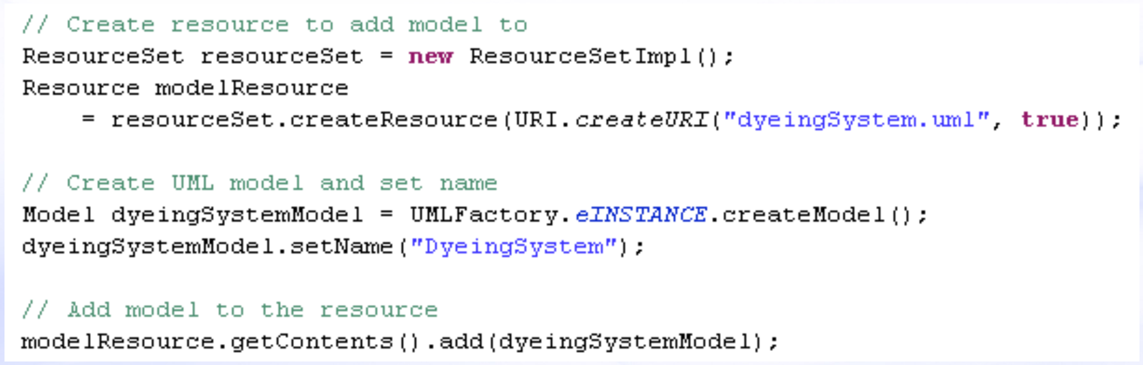
- Create a UML 2 model graphically
- How to
  - **File → New...** from the main menu
  - UML Model
    - Under Modeling
  - Set the model name
- Create elements using diagrams and project explorer
- Created model can be opened in the UML Model Editor



zeligsoft

Creating Models in Code

- Programmatically create UML models and elements
- Use of the standard EMF generated API
- EMF reflective capabilities available



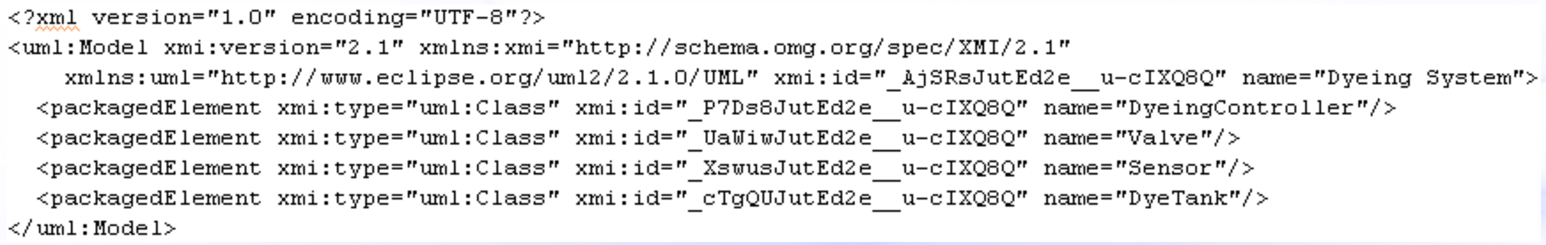
```java
// Create resource to add model to
ResourceSet resourceSet = new ResourceSetImpl();
Resource modelResource
    = resourceSet.createResource(URI.createURI("dyeingSystem.uml", true));

// Create UML model and set name
Model dyeingSystemModel = UMLFactory.eINSTANCE.createModel();
dyeingSystemModel.setName("DyeingSystem");

// Add model to the resource
modelResource.getContents().add(dyeingSystemModel);
```

More Creating Models in Code

- To create UML model elements use
  - UMLFactory, or
  - For Packages and Models
    - createdOwnedClass
    - createNestedPackage
    - createdOwnedPrimitiveType
    - createdOwnedEnumeration
    - …

Persistence and Serialization

- Models are persisted in an XMI compliant format
- Tools built on top of UML2 project should be able to load the model
  - Likely won't maintain diagrams



```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Model xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
    xmlns:uml="http://www.eclipse.org/uml2/2.1.0/UML" xmi:id="_AjSRsJutEd2e__u-cIXQ8Q" name="Dyeing System">
  <packagedElement xmi:type="uml:Class" xmi:id="_P7Ds8JutEd2e__u-cIXQ8Q" name="DyeingController"/>
  <packagedElement xmi:type="uml:Class" xmi:id="_UaWiwJutEd2e__u-cIXQ8Q" name="Valve"/>
  <packagedElement xmi:type="uml:Class" xmi:id="_XswusJutEd2e__u-cIXQ8Q" name="Sensor"/>
  <packagedElement xmi:type="uml:Class" xmi:id="_cTgQUJutEd2e__u-cIXQ8Q" name="DyeTank"/>
</uml:Model>
```

Summary

- In this module we explored
  - What the UML2 project is
  - How to create UML2 models
    - UML Model Editor
    - Rational Modeling Platform
    - Through code
  - Persistence

Extending UML

- UML 2 is a general purpose modeling language
  - Large and expressive
- Often specific domains need to extend it
  - Additional concepts
  - Restricting metamodel
  - Defining domain specific semantics for elements
- Different approaches
  - Feather weight - Tagging a model
  - Light weight - UML Profile
  - Heavy weight - Extending the metamodel

Tagging a model with keywords

- Lightweight approach to adding data to a model
  - Control code generators
  - Categorizing
- Create by
  - Adding Annotation with source set to UML
  - Add details to the Annotation with key being the keyword
- Simple API to retrieve keywords
  - addKeyword, removeKeyword, and hasKeyword

zeligsoft

Extending UML with a Profile

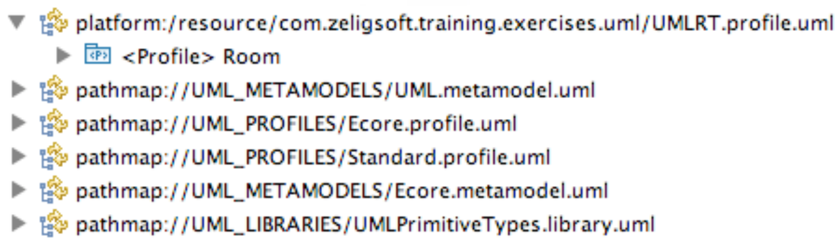
- UML profiles provide a lightweight approach to extending the UML metamodel
- UML profiles can be created in the same way as UML models
  - UML Model Editor
  - Rational Modeling Platform
  - Programmatically
- Profiles can be published or registered
  - Makes them accessible to others
  - Approach used by RSA RTE

UML Profile

- Primary construct in a profile is a stereotype
  - Extends one or more metaclasses from the UML
  - May add information and/or constraints
  - May add/or constrain semantics
  - May change graphical display
- Can be **applied** to one or more UML models or packages
  - Stereotypes applied to model or package and its contents

Creating UML Profile

- Using UML Model Editor
- Select **File → New...**
- Choose UML Model and provide a name
  - Under Example EMF Model Creation Wizards
- Set Model Object to Profile
- Open with UML Model Editor
- Select **UML Editor → Profile → Reference Metamodel...** from the main menu
- Choose the UML metamodel

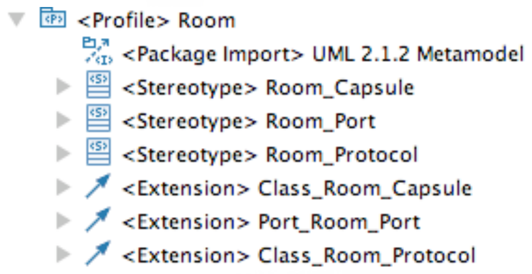


```
▼ platform:/resource/com.zeligsoft.training.exercises.uml/UMLRT.profile.uml
    ▶ <Profile> Room
▶ pathmap://UML_METAMODELS/UML.metamodel.uml
▶ pathmap://UML_PROFILES/Ecore.profile.uml
▶ pathmap://UML_PROFILES/Standard.profile.uml
▶ pathmap://UML_METAMODELS/Ecore.metamodel.uml
▶ pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml
```

zeligsoft

Creating Stereotype

- Open profile in UML Model Editor
- Select the profile object and **New Child ➔ Owned Stereotype ➔ Stereotype** from the context menu
- Select the stereotype and **UML Editor ➔ Stereotype ➔ Create Extension...** from the main menu
- Select the UML metaclasses to extend
- Creates Extension object in the profile



```
<Profile> Room
    <Package Import> UML 2.1.2 Metamodel
    <Stereotype> Room_Capsule
    <Stereotype> Room_Port
    <Stereotype> Room_Protocol
    <Extension> Class_Room_Capsule
    <Extension> Port_Room_Port
    <Extension> Class_Room_Protocol
```

UML Profile Static vs. Dynamic

- Dynamic Profile
  - The profile is defined in a model whose model object is a Profile
  - No code is generated
  - Profile is deployed in a plug-in and registered as a dynamic package
- Static Profile
  - The profile is defined in a model whose model object is a Profile
  - An API for the profile is generated to make it easier to work with the profile in code
  - Profile code is deployed in a plug-in and registered as a static package

Deploying a Dynamic Profile

- Define the profile
  - Converts the profile elements into Ecore representation
  - Select the profile element in the model
  - Select **UML Editor → Profile → Define** from main menu
  - This stores Ecore representation as an annotation in the profile
- Make the project a plug-in project and make sure the profile model is in the build
- Register the profile
  - org.eclipse.uml2.uml.dynamic_package

Deploying a Static Profile

- Generate the profile code
- Make the project a plug-in project and make sure the src folder with the generated code is in the build
- Register the profile
  - org.eclipse.uml2.uml.generated_package

Generating Profile Code

- Apply Ecore profile to the profile object
- Apply ePackage stereotype to the profile object
  - Set the following ePackage properties
    - NS URI
    - NS Prefix
    - Base Package
- Create an EMF model from the profile using the UML model importer
- Configure generator settings
- Generate Model code for the EMF model

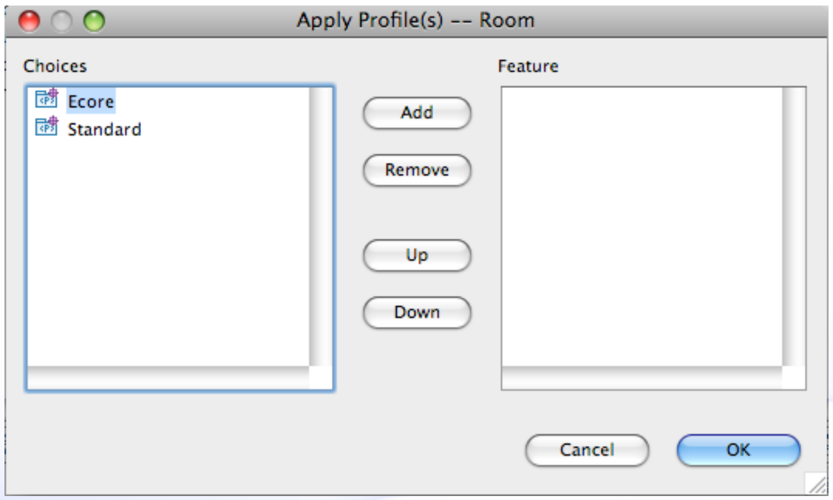
zeligsoft

Summary

- In this module we explored

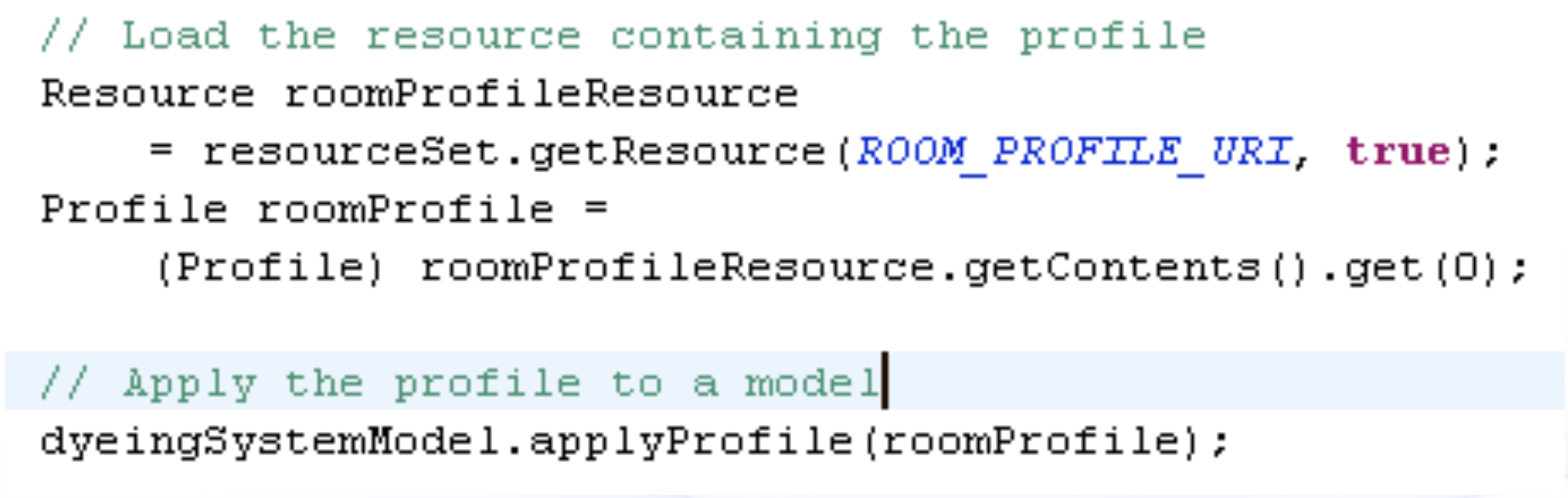
Applying a UML profile

- A Profile is applied to a Model or Package
- Open model in UML Model Editor
- Select model or package object and **UML Editor → Package → Apply Profile...** from the main menu
- Choose the profiles to apply
- Creates a Profile Application object in the package



zeligsoft

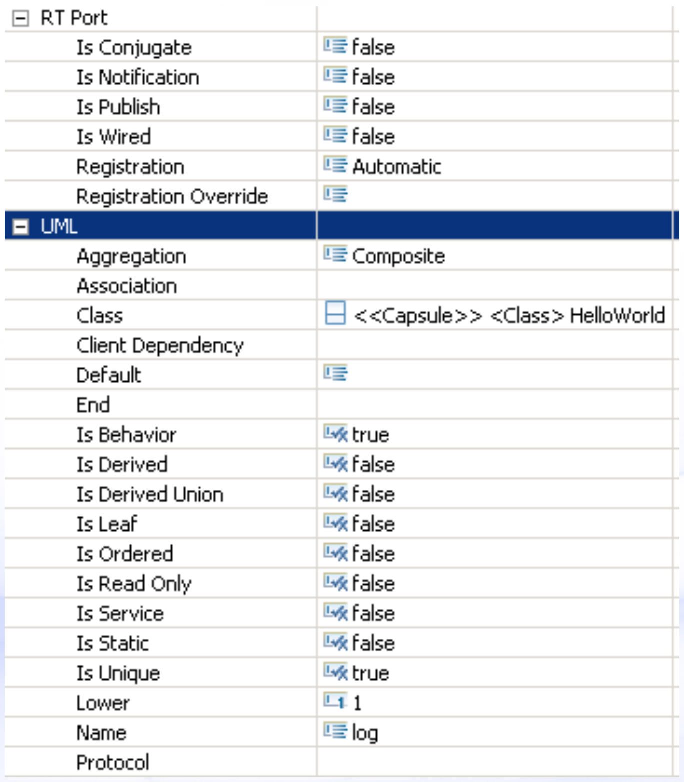
Applying a UML Profile - Code



```
// Load the resource containing the profile
Resource roomProfileResource
    = resourceSet.getResource(ROOM_PROFILE_URI, true);
Profile roomProfile =
    (Profile) roomProfileResource.getContents().get(0);

// Apply the profile to a model
dyeingSystemModel.applyProfile(roomProfile);
```

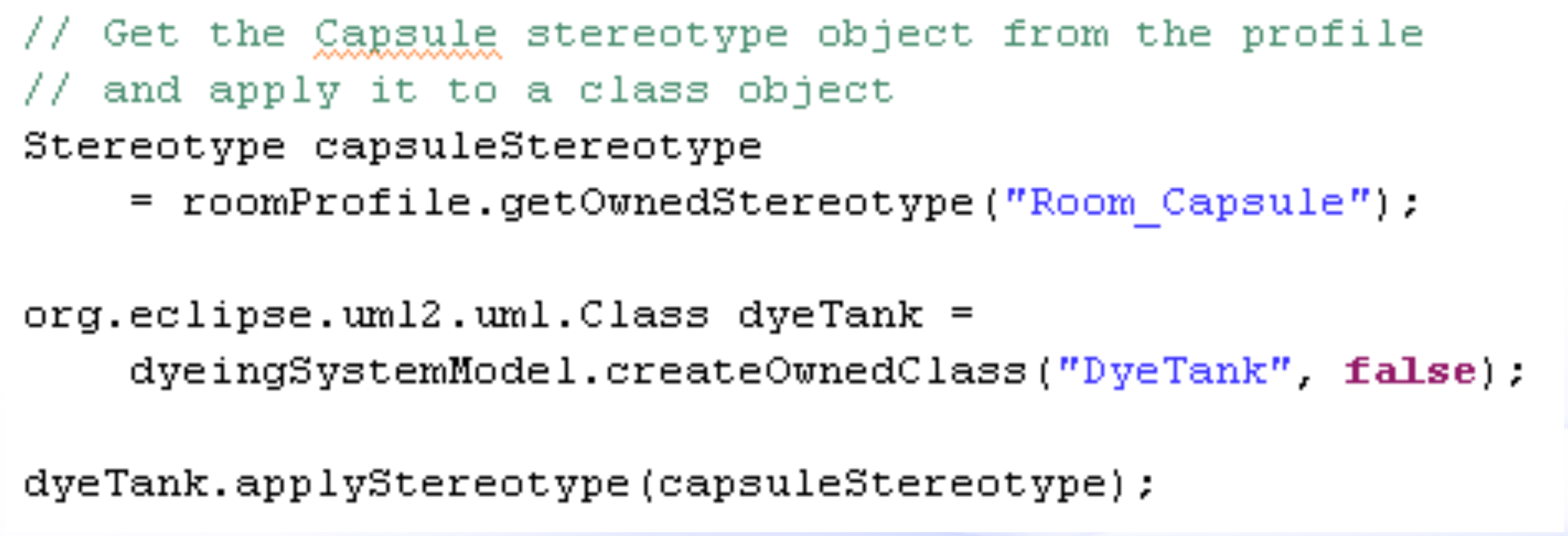
Applying a Stereotype

- Stereotypes are applied to elements in a model
  - More than one can be applied
  - Applicable elements defined by Stereotypes metaclass extensions
  - Attributes of Stereotypes can be set in the properties view of the UML Model Editor
- Select the element in the editor and **UML Editor ⇒Element ⇒Apply Stereotype...** from the main menu



⊟ RT Port	
Is Conjugate	false
Is Notification	false
Is Publish	false
Is Wired	false
Registration	Automatic
Registration Override	
⊟ UML	
Aggregation	Composite
Association	
Class	<<Capsule>> <Class> HelloWorld
Client Dependency	
Default	
End	
Is Behavior	true
Is Derived	false
Is Derived Union	false
Is Leaf	false
Is Ordered	false
Is Read Only	false
Is Service	false
Is Static	false
Is Unique	true
Lower	1
Name	log
Protocol	

Applying a Stereotype - Code



```
// Get the Capsule stereotype object from the profile
// and apply it to a class object
Stereotype capsuleStereotype
    = roomProfile.getOwnedStereotype("Room_Capsule");

org.eclipse.uml2.uml.Class dyeTank =
    dyeingSystemModel.createOwnedClass("DyeTank", false);

dyeTank.applyStereotype(capsuleStereotype);
```

zeligsoft

Working Stereotype Properties

Accessing a stereotype value

```
drainValvePort.getValue(portStereotype, "conjugated");
```

Setting a stereotype value

```
drainValvePort.setValue(portStereotype, "conjugated", false);
```

Adding to a stereotype list property

```
((List) dyeSystemComponent.getValue(componentStereotype, "includes"))
    .add("MyInclude.h");
```

Summary

- In this module we explored
  - Applying a profile
  - Applying stereotypes to model elements
  - Working with profiles in code
    - Applying profile
    - Applying stereotype
    - Getting/Setting stereotype values

Keywords in RSA-RTE

- [TM TODO]

Profiles in RSA-RTE

- Profiles are used extensively in RSA-RTE
  - Extending/constraining the UML to UML-RT semantics and notation
    - UMLRealTime profile
    - UMLRealTime model libraries for things like Frame, and Log
  - Adding information to model elements to generate code
    - C++ support through CPPPropertySets profile
    - C++ model libraries for
- When a RSA-RTE model is created
  - UMLRealTime profile is applied
  - CPPPropertySets profile is applied
  - RTClasses, RTComponents and CPPPrimitiveDatatypes libraries

zeligsoft

Creating Profiles in RSA-RTE

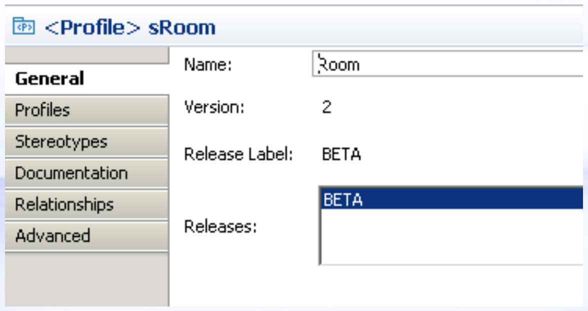
- Model the domain
  - Concepts, attributes and relationships in a Profile
  - Complimentary model libraries
  - Constraints - OCL or Java
- Publish the Profile
  - Helps RSA-RTE with versioning and migration
- Register the Profile in a plug-in
  - Make it available for RSA-RTE to apply it
- Can export to open source UML 2

Profile

- **File → New → Other…**
- Select UML Profile or UML Profile Project…
  - Under Modeling → UML Extensibility
- Set name and import the UML Primitive Types library
- Creates a Profile object that already imports the UML metamodel
- Add stereotypes, classes and relationships

Releasing a Profile

- RSA-RTE provides a release capability for profiles
  - A released profile is to be additive otherwise backwards compatibility may not work
- To release a profile
  - Select Release in the context menu of the profile object
  - Associate a label with the release
  - Be careful how often a profile is released



Name: Room
Version: 2
Release Label: BETA
Releases: BETA

Publishing a Profile

- Publishing a profile makes it available for others to install and use in their RSA-RTE models
- To publish
  - Make the project a plug-in project
  - Add the profile file to the build
  - Extend “com.ibm.xtools.uml.msl.UMLProfiles”
    - Enables RSM to find the profile
  - Publish the plug-in

```
<extension point="com.ibm.xtools.uml.msl.UMLProfiles">
    <UMLProfile id="com.zeligsoft.exercises.profiles.room"
        name="ROOM"
        path="pathmap://EXERCISE_PROFILES/Room.epx" required="false" visible="true" />
</extension>
```

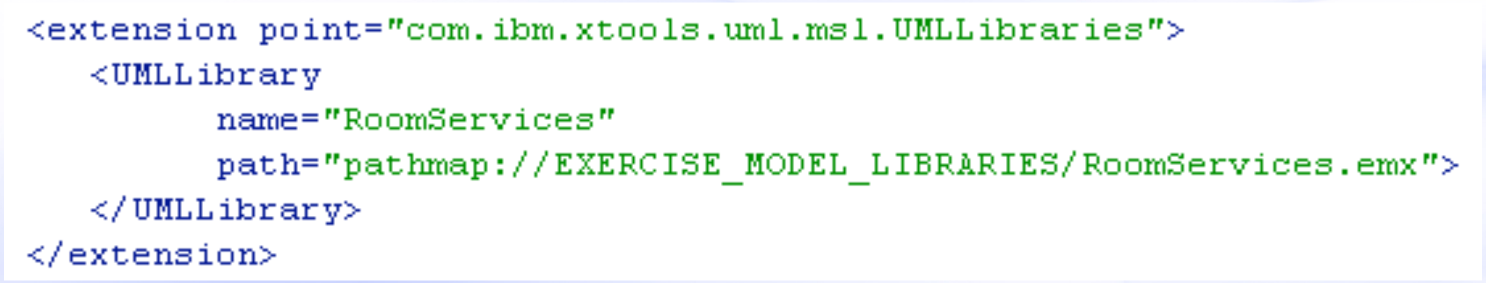
zeligsoft

Model Libraries

- Model libraries provide a mechanism for publishing a set of model elements for reuse
  - e.g. Data types, reusable components
- It is a model with the modelLibrary stereotype applied
  - modelLibrary is part of the Standard profile
- The elements will be read-only in the model that imports the library

Publishing a Model Library

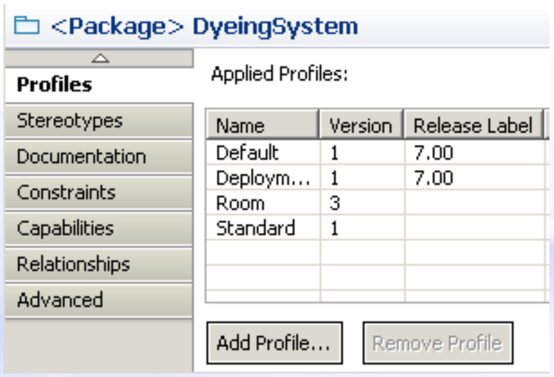
- Publishing a model library makes it available for others to import and use in their RSA-RTE models
- To publish
  - Make the project a plug-in project
  - Add the model file to the build
  - Extends com.ibm.xtools.uml.msl.UMLLibraries
    - Enables RSM to find the profile
  - Publish the plug-in



```
<extension point="com.ibm.xtools.uml.msl.UMLLibraries">
   <UMLLibrary
         name="RoomServices"
         path="pathmap://EXERCISE_MODEL_LIBRARIES/RoomServices.emx">
   </UMLLibrary>
</extension>
```

Applying Profiles in RSA-RTE

- On a Model or Package "Add Profile…"
- Apply Stereotype to Elements in the model
- Set attributes of the Stereotype
  - Advanced tab
  - Stereotype tab

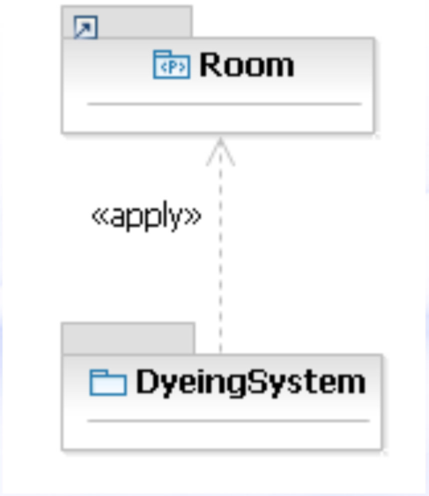


<Package> DyeingSystem

Profiles | Stereotypes | Documentation | Constraints | Capabilities | Relationships | Advanced

Applied Profiles:

Name	Version	Release Label
Default	1	7.00
Deploym…	1	7.00
Room	3	
Standard	1	



Summary

- In this module we explored

zeligsoft

Transactions with RSA-RTE

- Recall that transactions ensure that a command on the model is executed exclusively
- And are useful for integrating with the workbench’s undo/redo infrastructure
- To get the editing domain used by RSA-RTE
  - Use the UMLModeler object UMLModeler.getEditingDomain()