Tutorial

19.03.2011

Manuel Renz & Matthias Ziegler & Darko Palic

XENOVATION

1 Table of Contents

2 License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. **"Adaptation"** means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

b. **"Collection"** means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined below) for the purposes of this License.

c. **"Creative Commons Compatible License"** means a license that is listed at http://creativecommons.org/compatiblelicenses that has been approved by Creative Commons as being essentially equivalent to this License, including, at a minimum, because that license: (i) contains terms that have the same purpose, meaning and effect as the License Elements of this License; and, (ii) explicitly permits the relicensing of adaptations of works made available under that license under this License or a Creative Commons jurisdiction license with the same License Elements as this License.

d. **"Distribute"** means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.

e. **"License Elements"** means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, ShareAlike.

f. **"Licensor"** means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

g. **"Original Author"** means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

h. **"Work"** means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment

in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

i. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

j. **"Publicly Perform"** means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

k. **"Reproduce"** means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights. Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;

b. to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked "The original work was translated from English to Spanish," or a modification could indicate "The original work has been modified.";

c. to Distribute and Publicly Perform the Work including as incorporated in Collections; and,

d. to Distribute and Publicly Perform Adaptations.

e. For the avoidance of doubt:

i. **Non-waivable Compulsory License Schemes**. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;

ii. **Waivable Compulsory License Schemes**. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor waives the exclusive right to collect such royalties for any exercise by You of the rights granted under this License; and,

iii. **Voluntary License Schemes**. The Licensor waives the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(c), as requested.

b. You may Distribute or Publicly Perform an Adaptation only under the terms of: (i) this License; (ii) a later version of this License with the same License Elements as this License; (iii) a Creative Commons jurisdiction license (either this or a later license version) that contains the same License Elements as this License (e.g., Attribution-ShareAlike 3.0 US)); (iv) a Creative Commons Compatible License. If you license the Adaptation under one of the licenses mentioned in (iv), you must comply with the terms of that license. If you license the Adaptation under the terms of any of the licenses mentioned in (i), (ii) or (iii) (the "Applicable License"), you must comply with the terms of the Applicable License generally and the following provisions: (I) You must include a copy of, or the URI for, the Applicable License with every copy of each Adaptation You Distribute or Publicly Perform; (II) You may not offer or impose any terms on the Adaptation that restrict the terms of the Applicable License or the ability of the recipient of the Adaptation to exercise the rights granted to that recipient under the terms of the Applicable License; (III) You must keep intact all notices that refer to the Applicable License and to the disclaimer of warranties with every copy of the Work as included in the Adaptation You Distribute or Publicly Perform; (IV) when You Distribute or Publicly Perform the Adaptation, You may not impose any effective technological measures on the Adaptation that restrict the ability of a recipient of the Adaptation from You to exercise the rights granted to that recipient under the terms of the Applicable License. This Section 4(b) applies to the Adaptation as incorporated in a Collection, but this does not require the Collection apart from the Adaptation itself to be made subject to the terms of the Applicable License.

c. If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and (iv) , consistent with Ssection 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

d. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to

the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

b. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

f. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional

rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

http://creativecommons.org/licenses/by-sa/3.0/legalcode

3 Introduction

This tutorial basically consists of the practical part of the degree dissertation with the titles *„Erstellung eines MDA-Generators für Validierungen auf Basis der Bean-Validation-API“* and *„Erstellung eines MDA-Generators für CRUD-Operationen in einer Persistenzschicht auf Basis der Java Persistence API“*, which are the basis for droMDAry. The following chapters were adapted and summarised for the droMDAry tutorial.

The Tutorial is split up into the following sections:

- Installation and configuration of different software packages
- Use of cartridges (BVA/JPA)
- Instructions for use of *Enterprise Architect*

4 Installation of the integrated development environment

Basically the installation could be divided into the following steps:

- Download and installation of different software packages
- Configuration of the *Eclipse* IDE and installation of divers plug-ins
- Import of cartridges

4.1 Preconditions

DroMDAry was developed under *Windows*. The used software-tools as well as MDA-Cartridges work correct with *Windows XP / Vista / 7*. The compatibility of droMDAry to *Linux/Unix/Mac* was not tested. There is a CrossOver-Version for the software-tool *Enterprise Architect*. For further information see this home-page:

http://www.sparxsystems.com/support/faq/ea_on_linux.html

If you have had an installed the *EclipseIDE*, you have to check whether the conditions from the chapters 4.3.3 and 4.3.6 were fulfilled. This step is necessary because some plug-ins may need to be updated.

There are some versions of the *Eclipse* IDE for *Linux* and *Mac*. For further information follow this link:

http://www.eclipse.org/.

4.1.1 System Requirements

- CPU with 1,8 GHz
- 1 GB RAM
- 600 MB space on hard disc

These specifications apply to the operation systems *Windows XP/Vista*. Other operation systems have not been tested.

4.2 Installation of Enterprise Architect

A current trial version of *Enterprise Architect* can be found at the following homepage: http://www.sparxsystems.de/enterprise-architect/download-trial/

By clicking "EA Trial Version" you will start the download of the trial version of *Enterprise Architect*. The trial version is available for 30 days. Purchasing will make it possible to work in *EA* after this period. For further information follow this link:

http://www.sparxsystems.de/enterprise-architect/ea-price/
The **Version 7.5 Build 845** was used by the implementation of cartidges.

After starting the installation few instructions must be followed to complete this process.

4.3 Installation and Configuration of the IDE

The following section describes the steps of installation and configuration of the IDE *Eclipse.*

4.3.1 Preparations

Before beginning with the installation all required software packages must be downloaded. In the following a list with version numbers and download links:

1. Java SE Development Kit 6u22 (JDK)
   http://java.sun.com/javase/downloads/index.jsp
2. Eclipse Modelling Tools 3.6 SR1

   http://www.eclipse.org
3. droMDAry-Alpha
   http://sourceforge.net/projects/dromdary/files/droMDAry-Alpha/droMDAry-alpha.zip/download

4.3.2 Installation of JDKs

The whole toolchain depends on a functional Java Development Environment. At least JDK in version 1.5 must be installed. Also later versions can be installed, because *"Compiler Compliance Level"* can be changed later.

4.3.3 Installation of Eclipse

The downloaded ZIP-file *"eclipse-modeling-helios-SR1-incubation-win32.zip"* can be extracted to any folder. In this description the archive is extracted to *"C:/"*. The IDE *Eclipse* is available under *"C:/eclipse"*.

4.3.4 Configuring workspace directory

The *Eclipse* IDE can be started from *„C:\eclipse\eclispe.exe"*.

A new workspace at *"C:\Documents und Settings\<User>\workspace"* will be created by the first start. Some problems may occur because of blanks in the directory path. Because of this reason the workspace must be moved to another directory, for instance *"C:\workspace"*.

The use of the proposed path is not recommended. A separate directory can be used through clicking on *"File → Switch Workspace → Other"* (see Image 1:).

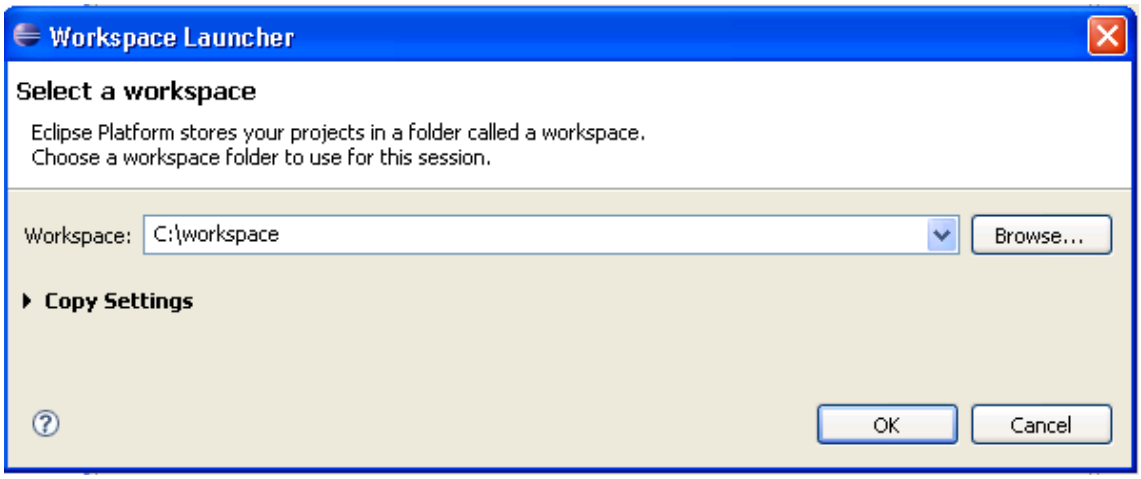


Image 1: Screenshot of workspace launcher

4.3.5 Installation of other Plug-Ins

Most of the required plug-ins are already installed in this version of *Eclipse Modelling Tools*. Nevertheless some more plug-ins must be installed:

- MWE 2 language SDK in version 1.0.1
- MWE 2 runtime SDK in version 1.0.1
- MWE SDK in version 1.0.0
- Xpand SDK in version 1.0.1

The *"MWE 2"* plug-ins are necessary to be able to open the UML2- and XMI-files. Use the integrated update manager to install these plug-ins. To open it go to

"Help → Install New Software...". Type *"MWE"* into the search bar and select *"Helios - http://download.eclipse.org/releases/helios"* from the dropdown menu *"Work with:"*. Then click *"Select All"* and follow the instructions. (see Image 2: Screenshot of *Eclipse* software update manager)

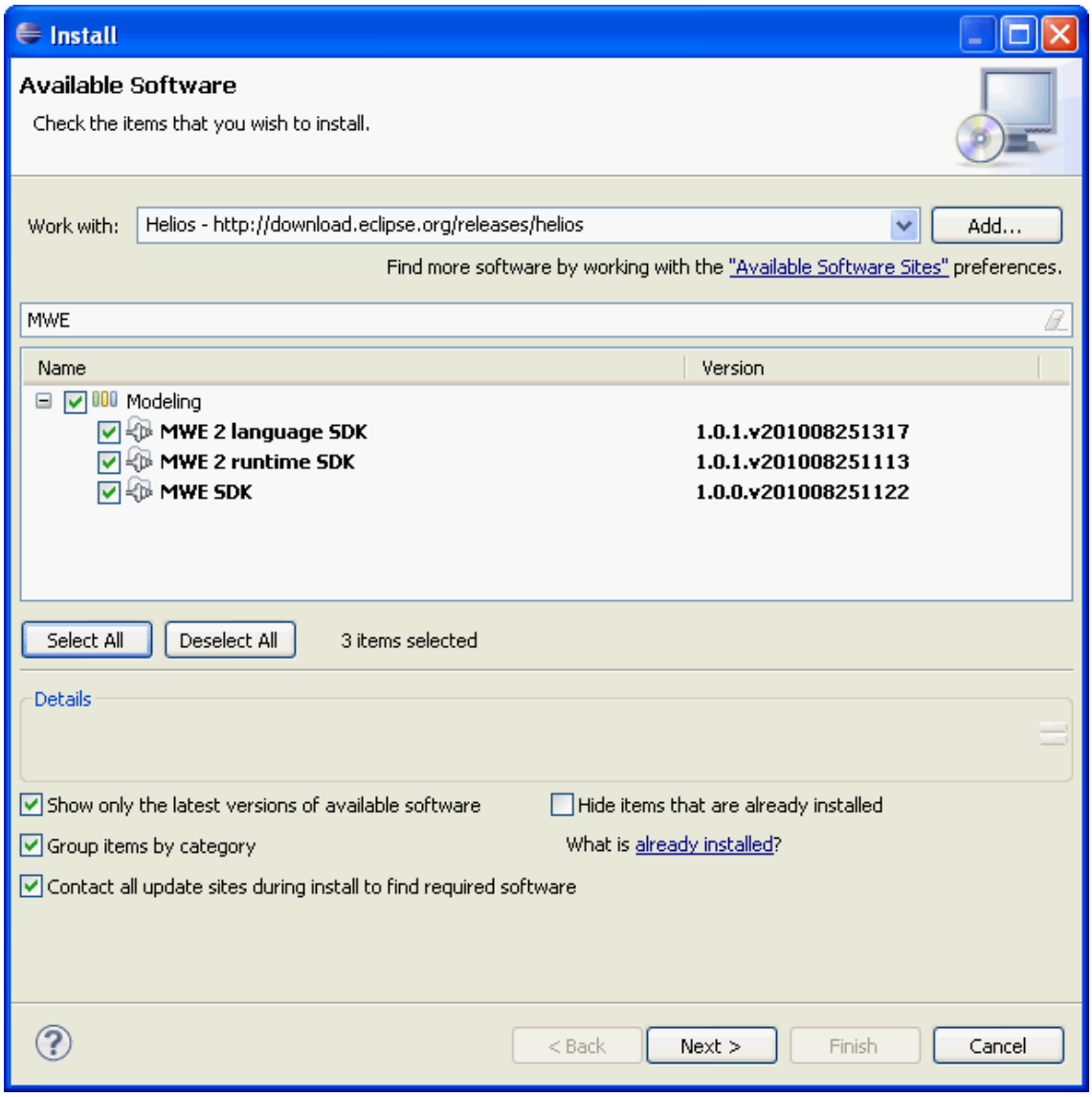


Image 2: Screenshot of *Eclipse* software update manager

When finished with installing the plugins click *"Restart Now"*. Start the update manager again, type in the search bar "Xpand SDK" and repeat the same steps as you did for *"MWE"*.

4.3.6 Initialisation of the Enterprise Architect API

The UML2-Exporter needs a DLL-file which provides the *Enterprise Architect* API , to access project files via this API. Furthermore this file must be accessible for the environment variable *"PATH".* To fulfil these conditions you can copy the DLL-file ie.g. to the folder *"C:\Windows\system32" or modify the PATH variable* . This DLL-file can be found at *"C:\Program Files\Sparx Systems\EA\Java API"* or at *"C:\Program Files\Sparx Systems\EA Trial\Java API".*

4.4 Import of cartridges

The last chapter deals with installation and configuration of the IDE. The next step is the integration of the MDA-cartridges into the configured workspace,

You can download the droMDAry cartridges form the projects sourceforge download site.

http://sourceforge.net/projects/dromdary/files/droMDAry-Alpha/droMDAry-alpha.zip/download
Extract this file to any folder, for instance *"C:\temp\"*
All six cartridges have to be imported to the workspace:

- bva.exporter
- bva.generator
- jpa.exporter
- jpa.generator
- common.exporter
- common.generator
- core
- docs
- environment

The exporter cartridges are responsible for the export of *Enterprise Architect* models and the generator cartridges are responsible for the generation of java source code from the exported models.

BVA and JPA are two separate cartridges. Because of this there are two exporter and generator projects.

The cartridges named *"Common"* have a special function. They connect the BVA and JPA cartridges when both artefacts, BVA and JPA, have to be generated from one *Enterprise Architect model*.

4.4.1 Import of cartridges

To import the cartridges go to *"File → Import"*, open the folder *"General"*, select *"Existing Project into Workspace"* and click *"Next"*. (see Image 3: Screenshot of the import assistant). Check if *"Select root directory"* is activated and browse to the folder with the cartridges (in this case *"C:\temp\"*). All available Eclipse-projects are visible now. To copy all project files activate the option „Copy into workspace" and click the button *"Finish"*. (see Image 4: Screenshot of the assistant to select cartridges and Image 5: Screenshot of the import status). The import of the cartridges is completed. Close the window *"Welcome"*. As result new projects occur inside the workspace in the window *"Project Explorer"* (see Image 6: Screenshot of the package explorer after the import).

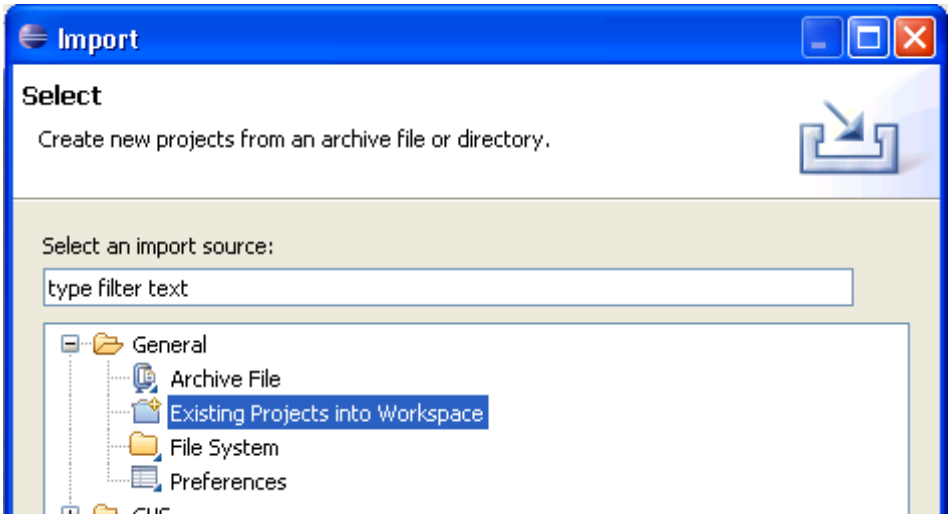


Image 3: Screenshot of the import assistant

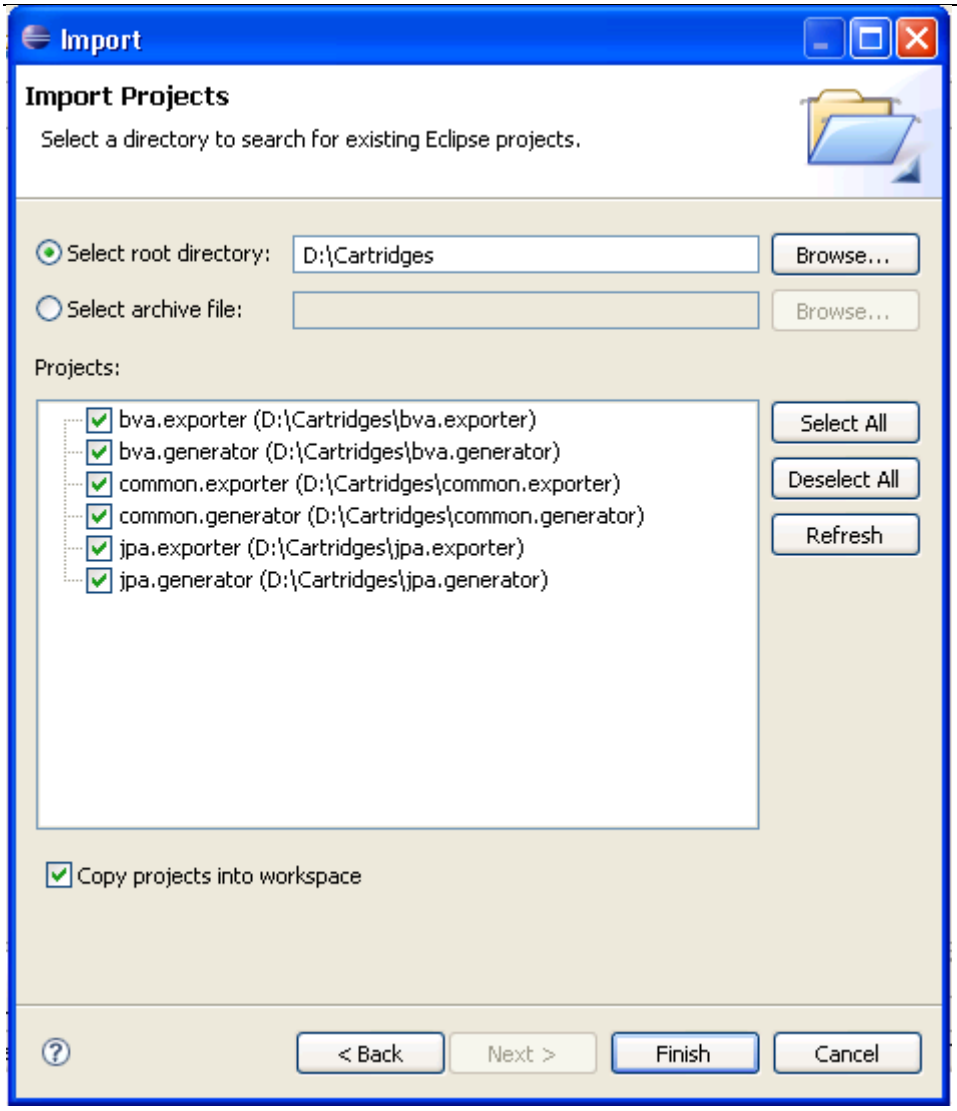


Image 4: Screenshot of the assistant to select cartridges

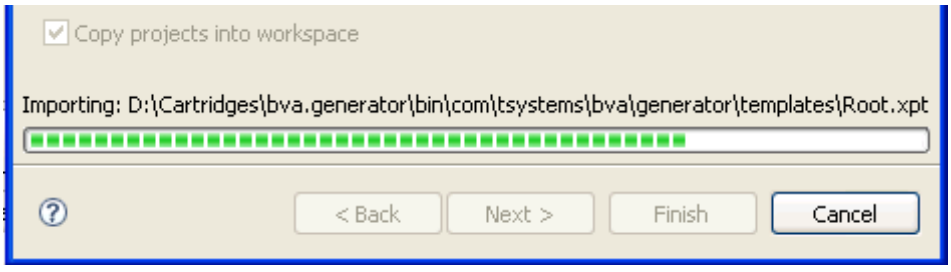


Image 5: Screenshot of the import status

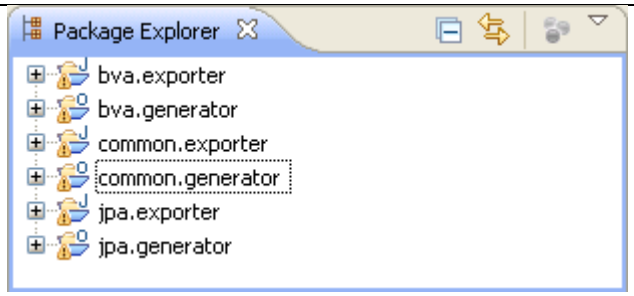


Image 6: Screenshot of the package explorer after the import

After these steps you have to fix build path for the *Enterprise Architect* API. To do this make right-click on *"common.exporter"* in the window *"Package Explorer"*. Go to *"Build Path → Configure Build Path"* and click *"Libraries"* in the new window. Select *"eaapi.jar"* and click "Edit". Than browse to *"C:\Program Files\Sparx Systems\EA\Java API\"* or to *"C:\Program Files\Sparx Systems\EA Trial\Java API\",* select *"eaapi.jar"* and confirm the dialogue.

Repeat this for *"bva.exporter"* and *"jpa.exporter".*

It is necessary to trigger a complete rebuild, you have to go to *"Project → Clean"*

Now it's possible to check the functionality of the prepared IDE.

4.4.2 Start of cartridges

Code generation is split up into two steps. The first step is the export of an EA-model into an EMF-model. The second is the code generation from the EMF-model.

To start the export process make right-click on the file *"common.exporter/wf/profileExportWorkflow.mwe"* in the window *"Package Explorer"* and than click "Run As → MWE Workflow". The status of progress is visible in the console-window.

When the workflow is completed, you will see the following message in the console window:

75061 INFO WorkflowRunner - workflow completed in 74026ms!

Note: When you are using a trial version of *Enterprise Architect* a licensing window opens up in the background, which you have to confirm by clicking *"Continue Trial"*. With Version 8.0 or higher you have to wait a while (10s) before confirming *"Continue Trial"* to avoid an error message.

To start the export of model make right-click on the file *"common.exporter/wf/modelExportWorkflow.mwe"* and than click *"Run As → MWE Workflow"*.

Now all required models are available in XMI-format, which is compatible with the EMF-format, and the generation of the source code can be started.

Start *"common.generator/src/main/resources/workflow.oaw"* in the same way as before. Abstract and interface classes with packages have been generated in the folder *"common.generator/src/generated/java".* Implementation classes have been generated in the folder "*common.generator/src/main/java".* These files won't be overwritten by all next generation processes to protect manual changes in the source code.

By default the BVA and JPA catridges were executed. You can change these setting in the file *"comon.generator/src/main/resources/workflow.properties"* from the line 32. The status of progress is visible in the console window. When the workflow is successful completed, you will see the following message in the console window:

```
INFO - Written 9 files to outlet [default](./src/generated/java)
INFO - workflow completed in 3335ms!
```

5 An example for using of the BVA cartridge

In this chapter you will see the use of the BVA cartridge. This process is split up in several steps. Firstly you will create a model, in which the BVA stereotypes will be implemented. You will use a standard validator for this implementation. Also creation and use of a user-defined validator will be shown. The next steps are transformation of the model into a XMI-file with *"common.exporter"* and generation of a source code with *"common.generator"*

5.1 Creation of model

The first step is creation of an UML-model, which will be the basis for generation. We use a class model, which describes a book catalogue. This book catalogue consists of these three parts:

- *Author*
- *Publisher*
- *Book*

It is not the object of this tutorial to model and create a detailed connection of classes, because this is the role of the JPA cartridge.

Note: The completed model which will be created in this section can be found at *"bva.exporter/Model/BVA_Tutorial.EAP"*.

You have to create the following attributes for the single classes (for more information see chapter 8):

Author

- *age* (int)
- *last name* (String)
- *first name* (String)
- *street* (String)
- *postcode* (String)

Publisher

- *street* (String)
- *publisherName* (String)
- *postcode* (String)

Book

- *edition* (int)
- *bookId* (int)
- *title* (String)
- *isbn* (String)

(see Image 7: UML-model for the BVA example)

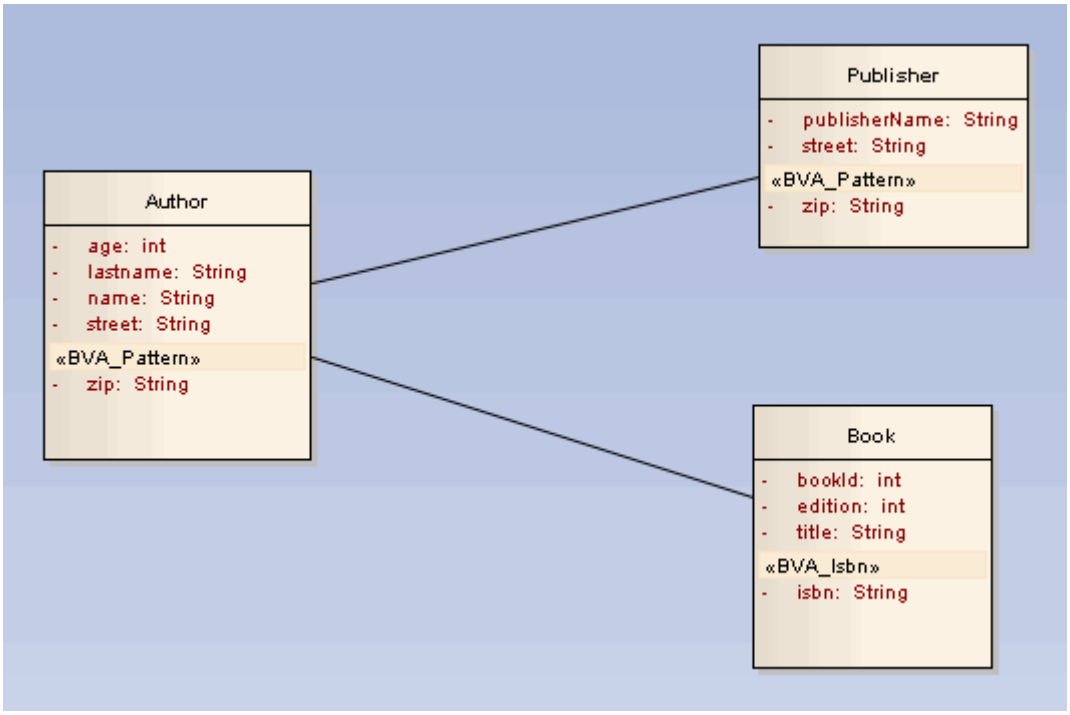


Image 7: UML-model for the BVA example

To illustrate the possibilities of the BVA-cartridge the following attributes should be validated with these rules:

- *Author::plz* → exactly 5 digits, all numbers
- *Publisher::plz* → exactly 5 digits, all numbers
- *Book::isbn* → 13 digits, all numbers, with checksum

5.2 Appliance of the standard BVA stereotypes

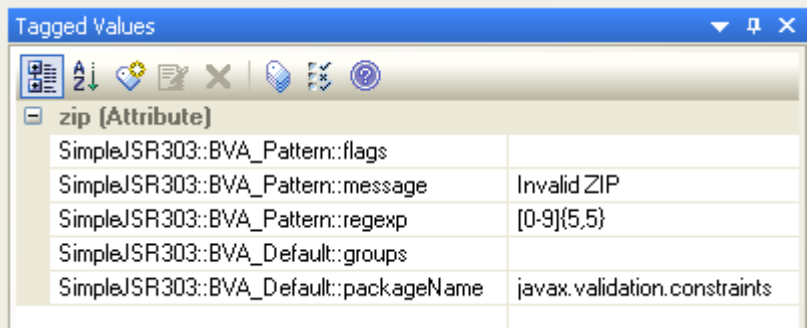
Firstly you have to connect the profile a *"SimpleJSR303"* to the model. The profile is available at *"/bva.exporter/Profil/SimpleJSR303_profile.xml"*. For more information see chapter 8.5

Now all standard validators are available at „*View → Other Project Tools → Ressources* “. You can apply the stereotype *“BVA_Pattern”* to the attributes *“Auhtor::plz” and “Publisher::zip”* as described in chapter 8.6.

You have just created the validation rules for the zip code by the attributes *“plz”*. (see Table 1: Tagged values and Screeenshot 8: Screenshot of the window „Tagged Values” of *the* stereotype *“BVA_Pattern*).

Tag Name	**Value**
message	Invalid ZIP
regexp	[0-9]{5,5}

Table 1: Tagged values for the postcode validation



Screeenshot 8: Screenshot of the window „Tagged Values” of the stereotype *“BVA_Pattern”*

5.3 Appliance of user-defined validators

The attribute *"Book::isbn"* is a number of 13 digits. This attribute is compatible to the EAN-code. The last digit is the checksum of the first 12 digits. The standard validators, which were created with *"SimpleJSR303"*, are not able to realise this requirements. For this reason you have to create a user-defined validator. This creation is split into three steps:

1. Creation of a user-defined profile and stereotype
2. Creation of a user-defined java annotation
3. Creation of a implementation of a user-defined validator

5.3.1 Creation of a user-defined BVA profile

Firstly you have to define a stereotype in a separate profile for the validator *"Isbn"*.

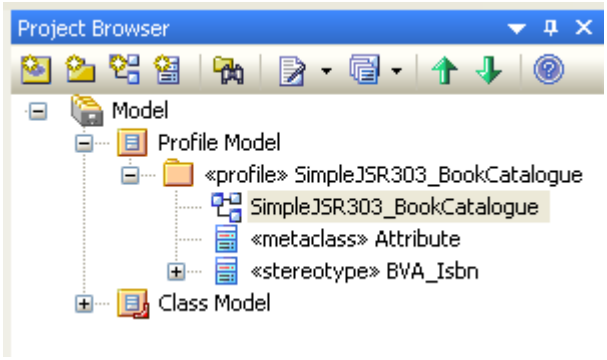


Image 9: Screenshot of the profile structure for user-defined validators

For this reason you have to create a profile structure. Follow the instructions below:

1. Right-click on *"Model"*
2. *"New View…"*
3. Type *"Profile Model"* in the field *„Name"* in the dialogue "*Create New View"* and click *"OK"*.
4. Right-click on *"View"* and select *"Add → Add Package…"*
5. Type *"SimpleJSR303_Bookcatalogue"* as package name in the dialogue *"New Package"* (see Image 10: Screenshot of the *dialogue "New Model* Package" *in*) and click *"OK"*.

6. Confirm the dialogue *"New Diagram"* by clicking *„OK"*.

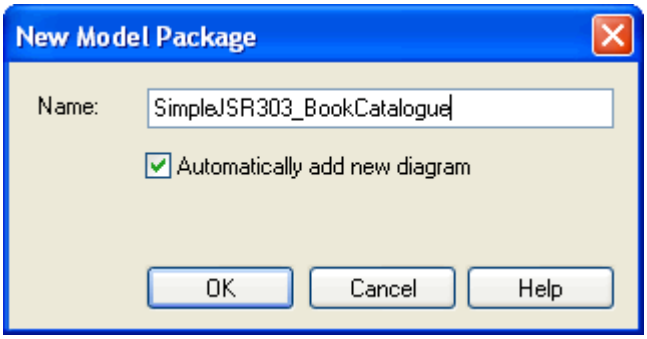


Image 10: Screenshot of the dialogue *"New Model Package"* in *EA*

The project structure is completed now. (see Image 11: Screenshot of the project structure without the stereotype *„profile"*).The package *"SimpleJSR303_Bookcatalogue"* must be connected to the stereotype *"profile"*. Double click on the package *"SimpleJSR303_Bookcatalogue"* and select *„profile"* in the field *"Stereotype"* (see Image 12: Screenshot of the package dialogue in *EA*).

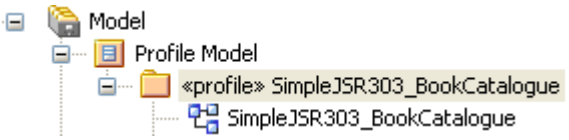


Image 11: Screenshot of the project structure without the stereotype *„profile"*

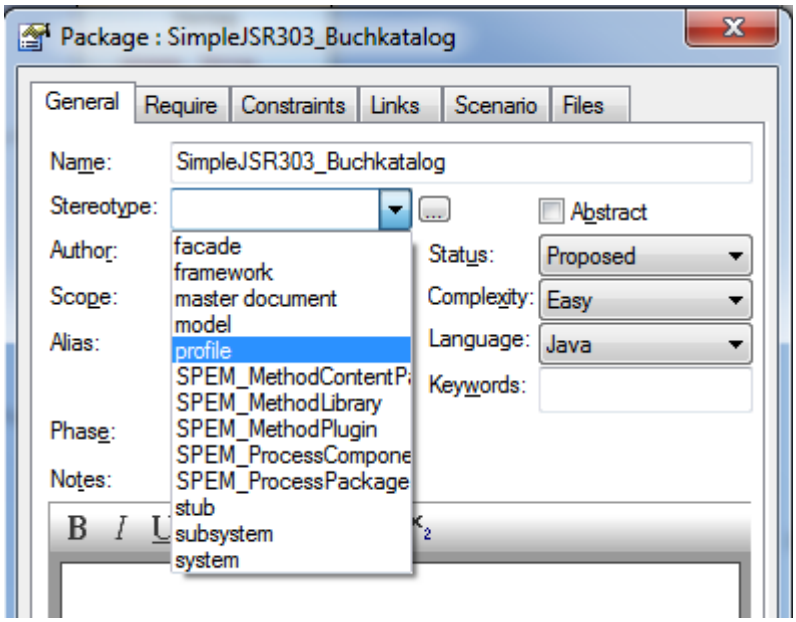


Image 12: Screenshot of the package dialogue in *EA*

5.3.2 Creation of a user-defined BVA stereotype

To create a user-defined BVA stereotype open the diagram *"SimpleJSR303_Bookcatalogue*" (see Image 11: Screenshot of the project structure without the stereotype *„profile"*). On the left side you can see the window *"Toolbox"* with profile elements.

(see Image 13: Screenshot of the profile toolbox in EA).

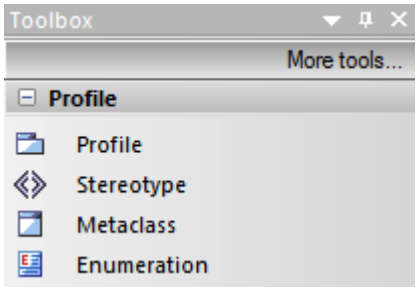


Image 13: Screenshot of the profile toolbox in *EA*

The new stereotype should only be applied to attributes. Because of this it has to be derived from the metaclass "A*ttributes"*. Click the toolbox element *"Metaclass"* and drag it onto the diagram window. In the new dialogue *"Create New Metaclass"* select *"Attribute"* in the field *"Element"* and click *"OK"* (see Image 14: Screenshot of the dialogue *"Create New Metaclass"* in *EA*). Next click the toolbox element *"Stereotype"* and drag it onto the diagram window. When releasing a new dialogue shows up, type *"BVA_Isbn"* into the field *"Name"*. The prefix *"BVA_"* is inevitable. Now you have to add some attributes to the stereotype, which will be available as *„Tagged Values"* in the model.

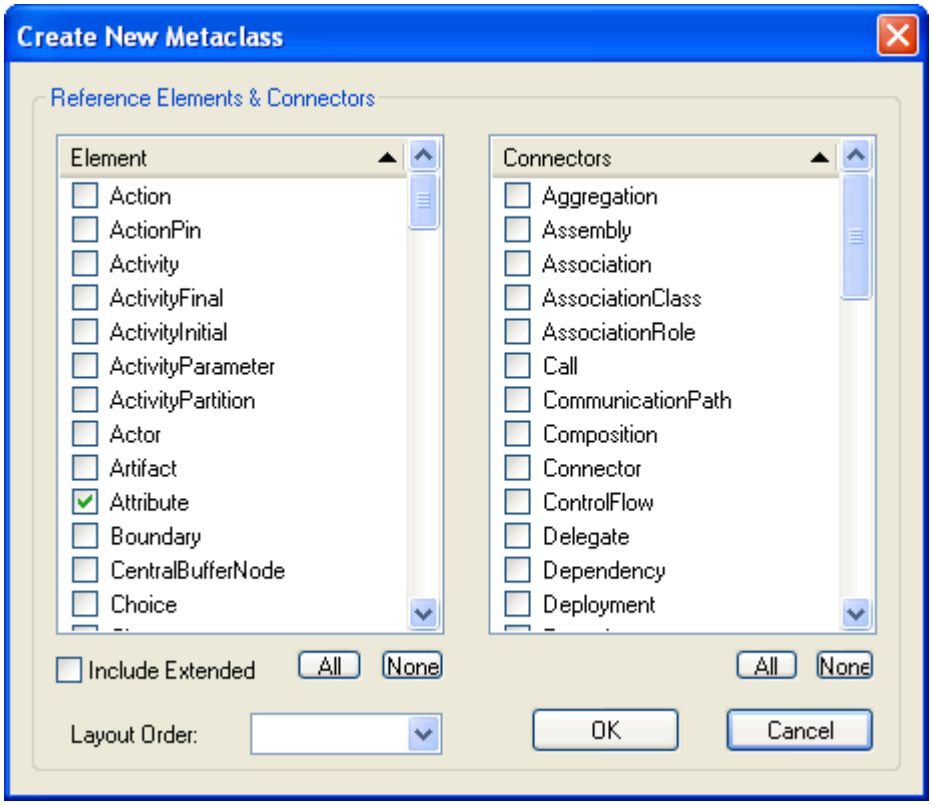


Image 14: Screenshot of the dialogue *"Create New Metaclass"* in *EA*

Click tab *"Details"* and than the button *"Attributes"*. Add the following attributes in the new dialogue *"BVA_Isbn Attributes"*. (see Table 2: Attributes of the stereotype *"BVA_Isbn"* and Image 15: Screenshot of the window *"BVA_Isbn Attributes"*).

Attribut Name	**Initial Value**

message (String)	Invalid ISBN
packageName (String)	com.tsystems.bva.buchkatalog

Table 2: Attributes of the stereotype *"BVA_Isbn"*

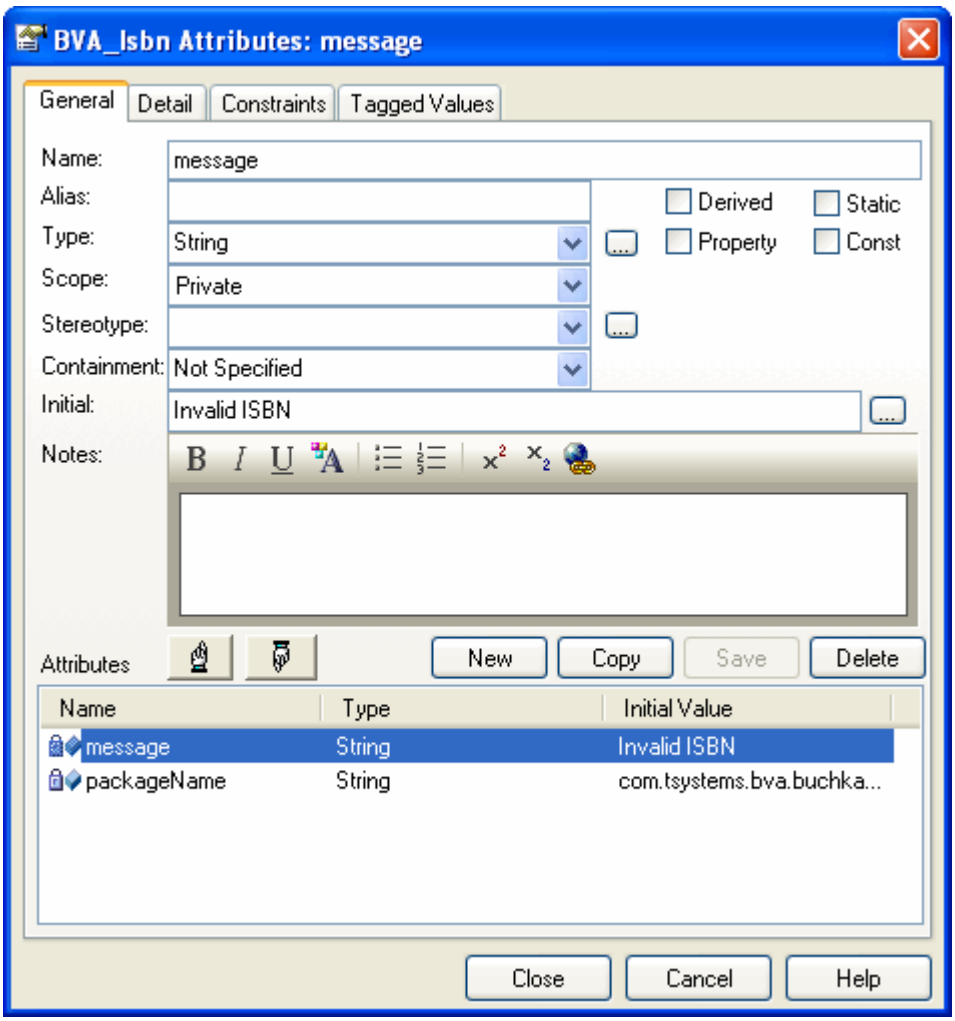


Image 15: Screenshot of the window *"BVA_Isbn Attributes"*

For more information see chapter 8.3. To finish the process click *"Close"* to close this dialogue.

Nun muss der Stereotyp nur noch von der Metaklasse ableiten und das Profil ist fertig. Dazu Klickt man einmal mit der linken Maustaste auf den Stereotypen, sodass die sogenannte „*Quick Link*" Funktion, wie in Image 16 dargestellt, eingeblendet wird.

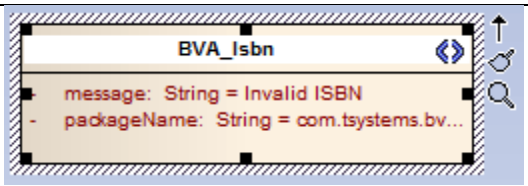


Image 16: Screenshot of stereotype with *"Quick Link"* symbol

Click the little black arrow and drag it onto to the metaclass. Than click *"Extend"* in the popup menu. The user-defined profile is completed now (see Image 17: Screenshot of extending of metaclass).

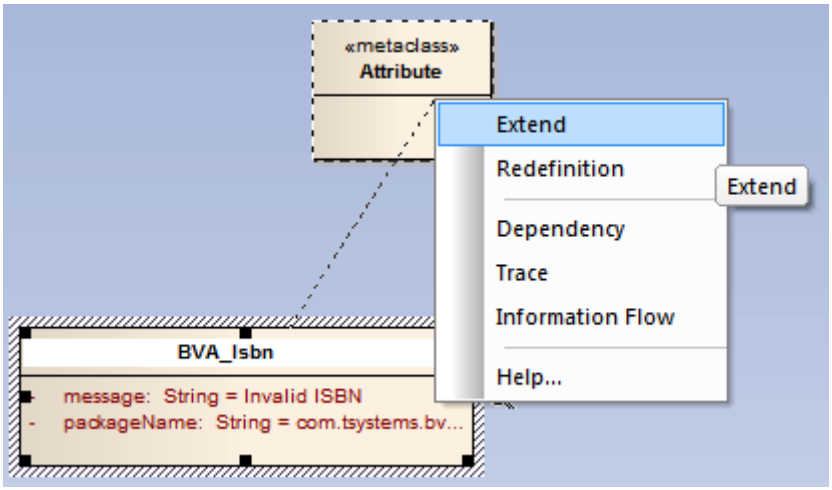


Image 17: Screenshot of extending of metaclass

Now you have to make some changes to the new profile to be able to use it in the model. Firstly you have to export this profile. Open a context menu by right-click to a empty spot in the profile diagram and select "*Save as Profile...*" to open the export dialogue. Select the saving directory "*bva.exporter/Profile/-SimpleJSR303_Bookcatalogue.xml*" (see Image 18: Screenshot of the profile export dialogue). Now you have to import this profile to the model.

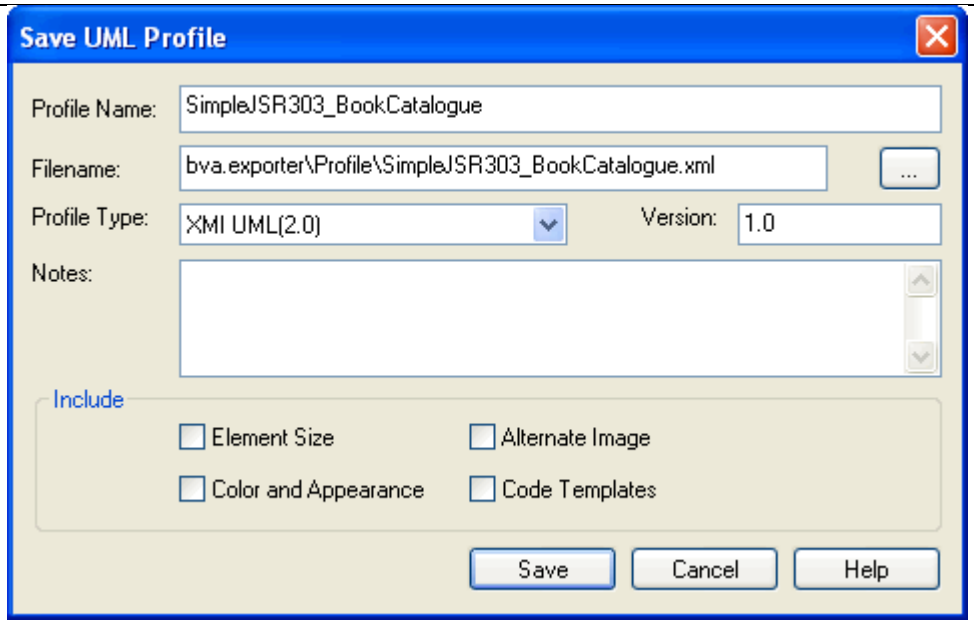


Image 18: Screenshot of the profile export dialogue

5.3.3 Using of a user-defined BVA-profile

To import the exported profile see chapter 8.5. Now you can apply the user-defined stereotype to the attribute *"Book::isbn"*. (see chapter 8.6)

5.3.4 Implementation of a user-defined validator

You also have to implement the user-defined validator in the java plattform so that the validator works precisely during the source code generation.

The BVA-stereotypes will be mapped to the actual java annotations. For this step you have to implement the appropriate java annotation *"@Isbn"* for the user-defined BVA-stereotype. The validation logic will be implemented in the separated java class *"IsbnValidator"* and not in the annotation. If the implementation already exist you have to check whether the package in the folder *"bva.generator"* is correctly configured in the build path.

```
package com.tsystems.bva.buchkatalog;

import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.*;

import java.lang.annotation.Documented;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import javax.validation.Constraint;
```

```
import com.tsystems.bva.buchkatalog.IsbnValidator;

@Target( { FIELD })
@Retention(RUNTIME)
@Constraint(validatedBy = IsbnValidator.class)
@Documented
public @interface Isbn {

    String message() default "{validator.isbn}";

    Class<?>[] groups() default {};

}
```

Image 19: source code of java annotation *"@Isbn"*

User-defined java annotations (see Image 19: source code of java annotation *"@Isbn"*) are available since *Java 1.5* The keywords *"@Target"*, *"@Retention"* and *"@Constraint"* are important.

"@Target" defines a list of java elements which can be annotated. Possible are *Type*[1], *Field*, *Method* and *Annotation_Type*[2].

"@Retention" must be set to *"Runtime"* because the annotations must be available during the runtime.

"@Constraint""(validatedBy = IsbnValidator.class)" sets the class which should be used for the validation of files.

The methods *"message()"* and *"groups()"* have to be declared in BVA. Other own methods are possible and have to be created parallel to the tagged values.

Now the validator from image 23 has to be implemented. (see Image 20: An excerpt from the source code of the class *"IsbnValidator"*) This validator implements an interface named *"ConstrainValidator"*, which specifies two types of parameters. The first parameter specifies the type of annotation and the second one the data type of the elements, which can be identified by the validator. With the method *"initialize()"* you can access all attributes, which are defined in the annotation. (For *"Pattern"* it would be for example *"regexp"* and *"flag"*). The actual validation rules will be implemented in the method *"isVa-*

[1] Class, interface, enum etc.

[2] Der Annotation_Type wird benötigt, wenn man eine Annotation mit einer weiteren Annotation auszeichnen möchte. Ermöglicht sozusagen verschachtelte Annotationen.

lid(...)". The package, included in dromedary, contains additionally a simple Prüfsummenberechnung

```
package com.tsystems.bva.buchkatalog;

import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;

import com.tsystems.bva.buchkatalog.Isbn;

public class IsbnValidator implements ConstraintValidator<Isbn,
String> {

   public void initialize(Isbn constraintAnnotation) {
      // nothing to do
   }

   public boolean isValid(String isbn,
         ConstraintValidatorContext constraintContext) {
      // Here the ISBN-13 validation rules must be implemented
      // according to Wikipedia http://de.wikipedia.org/wiki/ISBN
      if (isbn == null)
         return false;

      if (isbn.length() != 13)
         return false;

      // Calculate checksum
      // ...
   }

}
```

Image 20: An excerpt from the source code of the class *"IsbnValidator"*

Both classes will be created in the package, which was defined as *"packageName"* in the table 2. (see Table 2: Attributes of the stereotype *"BVA_Isbn"*)

5.4 Export with *"bva.exporter"*

In this example *„bva.exporter"* instead of *"common.exporter"* will be used. These adjustments are also valid for *"common.exporter"*. Before starting the exporter, you have to check whether the model from *Enterprise Architect* and the user-defined profile are correctly implemented.

5.4.1 Adjustment of "workflow.properties"

You have to make some adjustments in the file "*bva.exporter/wf/workflow.properties"*. In the first step you have to configure the model from *EA*. In the section *"Model definition"* you can reference the model. The property *"model.enterprisearchitect.file"* specifies the relative and absolute path to the *EA*-file. The property *"model.uml_file"* specifies the file, in which the XMI-file will be saved (the ending *".uml"* is intended). The property *"model.pkg"* specifies a name for the package, which contains the model (for more information see chapter 5.1) The property *"model.name"* specifies a name of the diagram. The last property *"output_slot"* is not allowed to be changed, because this property is necessary for the intern processing.

```
# Model definition
model.ea_file   = Model/BVA_Tutorial.EAP
model_file      = Model/model.uml
model_pkg       = Model/Class Model/BVATutorial
model_name      = BVATutorial
output_slot     = model
```

Image 21: Configuration of the model in the file *"workflow.properties"*

The property *"custom_profile.umlPackage"* specifies the exact name of the package in *EA* and the property *„custom_profile_name"* specifies the name of the profile in *EA*. The property *"custom_profile.uml_file"*, is very important, because it specifies the directory, which will contain an exported XMI-file. It is necessary that this file exists before beginning the next step.

Note: The property *"custom_profile.uml_file"* must contain the file, which was created as model-to-model transformation in *"bva.exporter"* and not the manual created file in the profile exporter of *EA* in chapter 5.3.1.

The property *"custom_profle.enterpriseArchitect.file"* specifies the file *"Model/BVA_Tutorial.EAP"* . This file contains the created profile.

Note: It isn't necessary to model the project model and the profile in the same file in *EA*.

```
# Custom Profile definition
custom_profile_pkg      = Model/Profile Model/SimpleJSR303_Buchkatalog
custom_profile_name     = SimpleJSR303_Buchkatalog
custom_profile_file     = Profile/SimpleJSR303_Buchkatalog.profile.uml
custom_profile.ea_file  = Model/BVA_Tutorial.EAP
custom_profile_dir      = Profile
```

Image 22: Configuration of the user-defined profile in the file *"workflow.properties"*

5.4.2 Starting the export process

All preparations have been done and you can start the export process now.

To start the export process of the profile make right-click on the file *"bva.exporter/wf/profileExportWorkflow.mwe"* in the package explorer window and than click *"Run As → MWE Workflow"*. The status of progress is visible in the console-window.

When the workflow is completed, you will see the following message in the console window:

75061 INFO WorkflowRunner - workflow completed in 74026ms!

Note: When you are using a trial version of *Enterprise Architect* a licensing window opens up in the background, which you have to confirm by clicking *"Continue Trial"*. With Version 8.0 or higher you have to wait a while (10s) before confirming *"Continue Trial"* to avoid an error message.

To start the export of the model make right-click on the file *"bva.exporter/wf/modelExportWorkflow.mwe"* and than click *"Run As → MWE Workflow"*.

All required models are now available in the XMI-format, which is compatible with the EMF-format, and the generation of the source code can be started.

5.5 Code generation with "*bva.generator*"

Start *"bva.generator/src/main/resources/workflow.oaw"* in the same way as before. Abstract and interface classes with packages have been generated to the folder *"bva.generator/src/generated/java"*. Implementation classes have been generated to the folder "*bva.generator/src/main/java"*. These files won't be overwritten by all next generation processes to protect manual changes in the source code.

The status of progress is visible in the console window. When the workflow is successful completed, you will see the following message in the console window:

```
INFO - Written 9 files to outlet [default](./src/generated/java)
INFO - workflow completed in 3335ms!
```

To get to know whether the generation were successful, you can open the generated class *"AbstractBook"* and check the user-defined validator *"BVA_Isbn"* :

```
@com.tsystems.bva.buchkatalog.Isbn(message = "Invalid ISBN")
private java.lang.String isbn = null;
```

You can also check the validators *"BVA_Pattern"* in the classes „*AbstractAuthor*" and *"AbstractPublisher"*:

```
@javax.validation.constraints.Pattern(
   message = "Invalid PLZ", regexp = "[0-9]{5,5}")
private java.lang.String plz = null;
```

5.6 Annotation to the example

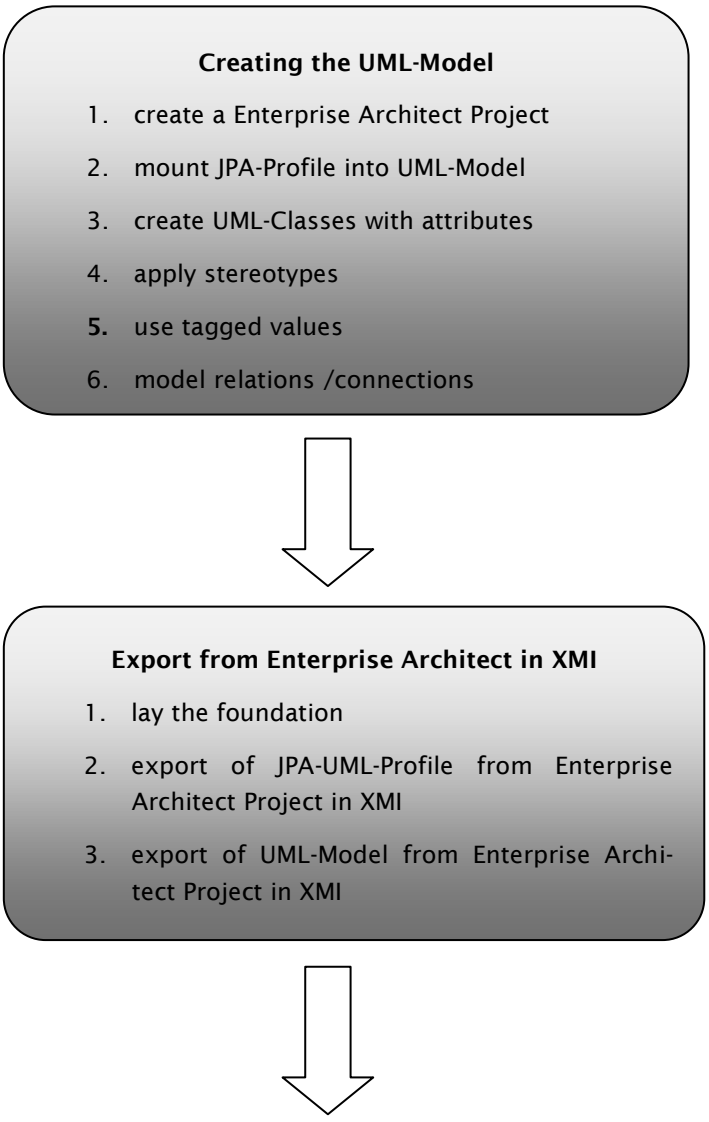
It is unhandy to define two equal validation rules for the attributes *"zip"*. It's better to use a user-defined validator named *"BVA_Zip" t*o ensure the consistence of the validation rules for zip codes. This approach has many advantages. The first advantage is that you can use this validator in other projects without any difficulty. The second one is that one change in

Dieses Vorgehen hätte mehrere Vorteile: Zum einen kann der Validator problemlos in späteren Projekten wiederverwendet werden und zum anderen ist bei einer Änderung sichergestellt, dass diese an allen Stellen an denen ein *BVA_Plz* Validator eingesetzt wird auch tatsächlich zum Tragen kommt.

6 Use of the JPA-Generator shown with an example

This chapter is a tutorial, it's goal is to deliver insight into the MDA-Generator. With this easy example you'll quickly learn the required steps to generate code. Starting with the creation of an UML-model using *Enterprise Architect* to completely generated javacode and finally to a generated DDL-File.

Overview



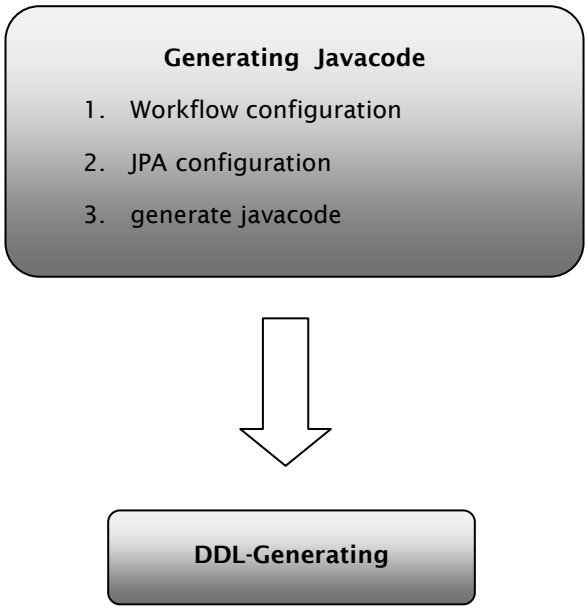


Image 23: Tutorial overview

6.1 Creation of sample model

6.1.1 create a Enterprise Architect Project

To be able to create a UML-Model it's necessary to start by creating a new Enterprise Architect Project with the name *"Tutorial"*. A detailed explanation can be found in chapter 8.1. The Project file should be placed at the directory *"/<Your Workspace>/jpa.exporter/Model/"*.

The classes *"Class1"* to *"Class3"* and the *"Interface"* in the *"Resources View"* should be deleted. They get created by Enterprise Architect when creating a new Project. Please rename the Package System and the Diagram System into *"JpaTutorial"*.

6.1.2 mount JPA-Profile in UML-Model

To be able to use the JPA-UML-Profile in the model, it can be mounted as explained in chapter 8.6. The required XML-File *"SimpleJSR220.xml"* is placed inside the droMDAry Package in the JPA profile folder.

6.1.3 create UML-classes with attributes

Now the model, which in this example is a simple entity-model, can be created. First the class *"Author"* with the attributes *"name"*, *"lastname"*, *"street"* of type *"String*" and *"age"* and *"zip"* of type *"Integer"* must be created. The workflow to create and edit classes and attributes can be found in chapters 8.2 and 8.3.

Further classes and their attributes that have to be created (they all got the scope "private"):

- Class: Book
  - Attribute: bookId, type int
  - Attribute: circulation, type int
  - Attribute: title, type String
  - Attribute: isbn, type int
- Class: Publisher
  - Attribute: publisherName, type String
  - Attribute: street, type String
  - Attribute: zip, type int
- Class: Paperback
  - Attribute: format, type String

Now all classes for the sample model are created. Each class is going to be mapped, with all attributes one to one, to the database. What means each class equates a table in the database.

6.1.4 Apply Stereotypes

Now you can apply the stereotypes from the JPA-UML-profile to the UML-elements. Because the MDA-cartridge works by the KISS-Principle (Keep it simple and stupid), not much specification is necessary to generate a executable java persistence layer.

The stereotype *"JPA-Entity"* has to be applied to every class, so these classes can be mapped to the database. Of course classes, which aren't entities, are editable in the model and won't be mapped to the database. In this case you don't have to apply the stereotype *"JPA_Entity"* to the class.

All other stereotypes are optional. What means if there are no other stereotypes applied, the MDA-cartridge creates the annotations and ID's impli-

cit,which are required for the persistence layer (this is called the **standard case**). Thereby the modeling act is simplified a lot, what disburdens the developer.

When using the standard case, the javacode for the entity-classes and attributes will be generated as shown below:

- The class name in the model matches the table name in the database
- The database scheme matches the scheme, defined in the file *"jpa.properties"*
- Each attribute in the model is mapped as an attribute to the database (annotation *"JPA_Column"*)
- The name of attributes in the model matches the name in the database (annotation *"JPA_Column"* without tagged value)
- An attribute named *"id"* of type integer including the annotation *"@id"* is created for each class (primary key)

When you don't want to apply the standard case, optional stereotypes from the JPA-UML-profile have to be used. If not done already, the stereotype *"JPA-Entity"* has to be applied to all classes in the sample model. How this is done in *Enterprise Architect* is shown in chapter 8.6.

In the sample model the class *"Book"* gets an explicit defined ID and thereby the optional stereotype *"JPA_Id"*. An ID for all other classes is generated implicit, because they are strictly necessary for a clear identification of one dataset. Here it's necessary to define the ID explicit, because in this example the class should get a functional composed primary key. To put this into effect the stereotype *"JPA_Id"* has to be applied to the attributes *"bookId"* and *"circulation"*. As soon as there is more than one applied stereotype *"JPA_Id"* in one class, the MDA-cartridge recognizes automatically that a primary key class has to be generated. The primary key class exists for a correct identification of a dataset, that only can be identified by several primary keys.

In example for a further optional specification the stereotype *"JPA-Column"* is applied to the attribute *"street"* of the class *"Author"*. Thereby is explicit defined that the attribute will also be used as an attribute in the database. When this annotation won't be changed, it'd be useless, because the standard case already covers this. To make it useful, a tagged value has to be used (see chap-

ter 6.1.5). As soon as the *"JPA_Column"* annotation is applied to one attribute in a class, all other attributes, that should be shown up as attributes in the database, have to be applied to the annotation too. Otherwise the attributes will be ignored in the database.

6.1.5 Using tagged values

The database name of the attribute *"street"* of class *"Author"* should in this example not match the name of the attribute in the model, instead of this the name should be *"str"*. Now tagged values have to be used. Some stereotypes from a profile have tagged values. The tagged value *"name"* now has to be set to *"str"* inside the Tagged Values View of *Enterprise Architect*. This step is explained in chapter 8.7.

6.1.6 Model relations

The last step of creating a UML-Model is to set the relations between the single classes in the model. How this is done is explained in chapter 8.4. The classes correlate to each other as shown below:

- Author:Publisher / *:* (ManyToMany) / Typ: Association
- Author:Book / 1:* (OneToMany) / Typ: Aggregation
- Book:Paperback / Typ: Conduction (Paperback is conducted from book)

For the relation "Author:Publisher" it's not necessary to specify the multiplicity, because the MDA-cartridge assumes an implicit "*.*" (ManyToMany) relation when no multiplicity is specified.

The completed UML-Model should now look like this:

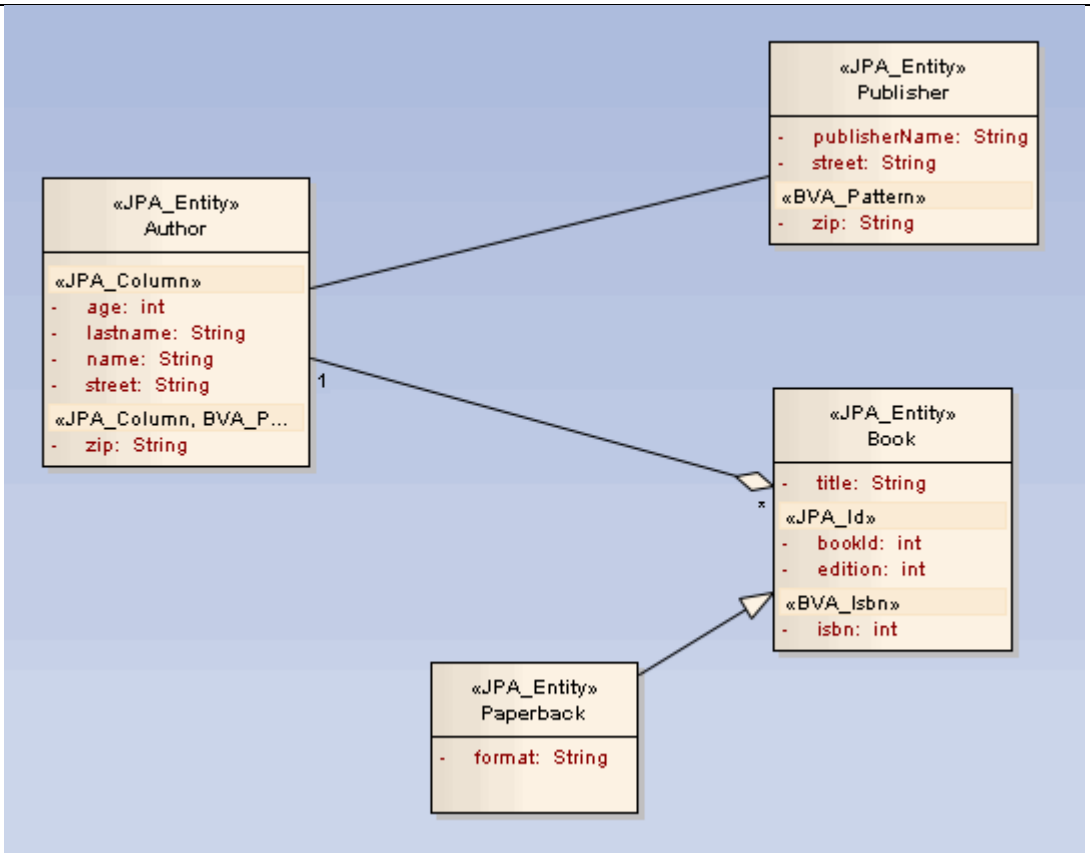


Image 24: UML sample model

6.2 Export workflow from Enterprise Architect into XMI

The Export process takes place in the "*Modeling Workflow Engine (MWE)*" in the project *"jpa.exporter"*.

Furthermore it's recommended to navigate with the "*Navigator View*" through the workspace. This menu can be opened at "W*indow – Show View – Navigator*".

6.2.1 Lay the foundation

Out of the box the file *"SimpleJSR220-profile.eap"* is placed in the directory *"model"*. In case the file is not in this directory it's required to copy it from the profile folder of the droMDAry package, into the directory *"/<your workspace>/jpa-exporter/Model"*. This file contains the whole JPA-UML-profile and will be converted to XMI-format later. You may wonder why the profile is also included in a separate *Enterprise Architect* project file, although it's already mounted into the UML-Model. This is because the mounted profile in the UML-

model is only used as an UML-Add-on and won't be converted as a separated profile, when converting to XMI. What means it wouldn't be usable in MWE. In addition the use of several profiles should be possible. So to have a clean way of putting this into effect it's necessary that every UML-Profile has its own *Enterprise Architect* project file.

6.2.2 Export of JPA-UML-Profile from Enterprise Architect in XMI

MWE offers so called workflow-files, which define the process of generating or converting. To start the generating process, the corresponding workflow file has to be executed. The corresponding workflow file for the export-/converting process is placed in the directory *"/<your workspace>/jpa-exporter/wf/profile.mwe"*.

Every MWE-project also contains a properties file, which includes a configuration of the generating process. In the project *"jpa.exporter"* it's placed at *"/<your workspace>/jpa-exporter/hf/workflow.properties"*. For the export process of the UML-JPA-profile, the following settings are necessary:

```
##################### Konfiguration des UML-Profils #####################

# Datei des Enterprise Architect Projekts.
profile.enterpriseArchitect.file = Model/SimpleJSR220-Profile.eap

# Datei, in welche das in XMI transformierte UML-Profil abgelegt wird.
# ACHTUNG: Diese Datei muss immer exisitieren (es darf sich auch um eine leere Datei handeln).
profile.emfUml2.file     = Profile/SimpleJSR220.profile.uml

# Pfad zum Package, in dem das Diagramm abgelegt ist.
# Der Pfad ist im Enterprise Architect im Project Browser zu erkennen.
profile.umlPackage   = Models/Profile Model/SimpleJSR220

# Name des Enterprise Architect Diagramms, welches sich in profile.umlPackage befindet.
profile.name     = SimpleJSR220

# Verzeichnis, in dem sich das Profil befindet.
profile.export.dir   = Profile

# Source Ordner für generierte Artefakte
generator.output.src.dir         = src-gen
```

Image 25: *"workflow.properties"* –configuration of the profile export process

Furthermore the property *"project_root"* (Basic directory) should be empty.

Out of the box this sample configuration and the configuration for the model is already contained in the file *"workflow.properties"*. Further information can be found in chapter:

Also out of the box the project *"jpa.exporter"* contains the file *"SimpleJSR220-Profile.eap"*. Through the export process the file *"SimpleJSR220.profile.uml"*,

which has the converted XMI-format, will be generated. The export- / converting process is started be right-clicking on *"profileExportWorkflow.mwe – Run As – MWE Workflow"*. If the Trial Version of *Enterprise Architect* is installed, a dialog named *"Evaluation Version of Enterprise Architect"* shows up. It can be confirmed by clicking *"Continue Trial"*. This dialog shows up because in the process the file *"SimpleJSR220-Profile.eap"* is accessed and converted, what requires an executable *Enterprise Architect* Version. Information about the process will be displayed In the console view of *Eclipse*. A successful process ends with the console output:

```
… INFO  WorkflowRunner      - workflow completed in … ms!
```

6.2.3 Export of UML-Model from Enterprise Architect Project in XMI

For the export process of the UML-model, the following setting are necessary in the file *"workflow.properties"*:

```
##################### Konfiguration des UML-Modells #####################

# Datei des Enterprise Architect Projekts.
model.enterpriseArchitect.file  = Model/jfs.eap

# Datei, in welche das in XMI transformierte UML-Modell abgelegt wird.
# ACHTUNG: Diese Datei muss immer exisitieren (es darf sich auch um eine leere Datei handeln).
model.emfUml2.file      = Model/model.uml

# Pfad zum Package, in dem das Diagramm abgelegt ist.
# Der Pfad ist im Enterprise Architect im Project Browser zu erkennen.
model.umlPackage        = Models/Class Model/javaforum

# Name des Enterprise Architect Diagramms, welches sich in model.umlPackage befindet.
model.name      = javaforum

# Interner Slot für oAW, in welchem sich das Modell im EMF-Format befindet.
output_slot     = model
```

Image 26: *"workflow.properties"* –configuration of the model export process

The sample model was created in chapter 6.1 and should already be in the directory. This file is going to be converted into XMI and placed in the file *"model.uml"*.

By right-clicking on *"modelExportWorkflow.mwe – Run As – MWE Workflow"* the export- / converting process starts. The *Enterprise Architect* dialog shows up

again, confirm by clicking *"Continue Trial"*. The export process was successful if the *Eclipse* console output shows:

```
… INFO  WorkflowRunner      - workflow completed in … ms!
```

6.3 Generating javacode

The generating process will be done in the project *"jpa.generator"*. Like in the project *"jpa.exporter"* there are properties files. These and also the workflow file, that starts the generating process, can be found in the directory *"/<your workspace>/jpa.generator/src/main/resources"*. Out of the box the properties files are already configured. To get a quick idea, the most important settings are explained in the following. As in the export process all settings are made in properties files.

The Generator contains two properties files:

- *"workflow.properties"* -> Configuration of the workflow
- *"jpa.properties"* -> JPA-settings

6.3.1 Workflow Configuration

The file *"workflow.properties"* has two sections. The first section is about the configuration of the JPA-generator for example configuring the output directory and the UML-models and profiles, which got generated through the export process and are now used as input for the JPA-generator (see Image 27: Configuration of the JPA-cartridge in the file *"workflow.properties"*)

```
##################### Konfiguration der JPA-Cartridge ####################
# Basisverzeichnis
basedir = .

# Encoding Einstellung für das Einlesen von Daten.
file.encoding = ISO-8859-1

# Encoding Einstellung für das Ausgeben von Daten.
file.encoding.output = ISO-8859-1

# Ordner für Artefakte, die generiert und nicht modifiziert oder gelöscht werden dürfen.
outlet.src.dir =${basedir}/src/generated/java

# Ordner des Artefakts persistence.xml.
outlet.persistenceXML =${basedir}/META-INF/persistence.xml

# Ordner für Interface Artefakte.
outlet.src.interfaces.dir =${basedir}/src/generated/java

# Ordner für Kofigurationsdateien.
outlet.res.dir =${basedir}/src/generated/resources
outlet.res.once.dir = ${basedir}/src/main/resources

# Ordner für Artefakte, die später durch die Entwickler angepasst werden
# können (Implementierungsklassen).
outlet.src.once.dir = ${basedir}/src/main/java

# Datei des UML-Modells
model.file = ${basedir}/../jpa.exporter/Model/model.uml
```

Image 27: Configuration of the JPA-cartridge in the file *"workflow.properties"*

The second section shows the configuration of the co-used javabasic-cartridge. Further information about the configuration of the javabasic-cartridge can be found at: http://fornax.itemis.de/confluence/display/fornax/JavaBasic+(CJB)

```
##################### Konfiguration der mitverwendeten Javabasic-Cartridge ####################

# Mehr Informationen zu diesen Properties findet man unter
# http://fornax.itemis.de/confluence/display/fornax/JavaBasic+(CJB)

# Inhalt des Headers für generierte Artefakte
type.header.text = '/* (c) 2009 - Your Copyright here \n\
    * E-Mail: no-reply@nowhere.com\n\
    * \n\
    * Diese Datei wurde generiert und darf weder modifiziert, noch gelöscht werden!\n\
    * Änderungen bitte stets im Modell oder Profil umsetzen!\n\
    */\n'

# Inhalt des Footers für generierte Artefakte
type.footer.text = ''

# Diese Property wird von der Javabasic Cartridge benötigt.
annotation.source.key = ''

# Generierung der Methoden equals, hashcode und toString in
# abstrakten Entity-Klassen.
# 'true'  - Methoden werden generiert.
# 'false' - Methoden werden nicht generiert.
use.overridden.equals.hashcode.toString = 'true'

# Java Version (Generic Support 5 oder höher)
java.version = 5

# Generierung von abstrakten Methoden in einer abstrakten Klasse.
# 'use_abstract_method' - Abstrakte Methode in abstrakter Klasse generieren.
# 'none'                - Keine abstrakte Methode generieren.
interface.operation.implementation.strategy = 'none'

# Generierung der SerialVersionUID.
# 'true'  - SerialVersionUID wird generiert.
# 'false' - SerialVersionUID wird nicht generiert.
javabasic.generateSerialVersionUID = 'true'

# Entscheidung ob eine Liste als 'set' oder als 'list' generiert wird.
# 'order'  - UML-Flag "isOrdered" wird unterstützt.
# 'unique' - UML-Flag "isUnique" wird unterstützt.
list.set.property = 'unique'
```

Image 28: Configuration of the Javabasic-cartridge in the file *"workflow.properties"*

6.3.2 JPA Configuration

The properties file *"jpa.properties"* is used for the JPA configuration. It defines the path of the *"persistence.xml"*, further settings about mapping the objects, settings about the database scheme of the tables and settings about the package names of to be generated DAO-files (see Image 29: JPA configuration in the file *"jpa.properties"*). More technical details are listed in chapter

```
##################### JPA Konfiguration ####################

# Pfad der persistence.xml ausgehend von src/generated/java.
persistence.xml.path = ../../main/java/META-INF/persistence.xml

# Properties für persistence.xml.
persistence.xml.properties = <!-- Scan for annotated classes and Hibernate mapping XML files -->\n\
                <property name=\"hibernate.archive.autodetection\" value=\"class, hbm\"/>\n\
                <!-- SQL stdout logging\n\
                <property name=\"hibernate.show_sql\" value=\"true\"/>\n\
                <property name=\"hibernate.format_sql\" value=\"true\"/>\n\
                <property name=\"use_sql_comments\" value=\"true\"/>\n\
                -->\
                <property name=\"hibernate.connection.driver_class\" \n\
                          value=\"org.apache.derby.jdbc.EmbeddedDriver\"/>\n\
                <property name=\"hibernate.connection.url\"\n\
                           value=\"jdbc:derby:miadb;create=true\"/> \n\
                <property name=\"hibernate.connection.username\" \n\
                          value=\"sa\"/>\n\
                <property name=\"hibernate.c3p0.min_size\" \n\
                          value=\"5\"/>\n\
                <property name=\"hibernate.c3p0.max_size\" \n\
                          value=\"20\"/>\n\
                <property name=\"hibernate.c3p0.timeout\" \n\
                          value=\"300\"/>\n\
                <property name=\"hibernate.c3p0.max_statements\" \n\
                          value=\"50\"/>\n\
                <property name=\"hibernate.c3p0.idle_test_period\" \n\
                          value=\"3000\"/>\n\
                <property name=\"hibernate.dialect\" \n\
                          value=\"org.hibernate.dialect.HSQLDialect\"/>\n\
                <property name=\"hibernate.hbm2ddl.auto\" value=\"create-drop\"/>

# Datenbankschema
table.schema = APP

# Packagename für DAO relevante Java Klassen.
dao.package.name = dao
```

Image 29: JPA configuration in the file *"jpa.properties"*

When both properties files are finally configured and the *Enterprise Architect* model and profile have been converted in XMI, the generating process can be started. The model and profile doesn't have be moved to the generator project, they are read out directly of the exporter project.

6.3.3 Generate Javacode

Execute the file *"workflow.oaw"* by right-clicking on it "*Run As – MWE Workflow*". In the *Eclipse* console output it can be checked if the workflow was executed correctly. If that's the case, for every entity has been generated an interface, an abstract class and an implementation class for own methods. Now there are two kinds of files generated, files that are allowed to be edited by the developer and files that aren't. The files that aren't can be found in the directory *"src/generated/java/jpatutorial"*. All interfaces and abstract classes are placed there. When these files need to be edited, the UML-model need to be

changed, after that it's necessary to repeat the whole generating process (incl. export process). In the directory *"src/main/java/jpatutorial"* all the implementation classes have been placed. They only get generated once. When the generating process is executed again these files can't be overwritten, so no changes made by the developer get lost.

If the implementation classes should be generated again, the package "JpaTutorial" simply has to be deleted. Then these files will be created again.

To understand how the generator converts the UML-model into javacode, it will be explained with the abstract classes *"AbstractAuthor"* and *"AbstractBook"* (see Image 30: Generated class *"AbstractAuthor"*)

```
@Entity
@Table(name = "Autor", schema = "APP")
public abstract class AbstractAutor

implements Autor {

    @Id
    private int id;

    @Column()
    private java.lang.Integer alter = null;

    @Column()
    private java.lang.String nachname = null;

    @javax.validation.constraints.Pattern(message = "Invalid PLZ", regexp = "[0-9]{5,5}")
    @Column()
    private java.lang.String plz = null;

    @Column(name = "str")
    private java.lang.String strasse = null;

    @Column()
    private java.lang.String vorname = null;

    @OneToMany(targetEntity = CommonTutorial.AbstractBuch.class)
    @JoinColumn(name = "Autor_id", nullable = false)
    public java.util.Set<CommonTutorial.Buch> buch = null;

    @OneToOne(targetEntity = CommonTutorial.AbstractVerlag.class)
    public Verlag verlag = null;
```

Image 30: Generated class *"AbstractAuthor"*

To support all JPA-annotations, the import has been generated from *"javax.persistence.*"*.

In the UML-model the class *"Author"* has been tagged with the stereotype *"JPA_Entity"*. That's why the cartridge creates the annotation *"@Entity"* and so

the class can be mapped correctly to the database, the generator creates the annotation *"@Table"* with the tagged values *"name"*, which describes the table name, and the annotation *"scheme",* which reads the scheme out of the *"jpa.properties"* file.

For the class *"Author"* no ID has been defined in the UML-model. So the casrtridge automatically creates an attribute of type integer and tags it with the annotation *"@Id".*

The attributes, that are tagged with the stereotype *"JPA_Column"* in the model, have been tagged with the annotation *"@Column"*. So these attributes are available later in the database. Especially for the attribute *"street"* in the UML-model the tagged value *"name"* has been set to *"str"*, to set the name of the attribute in the database to *"str".* That's why in the javacode the annotation *"@Column"* with the tagged value *"name"* has been generated for the attribute *"street".*

The attributes *"book"* and *"publisher"* should show the modeled relation in Java. The "Author to Book" relation has been modeled as a "1;*" relation. Because of this the cartridge tags the annotation *"@OneToMany"* with the required tagged value *"targetEntity"*, that matches the target class of the relation. The annotation *"@JoinColumn"* is generated for a correct mapping of the relation to the database.

The same applies for the relation "Author to Publisher", just with a "*:*" (ManyToMany) relation.

Then the Getter and Setter, the "Equals" and "Hash Code" for comparing the objects and the "To String" method have been generated.

```
@IdClass(BuchPk.class)
@Entity
@Table(name = "Buch", schema = "APP")
public abstract class AbstractBuch

implements Buch {

    @Id
    @Column
    private java.lang.Integer auflage = null;

    @Id
    @Column
    private java.lang.Integer buchId = null;
```

Image 31: Generated class *"AbstractBook"*

In the generated class *"AbstractBuch"* the annotation *"@IdClass"* shows that an composed functional primary key exists and that a primary key class *"BookPk"* has been generated (see Image 31: Generated class *"AbstractBook"*). The composed primary key consists of the attributes *"bookId"* and *"circulation"* indicated by the annotation *"@Id"*. Also it's cognizable, that every attribute has been annotated with the annotation *"@Column"*, although it hasn't been defined in the model (standard case).

```
/**
 *
 * Primary Key Class
 */
public class BuchPk implements Serializable {

    private final static long serialVersionUID = -2093529850L;

    @Id
    @Column
    public Integer auflage;

    @Id
    @Column
    public Integer buchId;
```

Image 32: Generated primary key class *"BookPk.java"*

In the primary key class all attributes are once again listed with the annotations *"@Id"* and *"@Column"*. Through this the composed functional primary key gets

mapped correctly to the database. This strategy derives from the suggested mapping-strategies in the Book *"JAVA Persistence with Hibernate"*.

The required classes for the standard CRUD-Operations have been generated in the package DAO. Therefore the class *"GenericJpaDAO"* contains all important methods. All other classes need in turn the class *"GenericJpaDAO"*, to be able to execute the operation. The DAO Package is build, like described by Hibernate. The whole package is generated automatically and isn't influenced by the UML-model.

The primary key class *"bookPk"*; required by the composed Primary Key in the class *"book"*, is generated in the same package *"JpaTutorial"*, that also contains the abstract classes and interfaces. The primary key class contains all important details that are required for a correct mapping. It's is built like recommended by Hibernate.

Further information about DAO and the primary key class can be found in chapter .

6.4 DDL-Generating

The generated DDL-file can be used for persistencing the entities to a random database. With this step all necessary steps from modeling a UML-model, to generating a persistence layer in Java and finally to generating a DDL-file are completed.

How the DDL-file is persistenced to a database is not part of this paper and won't be explained. Depending on the database type this step varies. But clearly persistenceing will be done quickly.

Hibernate-Tools are used for generating. Required libraries are already in the *"lib"* directory and integrated in the project.

The *"persistence.xml"* which is required for generating the DDL-file has been generated automatically in the previous process. It's placed in the directory *"/<your workspace>/jpa.generator/src/main/java/META-INF"*.

For generating the DDL-file, the file *"build.xml"*, which is placed in the basic directory of *"jpa.generator"*, needs to be executed by right-clicking – *Run As* – *Ant Build*. Caution: There is also the possibility to *"Ant Build…"*, please only use *"Ant Build"*

After that the file *"sql.ddl"* will be generated in the directory *"/<your workspace>/jpa.generator/ddl"*. This file should have the following content:

```
create table APP.Autor (id integer not null, alter integer, nachname varchar(255), plz integer, str varchar(255), primary key (id));
create table APP.Buch (DTYPE varchar(31) not null, buchPk varbinary(255) not null, auflage integer not null,
    buchId integer not null, isbn integer, titel varchar(255), format varchar(255), Autor_id integer not null,
    primary key (buchPk, auflage, buchId));
create table APP.Verlag (id integer not null, plz integer, strasse varchar(255), verlagName varchar(255), primary key (id));
create table Autor_Verlag (Autor_id integer not null, verlag_id integer not null, primary key (Autor_id, verlag_id));
alter table APP.Buch add constraint FK1FC4183D6B4AC3 foreign key (Autor_id) references APP.Autor;
alter table Autor_Verlag add constraint FK77CF8A6B3D6B4AC3 foreign key (Autor_id) references APP.Autor;
alter table Autor_Verlag add constraint FK77CF8A6BCEA4A091 foreign key (verlag_id) references APP.Verlag;
```

Image 33: Generated DDL-File

Now the end of the tutorial is reached. If you consider how fast a complex persistence layer has been generated in Java by a UML-model, you can roughly estimate, what huge advantage complex software projects can take by using this cartridge.

7 Tutorial for the common use of the JAP- and BVA Cartridge

This chapter shows you by a small example how the JPA- and the BVA-Cartridge can be used together. To do this a new cartridge, called "common-generator" has been developed.

This tutorial only explains parts detailed, that haven't been explained in the previous chapters.

As shown for the projects *"jpa.exporter"*, *"jpa.generator"*, *"bva.exporter"*, *"bva.generator"*, there is a common-generator project with *"common.exporter"* and *"common.generator"*. Both projects have to be imported in to the workspace.

UML-Model

In the previous tutorial a UML-model has already been created, so the needed UML-model is already done. It's placed in the droMDAry Package inside the directory model of the project *"common.exporter"*.

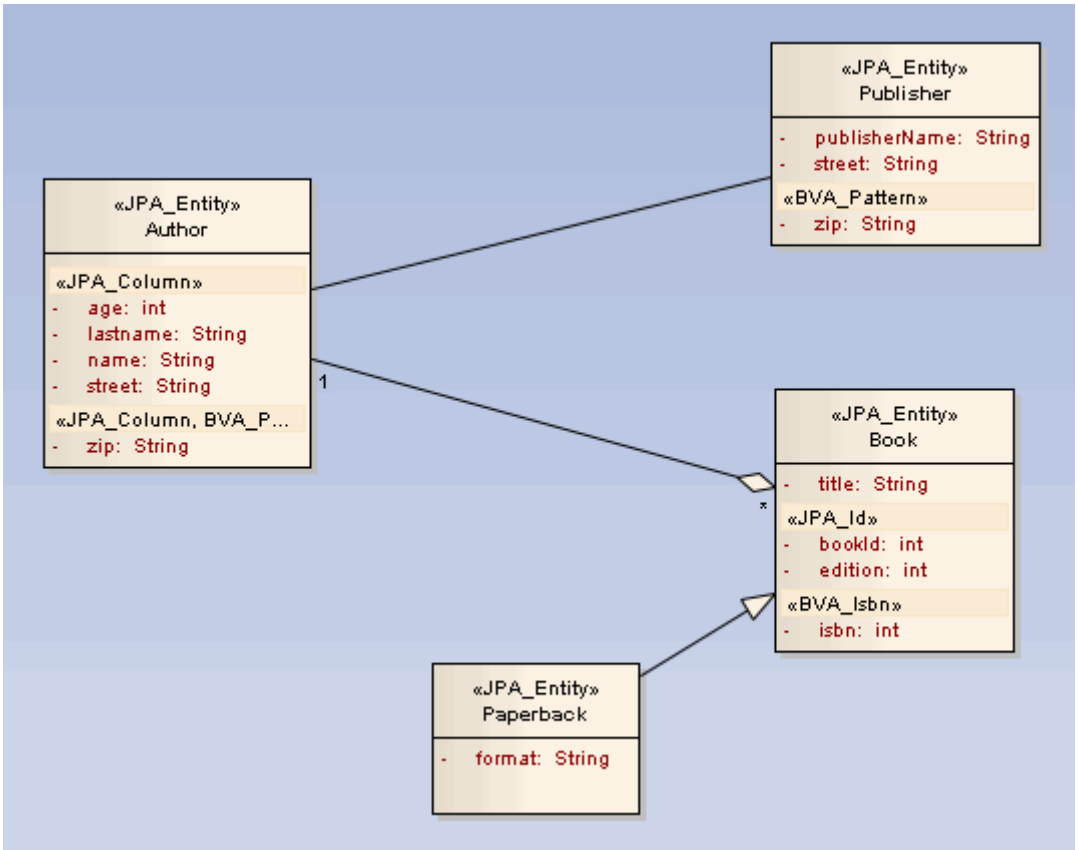


Image 34: UML sample for the common cartridge

How to use the *Enterprise Architect* and further information can be found in chapter 8. Detailed information on the topic Bean Validation API (BVA) can be found in Mr. Renz's diploma thesis.

7.1 Exporting from Enterprise Architect into XMI

The export process is done inside the project *"common.exporter"*.

Out of the box the file *"workflow.properties"* is already configured and placed in the project *"common.exporter"*. Basically the configuration is done exactly like in chapter 6.2.1. The difference is, that now three profiles have to be configured. One profile for the JPA-Cartridge, one for the BVA-Cartridge an another one for the customized Validator "Isbn".

So if you're using a Trial Version of *Enterprise Architect* the licensing window shows up three times.

After the files *"profile.mwe"* and *"workflow.mwe"* had been executed successfully, the profile and the model have been exported from *Enterprise Architect* into XMI-Format.

7.2 Generating Javacode

Because now both Cartridges, the JPA and BVA, can be used in the file *"workflow.properties"* in the project *"common.generator"*, gives you the possibility to turn the cartridges on and off (see…)

```
# JPA-Cartridge benutzen -> generate.jpa = true
# JPA-Cartridge nicht benutzen -> generate.jpa = false
generate.jpa = true

# BVA-Cartridge benutzen -> generate.bva = true
# BVA-Cartridge nicht benutzen -> generate.bva = false
generate.bva = true
```

Image 35: Configuration possibilities, which cartridge is used in the *"common.generator"* project

Note: For now the common cartridge has only be implemented to use both cartridges at the same time. The feature to use only one of the two cartridges is placed in preparation to a later release. Because of time constraints only the simultaneously use of both cartridges was tested.

All other settings are already configured. Further information can be found in chapter 6.2.2 there is the *"jpa.generator"* explained.

The *"jpa.properties"* ALWAYS have to be configured in the *"jpa.generator"* project, even when using the *"common.generator"* project.

Like in the *"jpa.generator"* project the build file, that is placed in the *"common.generator"* directory, can be used to generate the DDL-file. More about that is explained in chapter 6.4 in the diploma thesis.

After the file *"workflow.mwe"* had been executed successful and both cartridges have been used, the file *"AbstractAuthor"* shows the following lines:

```
@javax.validation.constraints.Pattern(message = "Invalid PLZ", regexp = "[0-9]{5,5}")
@Column()
private java.lang.String plz = null;
```

Image 36: Generated attribute in the class *"AbstractAuthor"*

Like tagged in the model, for the attribute *"zip"* two annotations have been generated: one by the BVA- and another one by the JPA-cartridge.

The implementation of the customized BVA-validator "Isbn", is placed in BVA-cartridge in the package *"com.tsystems.bva.Bookcatalogue"*.

8 Important information about modeling with Enterprise Architect

This chapter shows the most important steps, to quickly create a UML-model compatible to the MDA-cartridge. There are probably several ways of doing that, this chapter only shows one way. In my opinion the simplest one. A documentation about Enterprise Architect is available at the *Sparx Systems* homepage: http://www.sparxsystems.com/uml_tool_guide/index.html

Create a new Enterprise Architect Project

To create a new UML-model, first of all a *Enterprise Architect* project is required. Go to "*File – New Project…*" The dialog "*New Project*" shows up, first you have to select a directory to save the project. Then the dialog *"Select model(s)"* shows up (see Image 37: Selection of the UML model). To create a compatible UML-model you have to select *"Class"* and confirm by clicking *"OK"*.

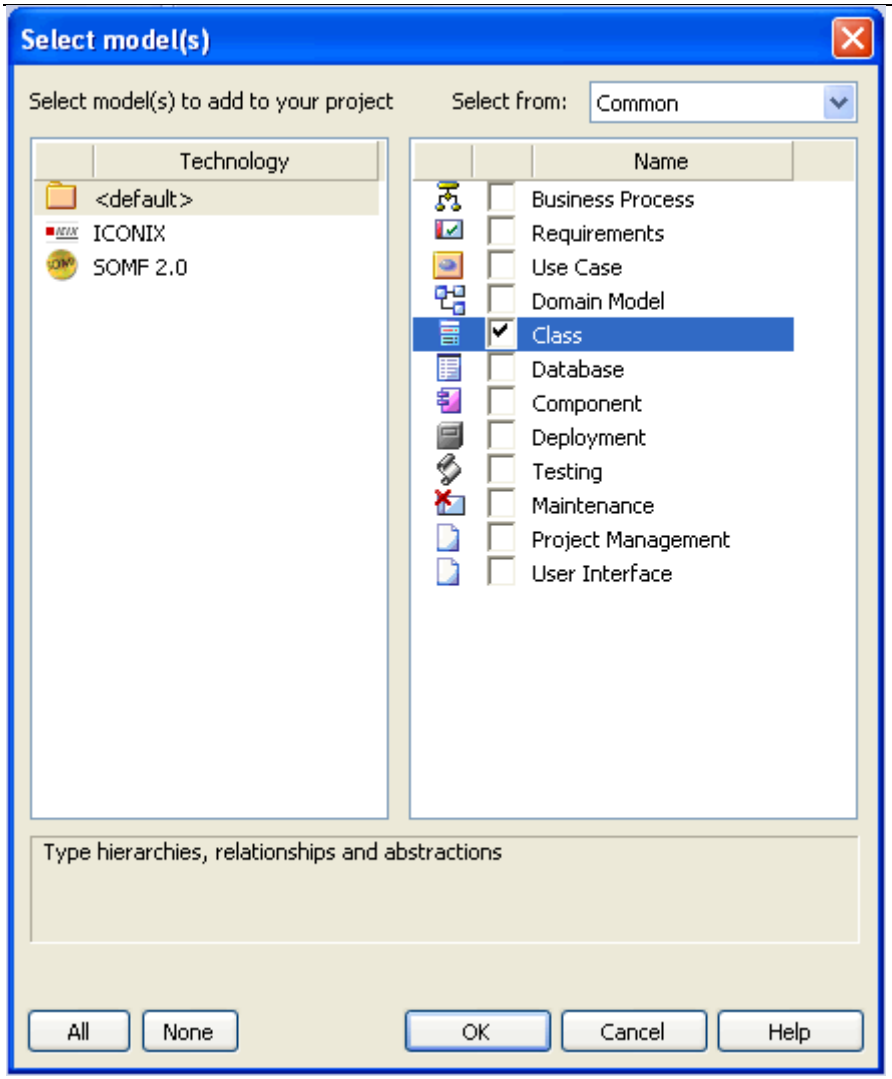


Image 37: Selection of the UML model

Now in the *"Project Browser"* a new package appears, by opening it the package *"Class Model"* can be seen. Open this package and the following package *"system"* to be able to open and edit the diagram *"System"*.

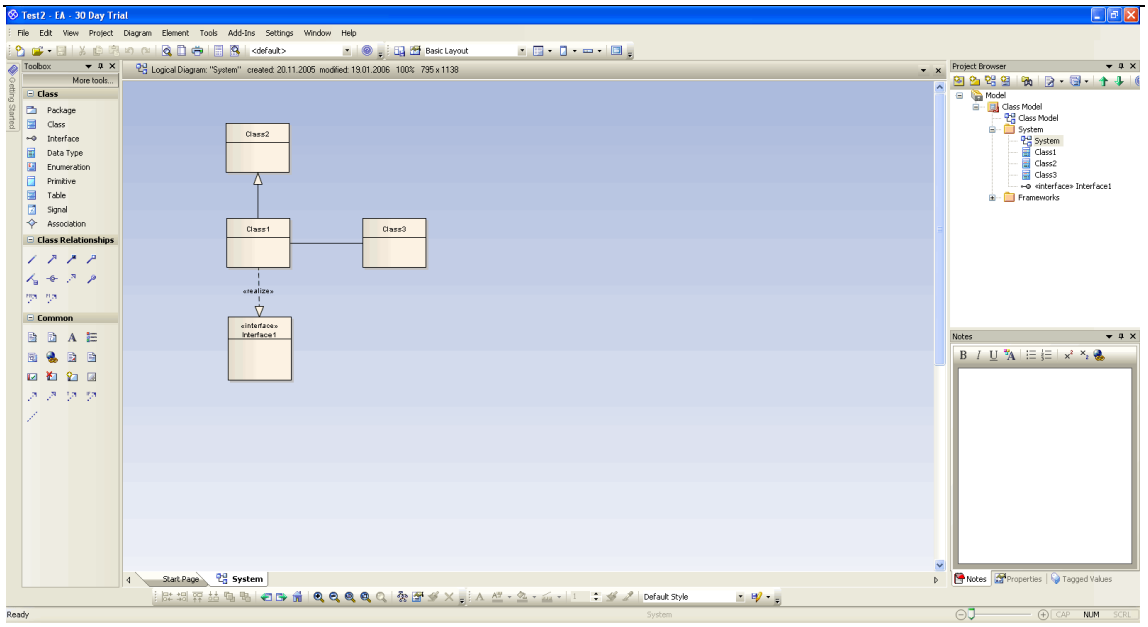


Image 38: New *Enterprise Architect* project

The created packages, diagrams and classes can now be edited and renamed. To change the name click right on the package- diagram name and select *"Properties"*. There is the entry *"name"*, which can be changed. This symbol means a package and this symbol a diagram.

8.1 Create a new UML-Class

To create a new UML-class use the *"Toolbox View"* on the left side. Simply click *"Class"* and drag it to the diagram. Afterwards the class dialog shows up. All setting for this class can be done there. Stereotypes can only be applied after the profile has been mounted. This process is explained in chapter 8.6.

8.2 Create and edit attributes

When double click on a UML-class in the diagram the *"class dialog"* shows up. In tab *"Details"* you can click the button *"Attributes..."* (see Image 39: Tab *"Details"* in the class dialogue).

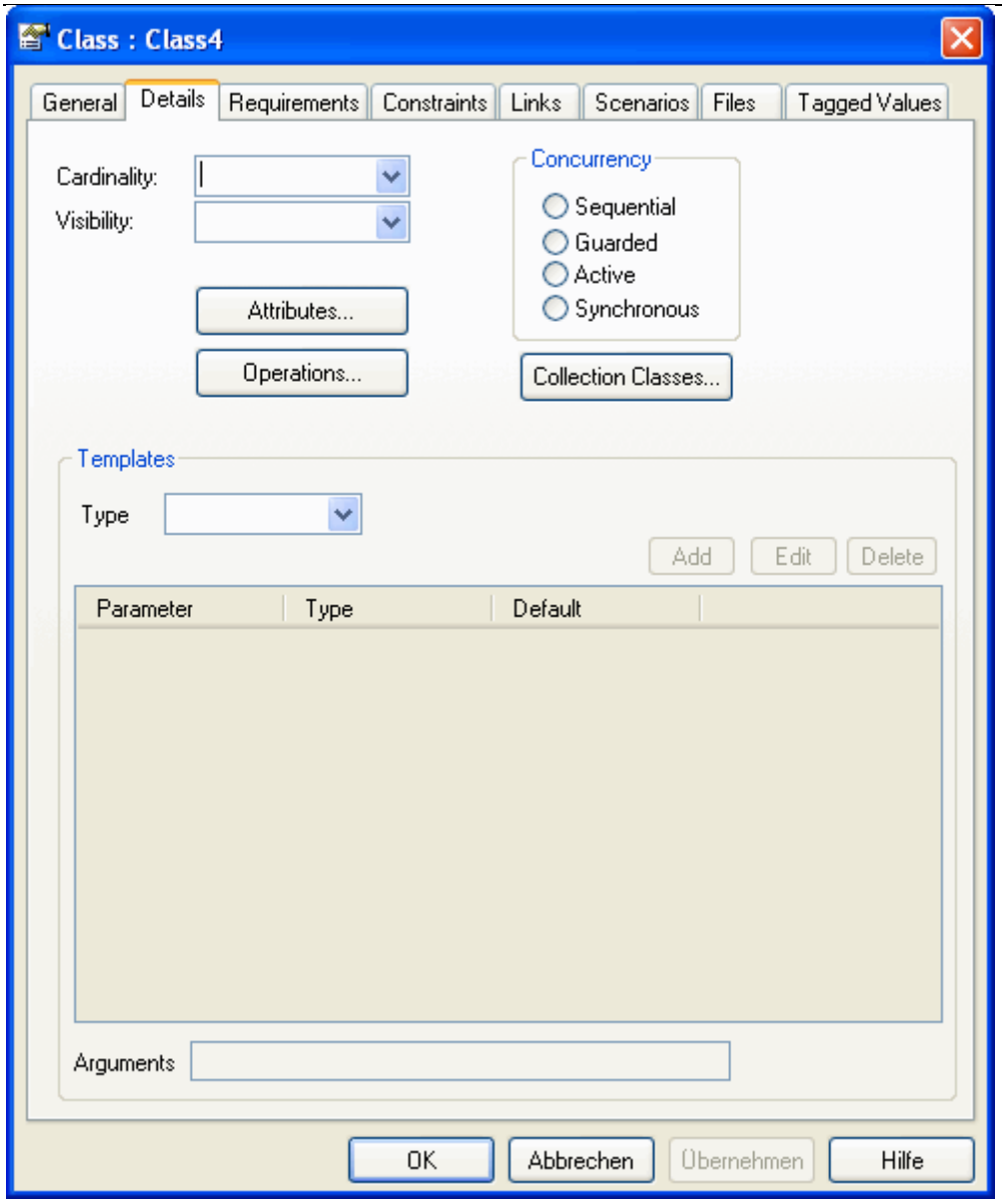


Image 39: Tab *"Details"* in the class dialogue

Which brings you to another dialog, where you can create and edit attributes. The fields *"Name"*, *"Type"* and *"Scope"* need to be filled in, however stereotypes can just be applied as soon as the profile is mounted. When the type of the attribute is not available in the drop down menu, it has to be entered manually. When you've filled in all fields it's important to click the button *"Save"*. Then the attribute appears in the lower section *"Attributes"* (see Image 40: New attribute in a UML-class).

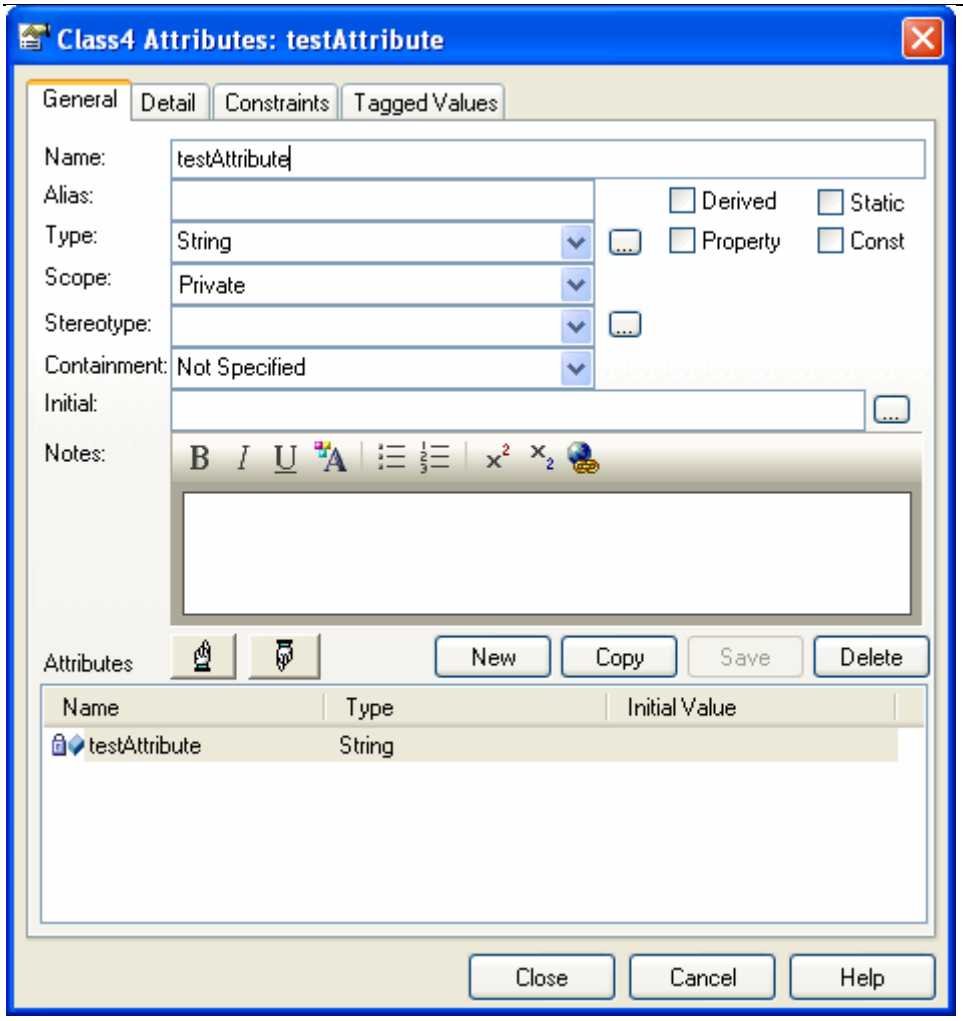


Image 40: New attribute in a UML-class

To edit an attribute, you have to mark it in the list. When finished you have to click *"Save"* again, to apply the changes. To delete an attribute simply mark it in the list and click *"Delete"*.

8.3 Create and edit methods

Methods have their own dialog Like attributes. To get there double click on a UML-class and then switch to the tab *"Details"* and there you'll find a button called *"Operations…"*. This dialog works the same like the attribute dialog. The fields *"Name"*, *"Return Type"* and if transfer parameters exists *"Parameters"* have to be filled in.

8.4 Create connectors

The easiest way to create a connector between two classes is to simply mark a class by clicking on it. Then at the top right corner an arrow appears. Now click on the arrow and drag it to another UML-class. Then a dialog appears to select the type of relation.

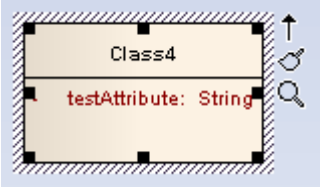


Image 41: Marked UML-class

Important types are:

- Association
- Aggregation
- Composition
- Generalization

As soon as the type is selected it's necessary to choose the cardinality (in *Enterprise Architect* called multiplicity). To do this double click ob the connection and a dialog will show up. The two tabs *"Source Role"* and *"Target Role"* are interesting for the configuration. Like shown in Image 45, the *"Source Role"* is *"TestClass1"* and the *"Target Role"* is *"TestClass2"*. The multiplicity has to be set in both tabs by using the drop down menu. The following multiplicities are possible:

- **Source Role Multiplicity:** 1, **Target Role Multiplicity:** 1 -> OneToOne
- **Source Role Multiplicity:** 1, **Target Role Multiplicity:** * -> OneToMany
- **Source Role Multiplicity:** *, **Target Role Multiplicity:** 1 -> ManyToOne
- **Source Role Multiplicity:** *, **Target Role Multiplicity:** * -> ManyToMany

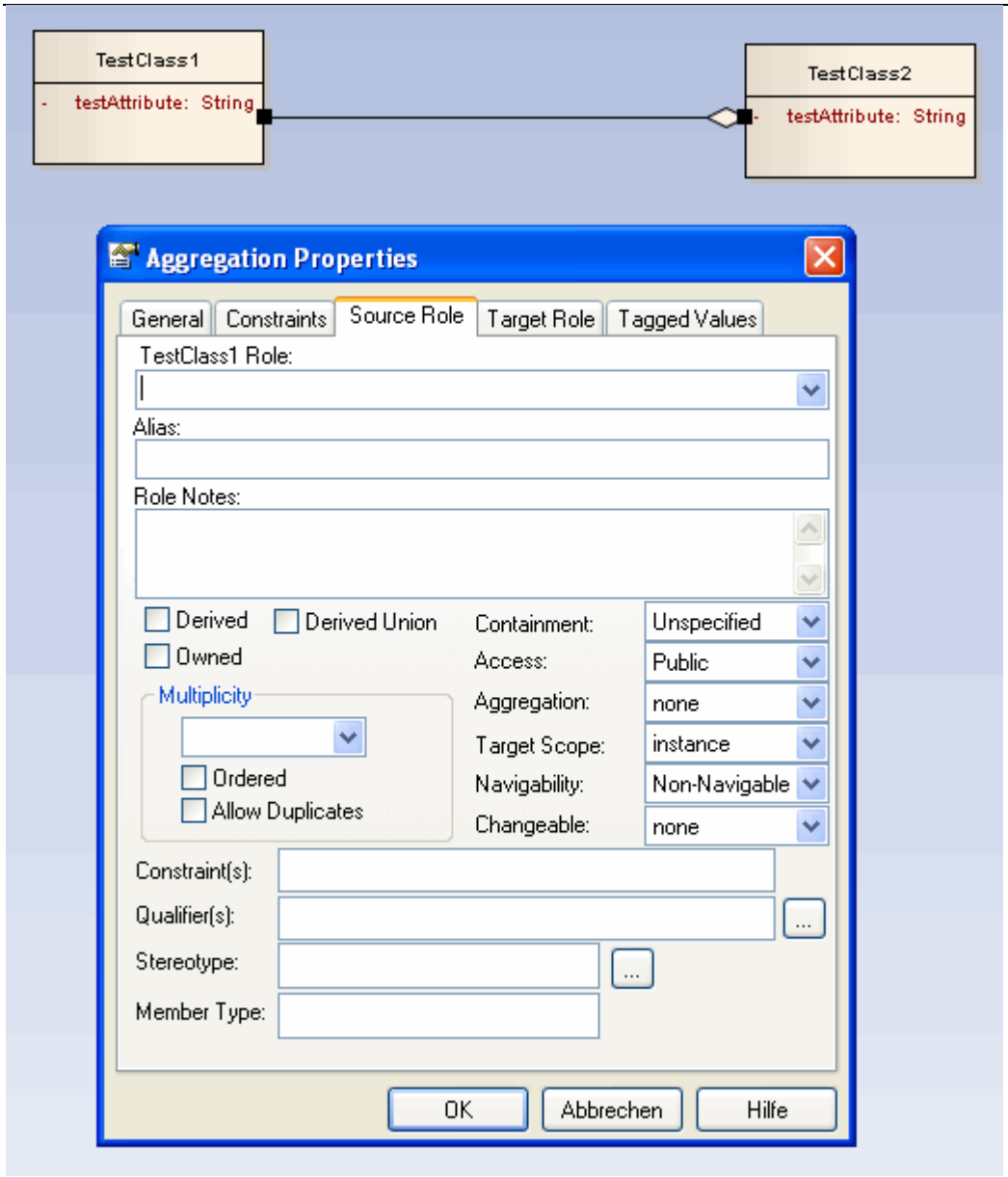


Image 42: Cardinality of a UML-relation

8.5 Mount the UML-Profile

If the "*Resources View*" in *Enterprise Architect* haven't been already opened it can be done by opening *"View – Other Project Tools – Resources"*. To select a UML-profile click right on *"UML Profiles – Import Profile"* and then browse to the profile. UML-profiles always have the XML-format. When the profile is selected it can be mounted by confirming by clicking *"Import"*.

8.6 Apply and delete a Stereotype to an UML-Element

When a stereotype of a certain UML-profile like for example of the JPA- or BVA-profile should be applied to a UML-element, the UML-profile needs to be mounted into *Enterprise Architect* as shown in chapter 8.5. Also the "*Resource View*" in *Enterprise Architect* is needed. It can be opened through *"View- Other Project Tool – Resource"*.

All mounted profiles can be seen under *"UML-Profiles"*. The stereotypes included in the profile are displayed when the profile is opened in the view *"Resources"* (see Image 43: UML-profile in *"Resources View"*)

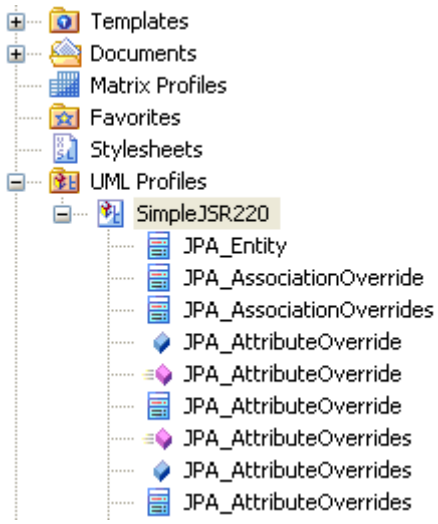


Image 43: UML-profile in *"Resources View"*

Stereotypes are tagged with three different symbols:

- This means the stereotype can be applied to a class
- This means the stereotype can be applied to an attribute
- This means the stereotype can be applied to a method

To apply a stereotype to a UML-element (class, attribute, method) first the class has to be selected in the diagram, by a single click and then the wanted element, by another click. The marked UML-element can be detected cause of the white background (see Image 44: Example for a marked UML-element (here an attribute)).

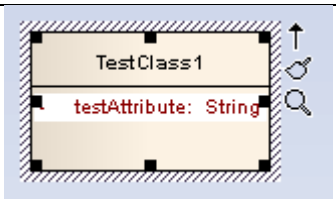


Image 44: Example for a marked UML-element (here an attribute)

As soon as the UML-element is marked, the stereotype can be selected in the view *"Resources"* and dragged to the UML-element. Then the stereotype will appear above the element. This process is identical to all UML-elements.

To delete a stereotype you can use, depending on the UML-element, the *"class dialog"* (see chapter 8.1), the *"attribute dialog"* (see chapter 8.2) or the *"method dialog"* (see chapter 8.3).

Caution: When a stereotype contains tagged values *Enterprise Architect* won't delete them if you delete the stereotype. There it's important to delete the tagged values in the tagged values view by using the button *"delete"*. It's always possible to check what tagged values are part of a stereotype, by looking into the profile. Tagged values are modeled as attributes. Image 48 shows the stereotype "*OneToMany"* with its tagged values *"cascade"*, *"fetch"*, *"mappedBy"* and *"targetEntity"*.

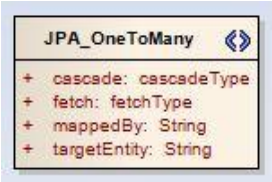


Image 45: Stereotype *"OneToMany"* with tagged values

8.7 Edit tagged values

Tagged values can be edited as soon as a stereotype has been applied to a UML-element (see chapter 8.6) and the stereotype contains tagged values.

Tagged values are displayed when the view *"Tagged Value"* is opened. You can open it by *"View – Tagged Values"* (or Strg + Shift + 6). To see the tagged values of an element firstly select the class in the diagram by simply clicking and then select the UML-element. The selected element is shown with a white background (see Image 44: Example for a marked UML-element (here an attribute)) If you switch to the view *"Tagged Value"* the tagged values are displayed (see Image 46: Example of *"Tagged Value View"* for the stereotype

"OneToMany"). Of course only if the stereotype contains at least one tagged value.

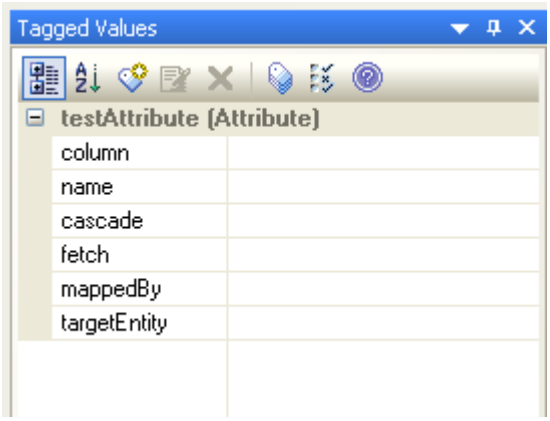


Image 46: Example of *"Tagged Value View"* for the stereotype *"OneToMany"*

Now the content of the tagged value can be changed in the right field. If it's a tagged value of type *"ENUM"* a drop down menu appears (see Image 47: Tagged values of type ENUM).

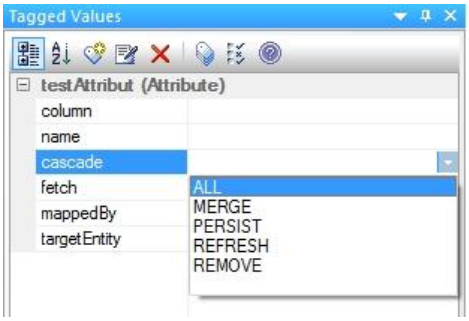


Image 47: Tagged values of type ENUM

When multiple stereotypes are applied to the same UML-element, and everyone contains tagged values, it can become very confusing to figure out which tagged value belongs to which stereotype. This can be cleared with the option *"Show Fully Qualified Tags"* (see Image 48: Tagged Values with option *"Show Fully Qualified Tags"*). Caution: If this option is activated, it's not allowed to edit tagged values. To do that the option has to be disabled again (see chapter 8.8.1).

8.8 Common Bugs

Unfortunately there are some known bugs in context of *Enterprise Architect* and the export process of the project into XMI-format. By now for these bugs no workaround was found. These bugs are listed here, because they are crucial to the modeling process. To hopefully get bug fixes in later releases the bugs have been reported to the developers.

8.8.1 Tagged values with the option *"Show Fully Qualified Tags"*

Description:

As soon as the option "*Show Fully Qualified Tags"* in the view *"Tagged Value"* is selected and a tagged value is edited, the tagged values won't be considered in the generation process, what means they will be missing in the javacode. However it's possible to use the option *"Show Fully Qualified Tags"* to get a quick survey, it's just no allowed to edit.

Basically this option is pretty helpful. Because if multiple stereotypes are applied to a UML-model the view *"Tagged Value""* displays all tagged values in a list sorted by the applied stereotypes (see Image 48: Tagged Values with option *"Show Fully Qualified Tags"*)

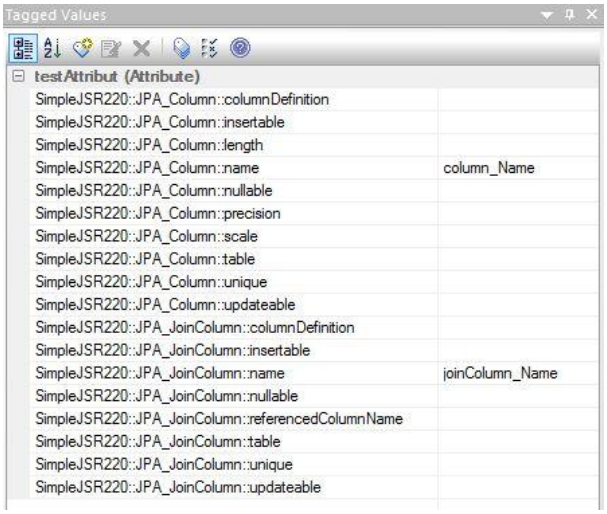


Image 48: Tagged Values with option *"Show Fully Qualified Tags"*

Solution:

When the option *"Show Fully Qualified Tag"* has been enabled, the following steps have to be executed to the UML-element to which a stereotype with tagged values was applied:

1. Select UML-element (White background)
2. Delete all tagged values in the view *"Tagged Value""*
3. Delete the stereotype that has been applied

Then it's required to disable the option *"Show Fully Qualified Tags"* in the view *"Tagged Values"*.

8.8.2 Multiple identical tagged values applied to one UML-element

Description:

When multiple stereotypes get applied to a UML-element, that have one identical tagged value description, the last tagged value will be applied to all tagged values with the identical description.

Example:

In class *"TestClass1"* the stereotypes "*JPA_Column"* and *"JPA_Join_Column"* are applied to the attribute *"testAttribute"* (see Image 49: Example for multiple stereotypes applied to on attribute)

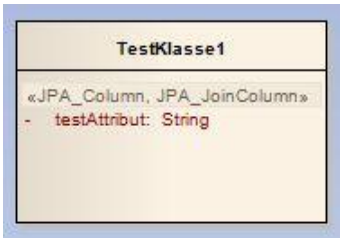


Image 49: Example for multiple stereotypes applied to on attribute

Both stereotypes have a tagged value *"name"* (see Image 50: Example for multiple identical tagged values applied to one UML-element). In such case the generated code contains for both tagged values the value *"joinColumn_Name"* instead of the expected tagged value *"JPA_Column::name"* with *"column_Name"* and *"JPA_JoinColumn::name"* with *"joinColumn_Name"*.

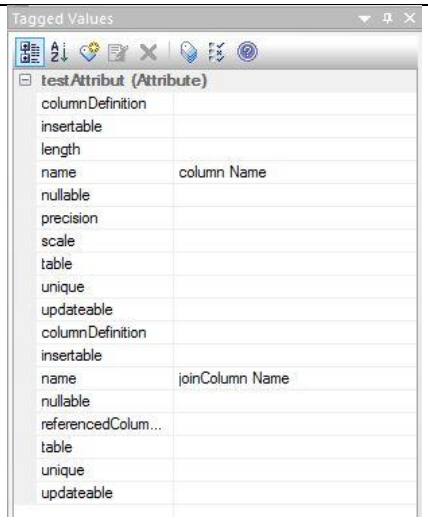


Image 50: Example for multiple identical tagged values applied to one UML-element

Solution

By now there is no solution.

8.8.3 When deleting a stereotype the associated tagged values won't be deleted

Description:

When the stereotype, that contains tagged values and is applied to a UML-element, will be deleted, the tagged values aren't deleted. They are still visible in the view *"Tagged Value"*.

Solution:

Every tagged value that belonged to the deleted stereotype need to be deleted in the view *"Tagged Value"*.

9 Important notes if you need to configure a new project

9.1.1 Configuring java compiler: compliance levels

To set the java compiler compliance level to 1.5 go to your new project right click on "*Properties*" In this dialogue go to "*Java → Compiler → Compiler compliance level"* and select the version 1.5 (see **Fehler! Verweisquelle konnte nicht gefunden werden.**).

This step sets the whole workspace to java version 1.5. The java version can be also changed for every single cartridge separately. The first way is recommended because a separate workspace is used in this tutorial.

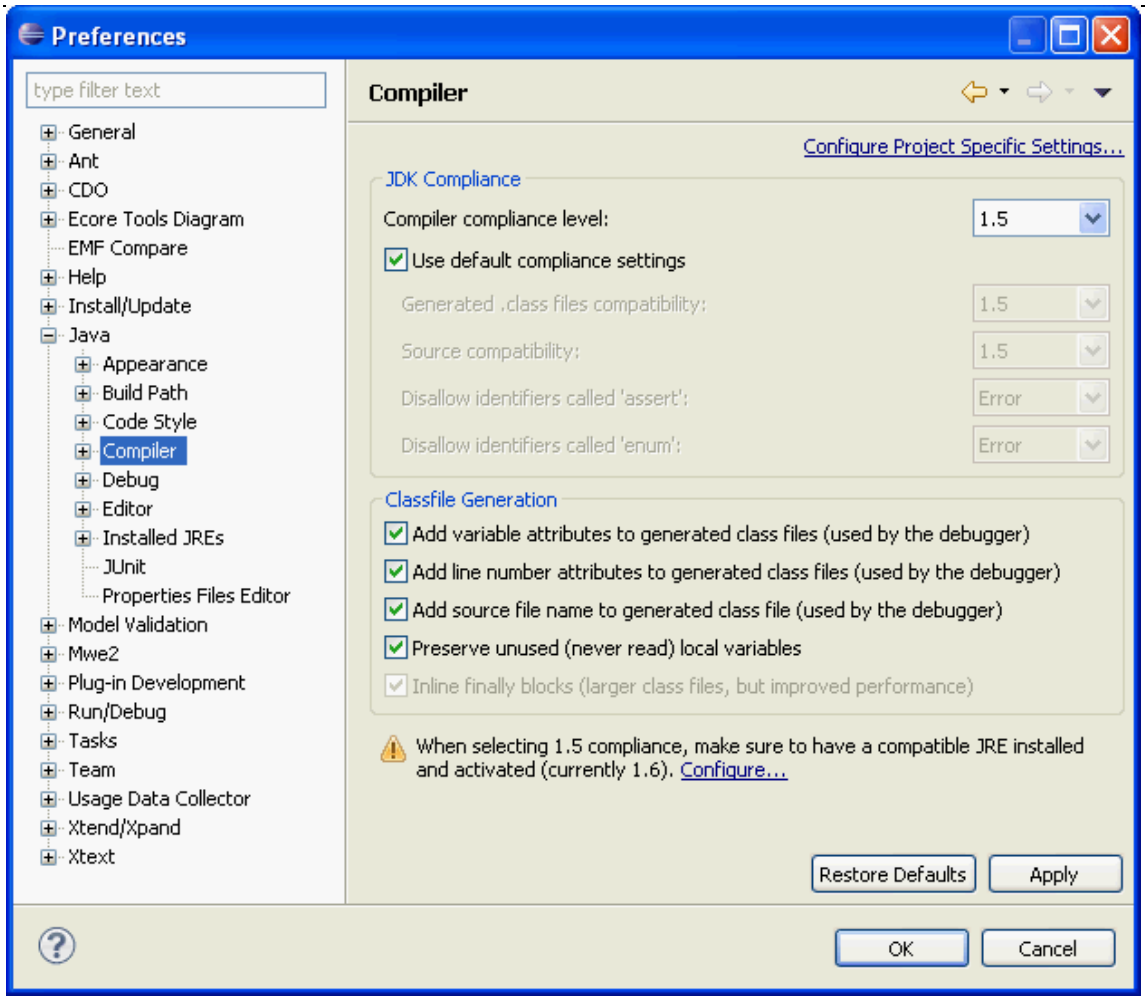


Image 51: Screenshot of preferences dialogue for java compiler

9.1.2 Configuring Xtend/Xpand

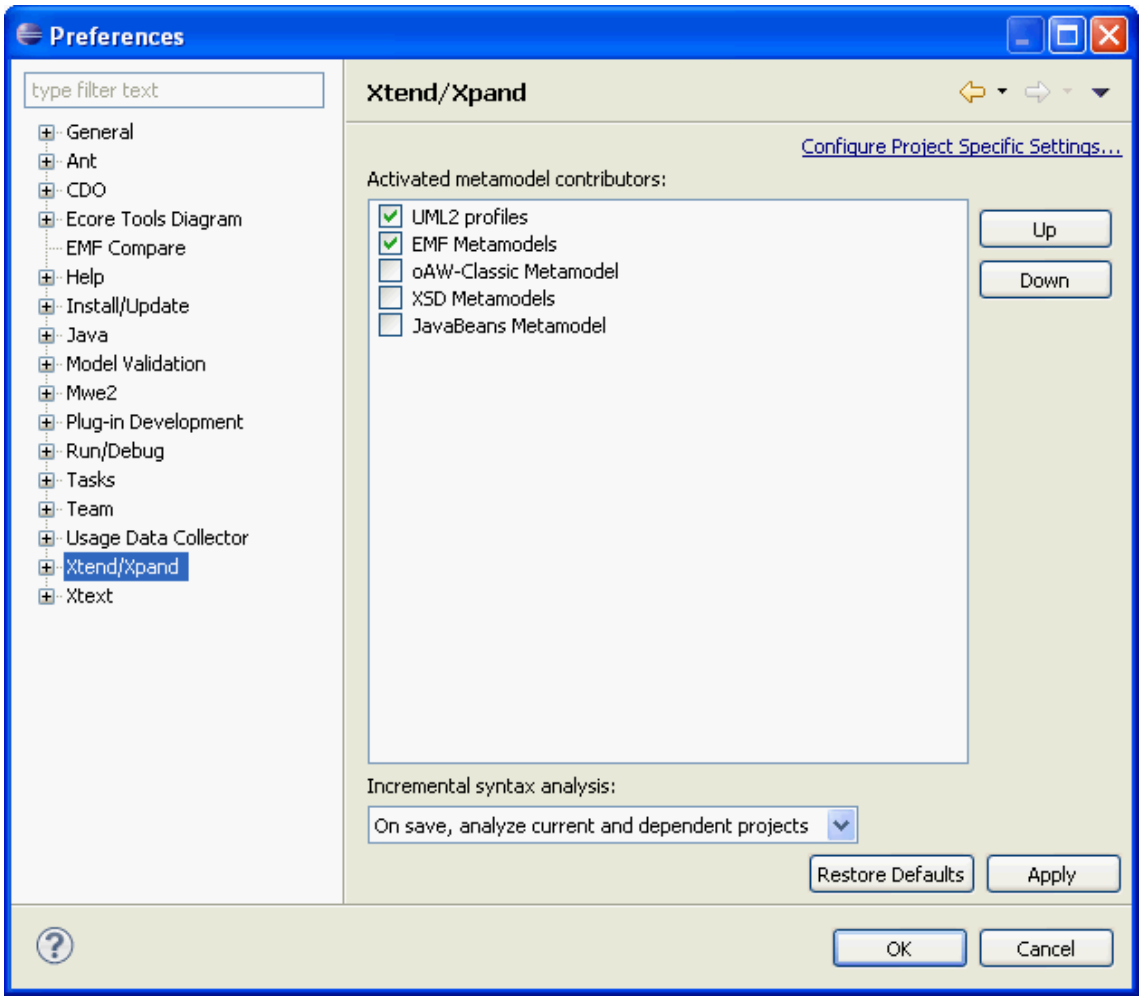


Image 52: Screenshot of preferences dialogue for configuration of the activated metamodel contributors

Go to *"Window → Preferences → Xtend/Xpand"*. Activate the entry *"UML2 profiles"* and move it to the topmost position by clicking the button *"Up"* (see **Fehler! Verweisquelle konnte nicht gefunden werden.** and **Fehler! Verweisquelle konnte nicht gefunden werden.**). After this click the button *"Apply"*.

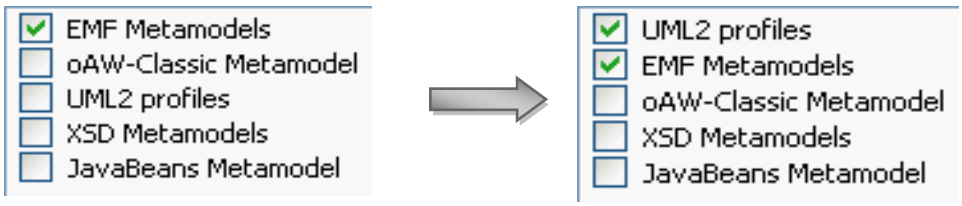


Image 53: Screenshot of the dialogue for activatiion of different UML2 profiles

10 List of Images

11 List of Tables

12 List of References

Brawand, U. (2009). *UML2Exporter*. Von http://uml2ea.blogspot.com/ abgerufen

Eclipse Foundation and Others. (2009). *Eclipse Modeling Framework Project*. Von http://www.eclipse.org/modeling/emf/ abgerufen

itemis AG. (2009). *JavaBasic-Cartridge*. Von http://fornax.itemis.de/confluence/display/fornax/JavaBasic+(CJB) abgerufen

Java Community Process (verschiedene Firmen beteiligt). (16. März 2009). *JSR 303: Bean Validation 1.0 RC1 Proposed Final Draft*. Von http://www.jcp.org/en/jsr/summary?id=303 abgerufen

No Magic Inc. (2009). *Magi Draw*. Von http://www.magicdraw.com/ abgerufen

Object Management Group. (2009). *OMG Model Driven Architecture*. Von http://www.omg.org/mda/ abgerufen

Object Management Group. (2009). *OMG Modeling and Metadata Specification*. Von http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML abgerufen

openArchitectureWare. (2009). *openArchitectureWare Projekt*. Von http://www.openarchitectureware.org abgerufen

Wikipedia Foundation and Others. (2009). *Wikipedia (English)*. Von http://en.wikipedia.org/ abgerufen

Wikipedia Stiftung und Andere. (2009). *Wikipedia (deutsch)*. Von http://de.wikipedia.org/ abgerufen

13 Index