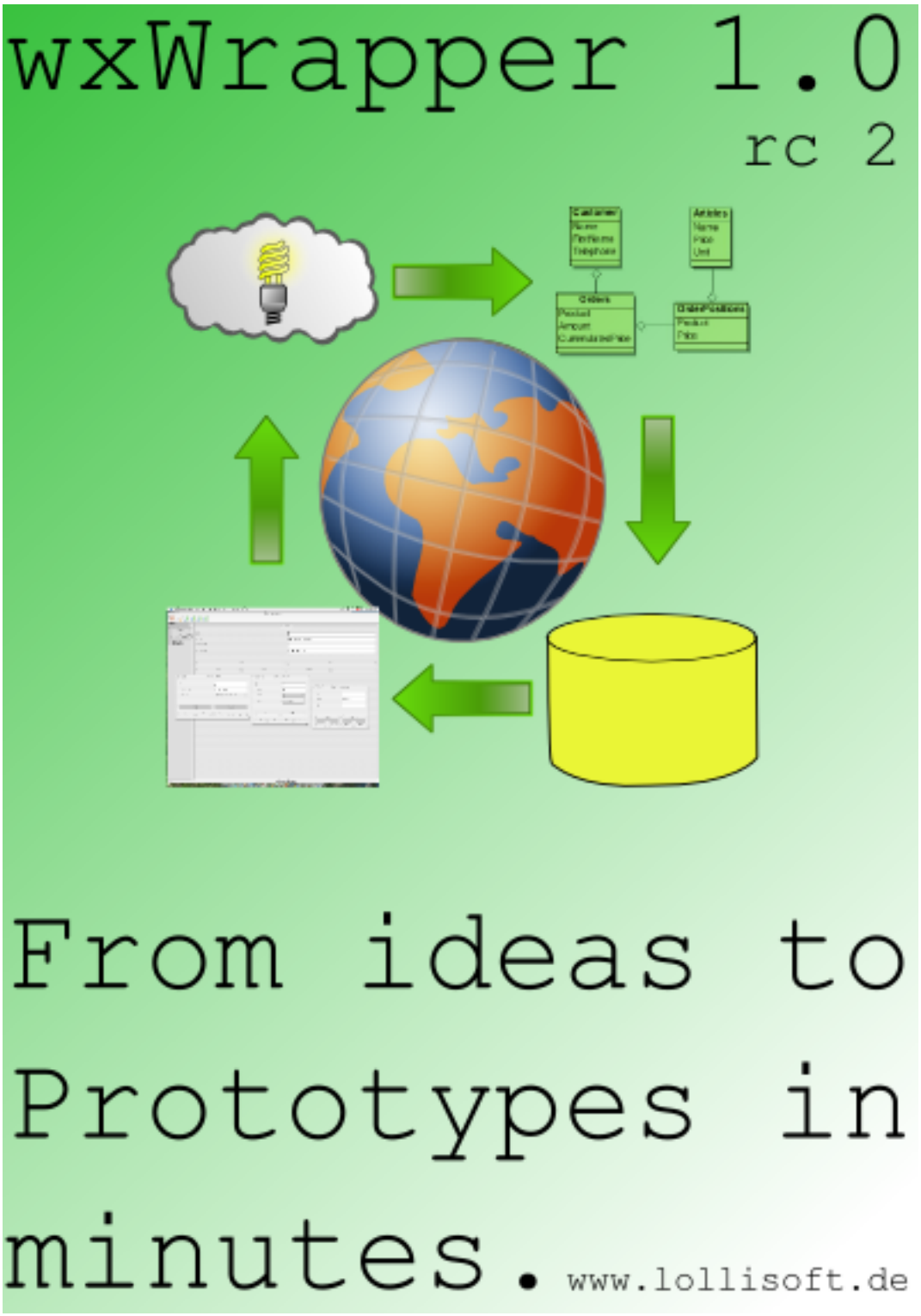


Fast Application Prototyping

driven by wxWrapper and its lbDMF framework in version 1.0rc2

Fast Application Prototyping

Documentation

Creating a database prototype in minutes

$Revision: 1.1 $

Table of contents

Introducion

Writing an application is not easy. Especially creating database applications, such as CRM systems or simple CD cataloging software.

wxWrapper enables you to load various application modules defined in a configuration database.

One predefined configuration is 'lbDMF Manager' that enables you to define other database applications. This definition is really based on a dynamic application module who uses further configuration data defined for a specific application. In this case 'lbDMF Manager'.

The DynamicApp module enables you to access to databases by a definition stored in a configuration database.

With this application module it would be a simple and less time consuming work until you have a first version of your database application. You don't need to write one line of code. All you have to do, is defining what data you would make available in the application.

A database application consists of various database forms, that will provide you with different views on to data. For all these views you will define one form definition per view.

Concept

Without any extra tool (UML would be such an extra tool) you may manually define your application. A simple CD collection database would be shown in the API documentation that is available online or in the Documentation package of the lbDMF project.

The API documentation can be found here.

Further in this document I like to introduce you in using an UML modelling tool to get the maximum speed in developing a database prototype. Any other modelling tool may be usable when XML export would be possible and a suitable import template exsists.

So the main concept is UML modelling, export, import and try out.

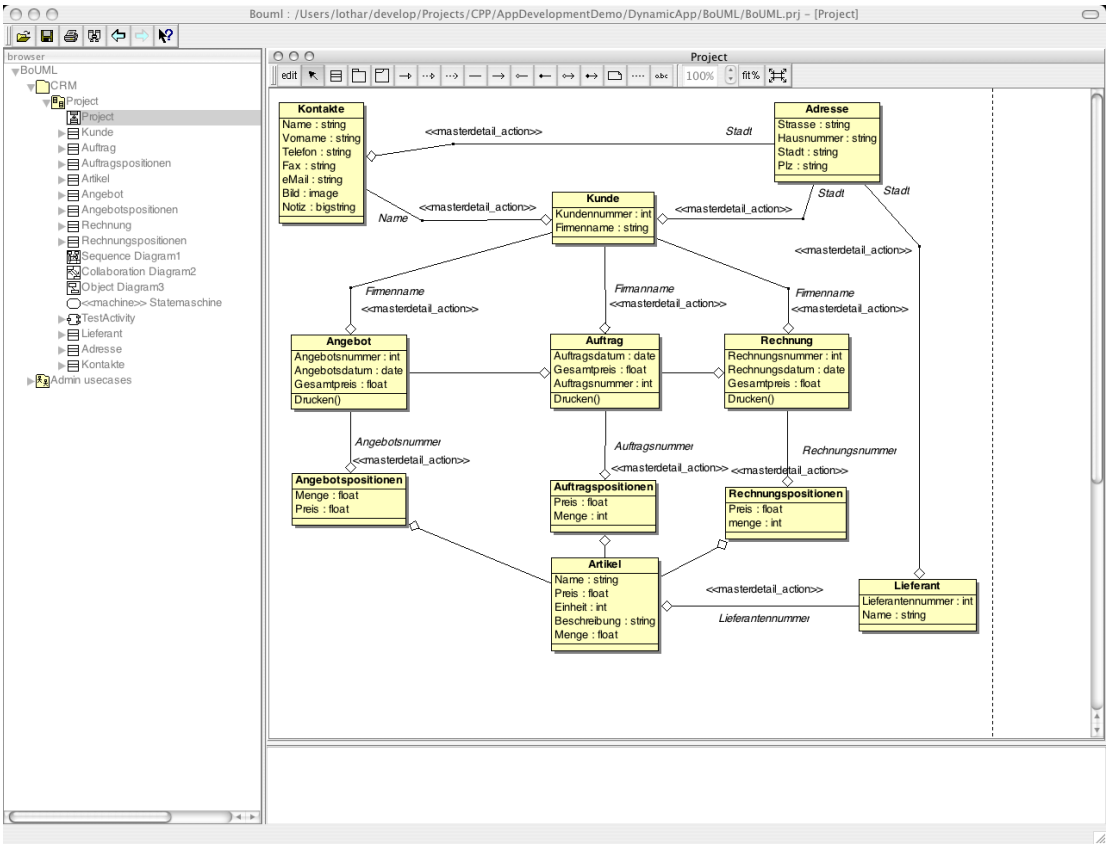
Quickstart

In the quickstart chapter I will give you a short overview on using the tools to show you what is possible and what is not yet possible.

Creating an UML model

To create an UML model, you will need to open BoUML. You then go to the Project menu and click on 'New'. You will be asked for a file name. Enter CRM for that. The first entry in the left tree would be the package name 'CRM' and this is important. It would be the application name in the Prototype.

The UML model from the project will look like picture 1.



It is a small CRM system you would be able to enter customers, articles, addresses, contacts, invoices and the like.

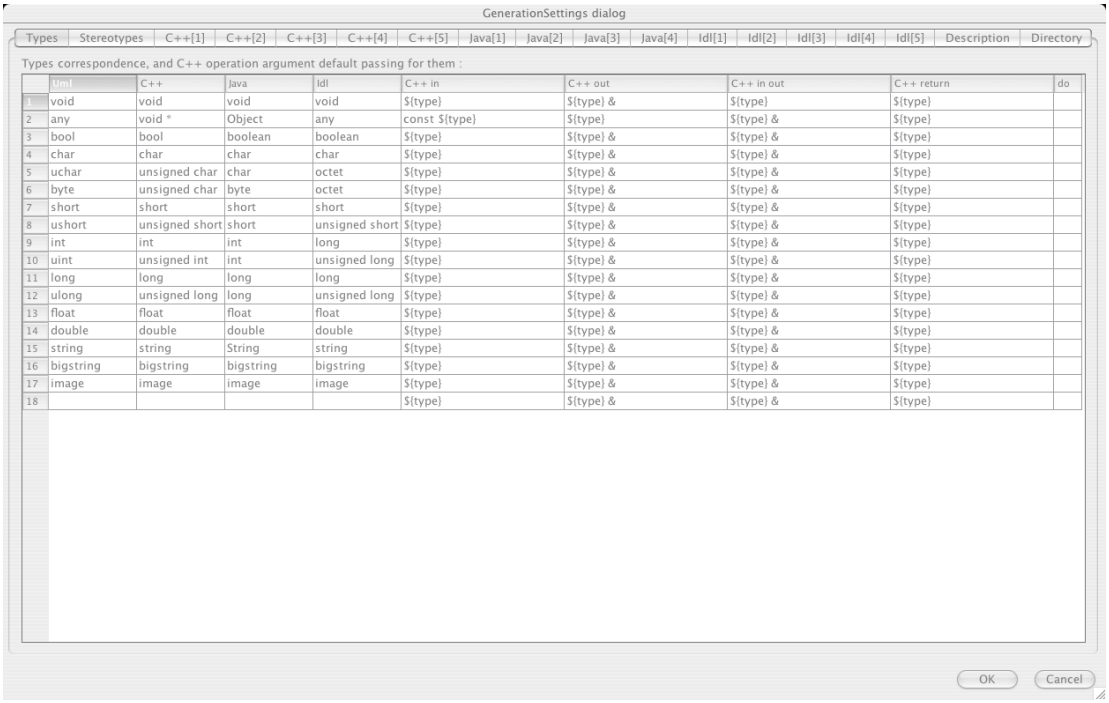
Prepare UML usage

Before you enter any UML classes, you need to setup some stereotypes and datatypes. These types are explained later.

New datatypes

Go to the toplevel entry in the browser pane on the left. Open the Project menu, in there open Edit->Edit generation settings.

You will see this screen:



The following types have been added in comparsion to a new UML project:

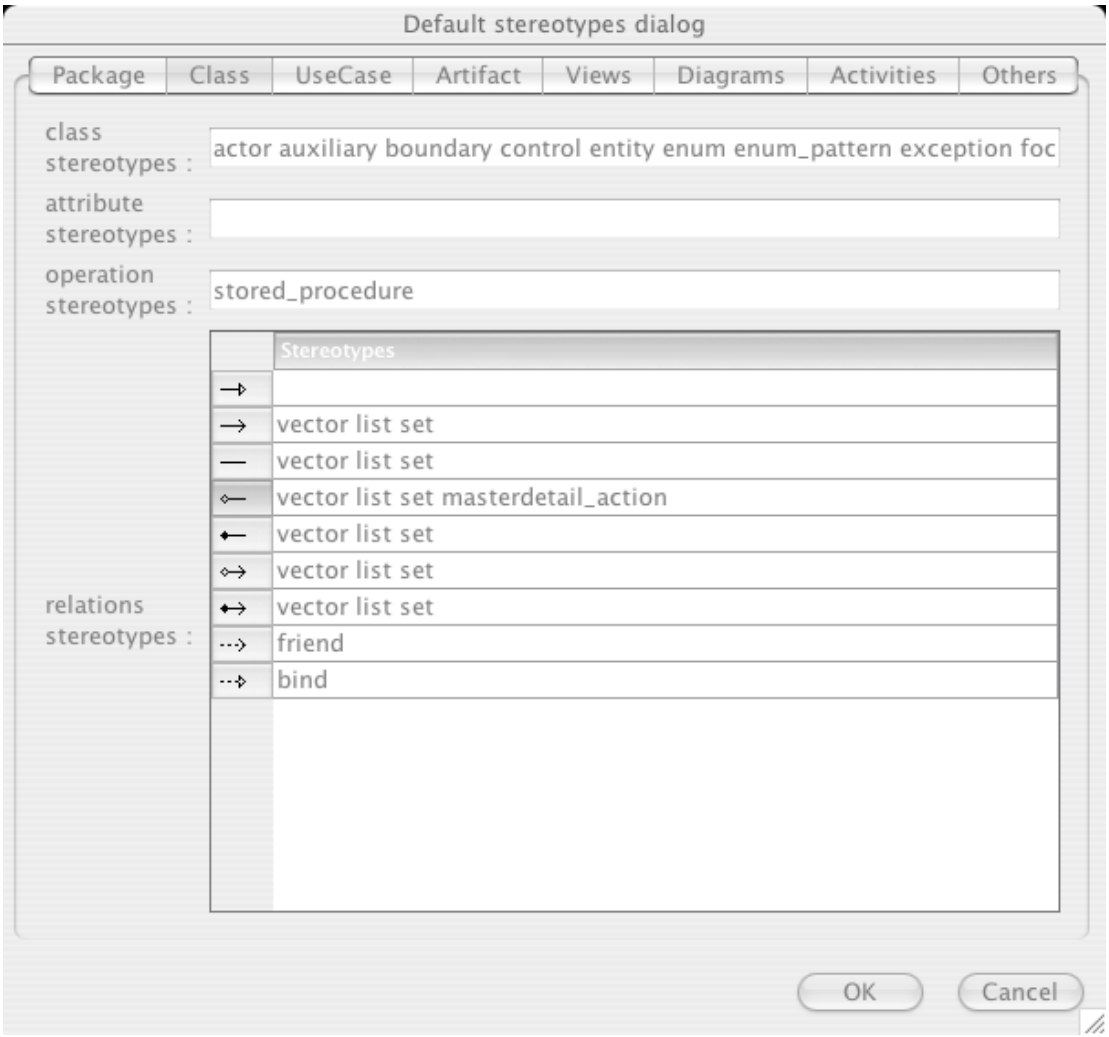
Type bigstring: The bigstring datatype is used to store large text into a database table. Use this for your memofields.

Type image: The image datatype is used to store various image formats in a database table.

New stereotypes

Go to the toplevel entry in the browser pane on the left. Open the Project menu, in there open Edit->Edit default stereotypes.

You will see this screen (Picture 3):



The following stereotypes have been added in comparsion to a new UML project:

Stereotype masterdetail_action:

> The masterdetail_action is used to indicate that an action button will be used in the master form (Adresse is a master) to open a detail form.

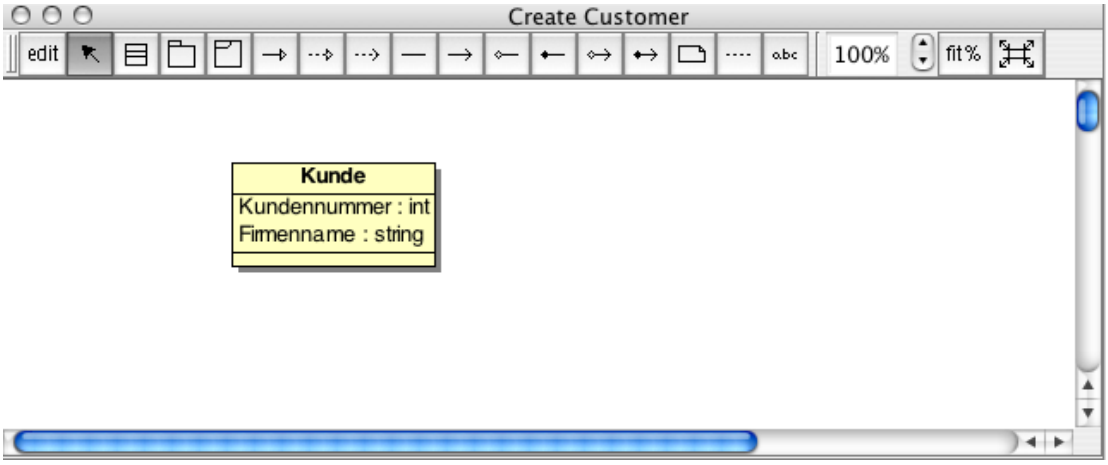
Creating the first UML classes

We will begin the UML model for our small CRM system with the customer class (Kunde). This may be the first class in mind for a CRM.

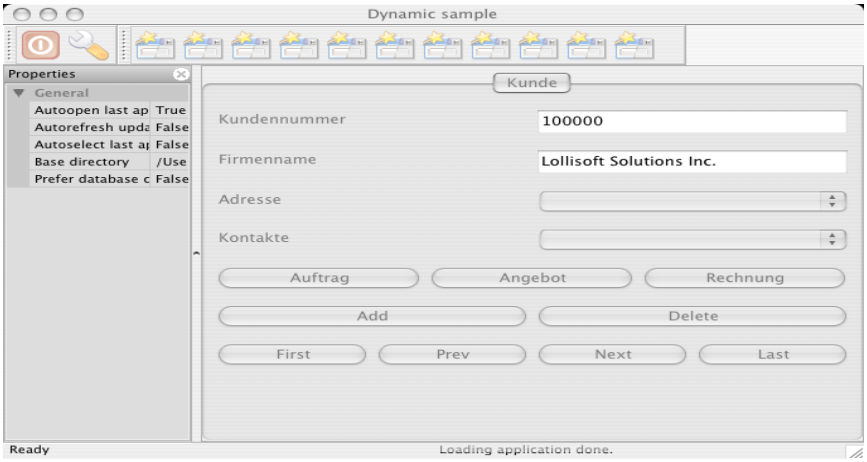
To show this, I will create separate class view, because I still have the model. You could use the main project's classview (Project). It takes no matter how the class view name looks like. I do not document the complete UML diagram because you will get it from the project.

Add the customer class to the model

The customer class looks like this (Picture 4):

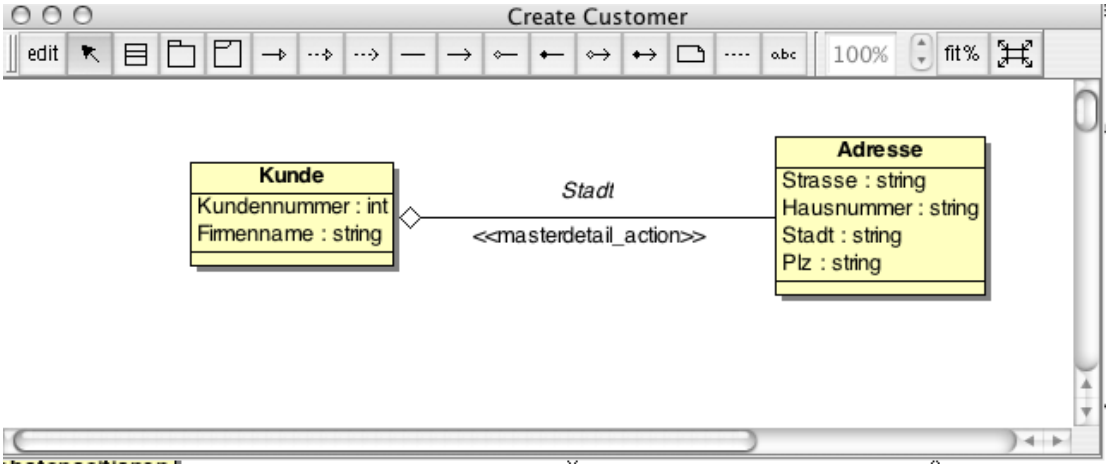


You see, that I do not add any address related data to this class. This is because an employee may also located at any address. Therefore we will add a separate class for the address (Adresse). The customer mask looks like here (picture 8, ignore the dropdown controls):



Add the address class, aggregate to customer

The address and an aggregate (Picture 5):



Here you see the new class Adresse. It is yet linked to the customer class with an aggregate. This implies that the customer will have exactly one address, but the address may have multiple customers.

Currently I could only use aggregates for an application prototype. This is because the way I import the UML model described later.

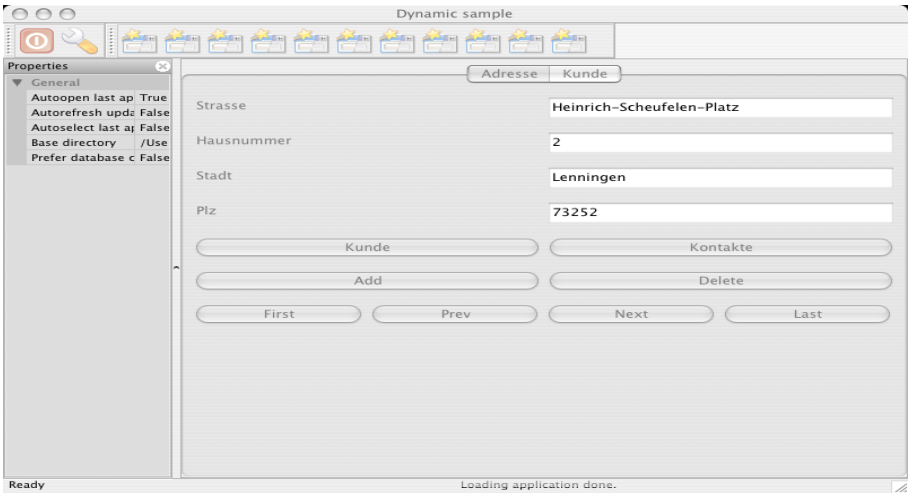
Here I will explain the details on aggregations. On the given sample you will see that the aggregate has an associated stereotype of masterdetail_action. This is important for the later navigation between the database forms. Here you will be able to open the customers located at a specific address.

The bad side on this model is, that you do not see the address of a customer at the same time because I do not support sub panels to show all the details of the selected address. Also there is not automatically an address tuple for this customer. It may be empty for now.

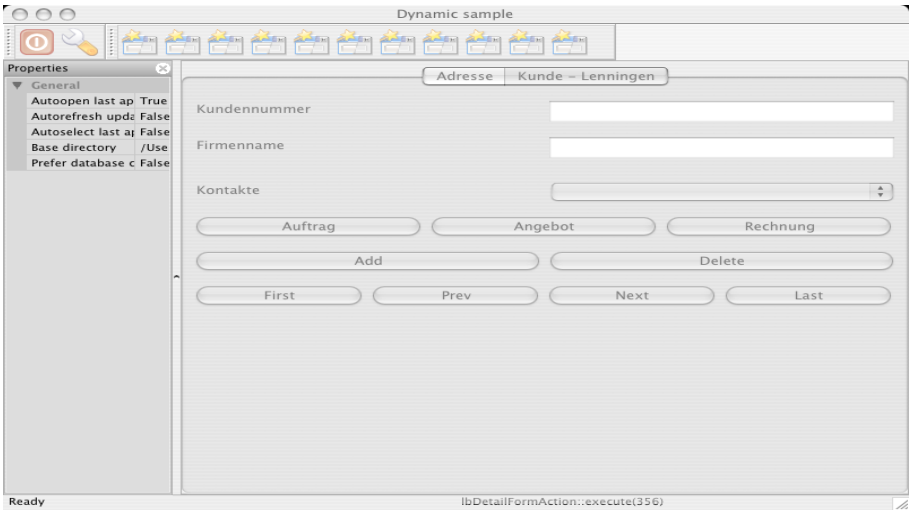
To enter a full customer entry with an address, you first need to enter the address and then save it by navigating once next and back. I do not have a save button yet. Then open the action 'Kunde' from the selected address.

You will get an empty form without any customers. No customers have their location at the given address. Simply press 'Add', then edit the customer data. You do not need to choose the address this way because you have opened the customer form from a given address. This automatically associates the address to the customer. This has advantages and disadvantages.

Here is a screenshot of the address form (Picture 6):



The opened customer form from address form (Picture 7):

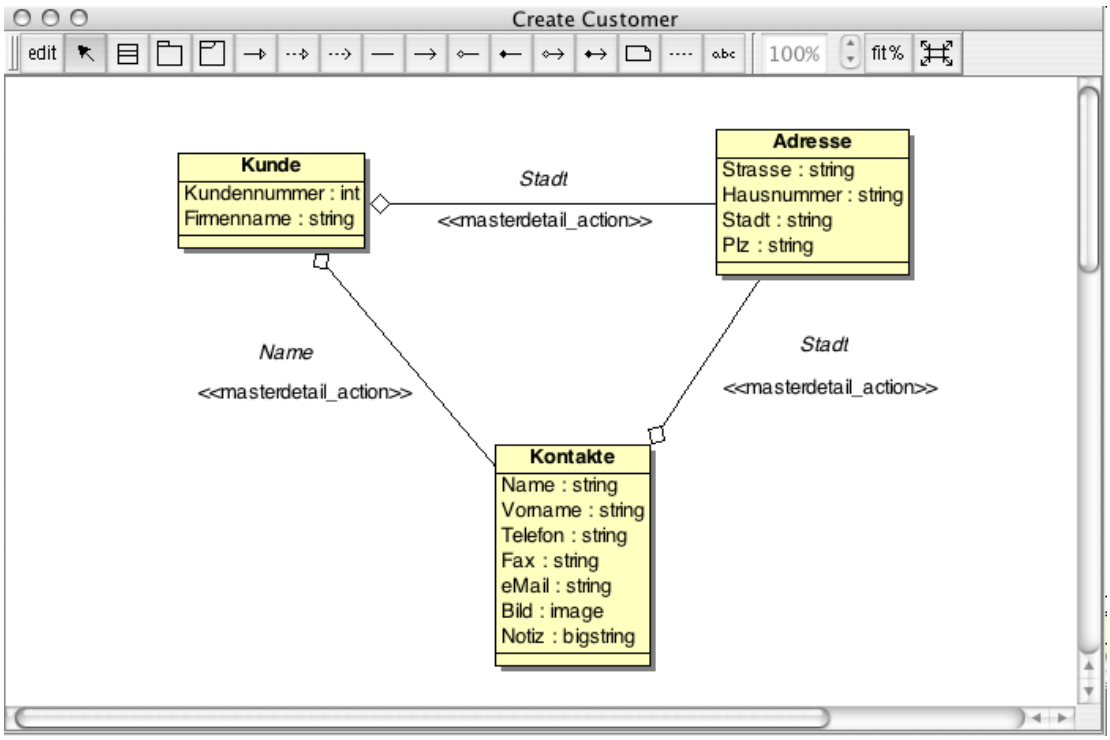


There is no data because you have newly created an address. You will see that there is no Address dropdown control. Compare to the picture in adding the customer class. By the way, you see the contacts control. This is there because I have added a contacts class and also assotiotiated it to the customer. The resulting UML model is shown on the next page.

Add the contacts class, aggregate to customer

A customer may have various employees. Each employee may be your contact person. Therefore this is a contact.

The UML model until now (Picture 9):



Here you will see, that a contact may have an address, but it could also assigned the same address as the customer entry has. Also you will see the stereotypes given to each aggregate.

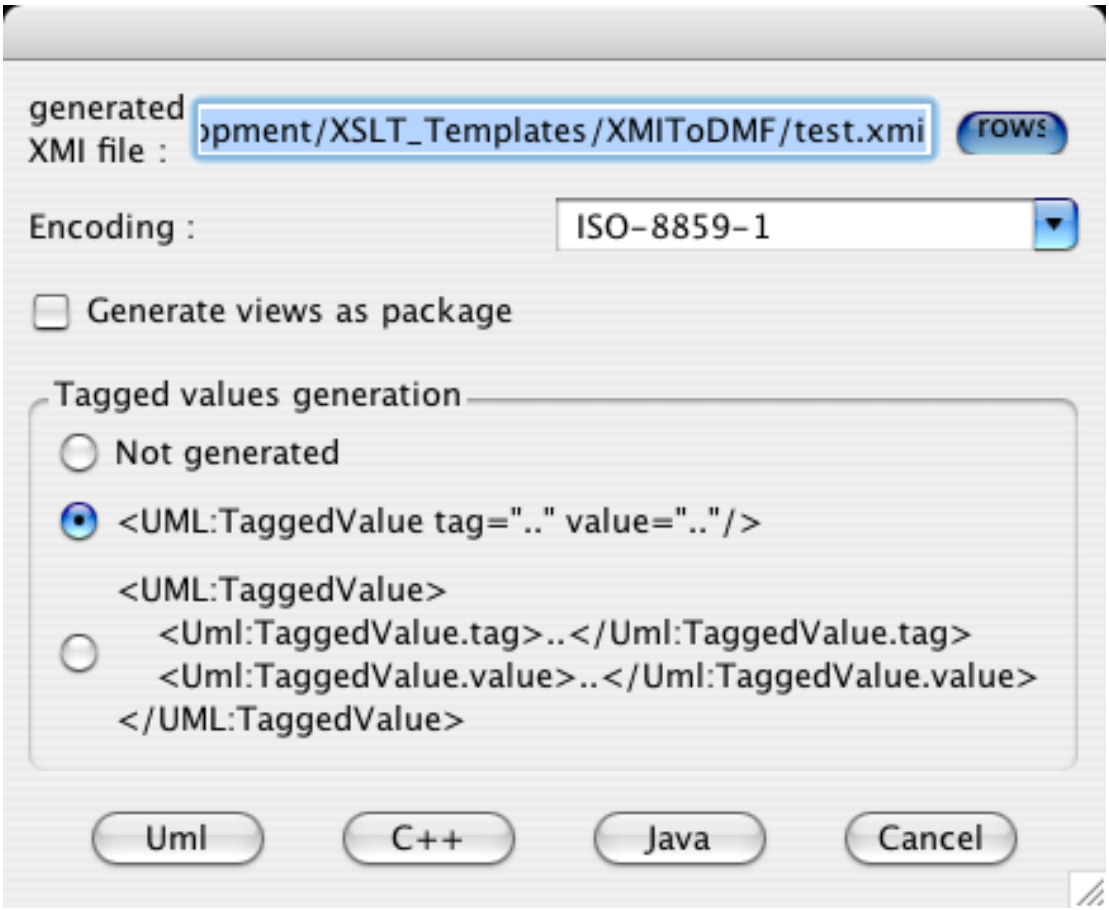
You may open contacts or customers from a given address. The aggregate has in both situations the master link on 'Adresse'. Now you will understand the picture Picture 6

What you also will see – after some customers have been entered – are the address and contact fields showing the town (Stadt) and surename (Name) field's data.

This is a direct relation to the text above the stereotype in Picture 9. This UML model is as complete as a database schema could be created and also an application model could be generated when exported. (XMI)

Exporting your UML model as XMI file

After you have created an UML model, you are ready to export it for further development. To do this, open the Tools menu in BoUML and click on 'Generate XMI'. It looks like this (Picture 10):



You need to specify the output file once and leave the rest as shown. After the settings are correct, click on 'Uml' to do the export.

Note:

On Mac OS X, I have to switch the window by pressing 'Meta'+'Tab' to see this window.

Importing the UML model via XMI file

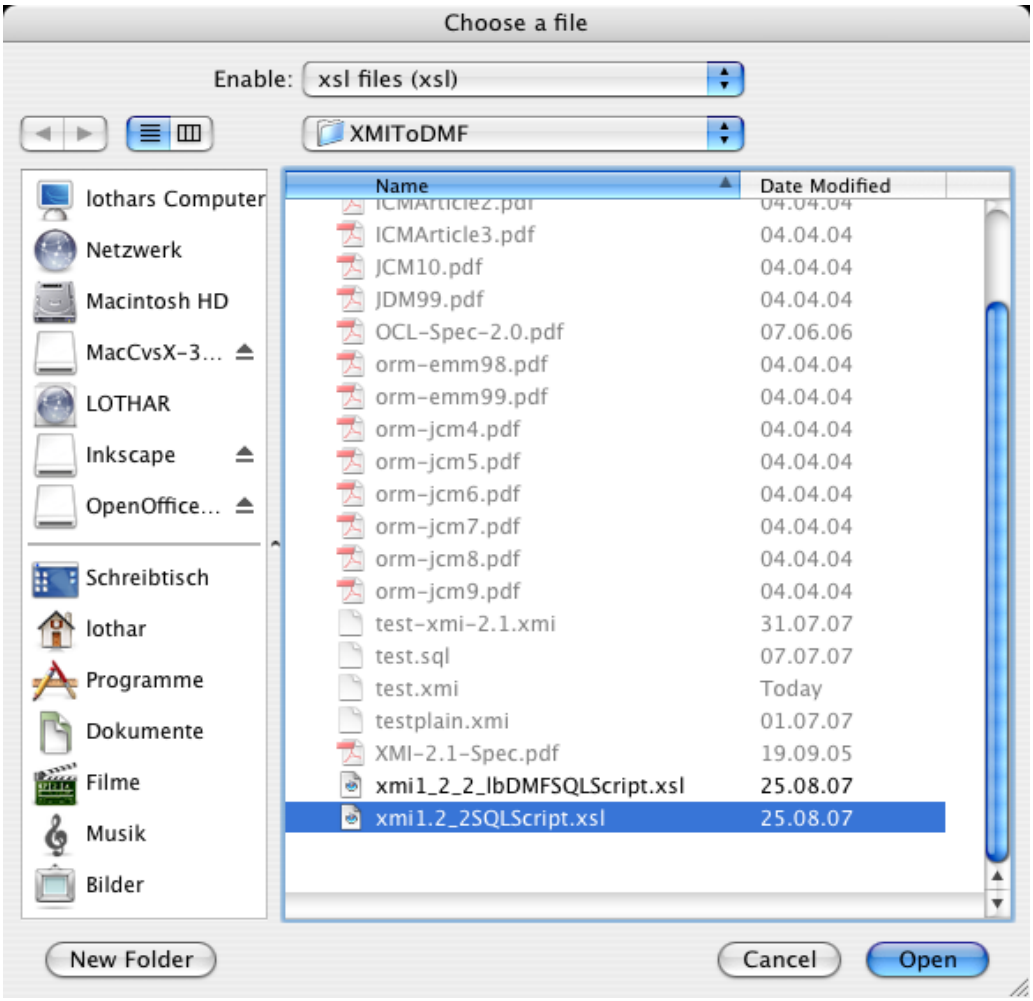
Before you import any XMI file, be sure you not yet have imported that file before. If so have a look at Deleting an application definition

To import an application definition, start wxWrapper, login with the default user 'user' using 'TestUser' as password. Then select the application 'lbDMF Manager' and proceed with importing by opening the File menu and clicking 'import application from UML (as XMI file)'

The first file to be opened then is the related XMI file you will import. Select your XMI file. Now you get asked to create the database for the application. Choose 'Yes' if you not yet have it imported.

Step 1. Creating the application database

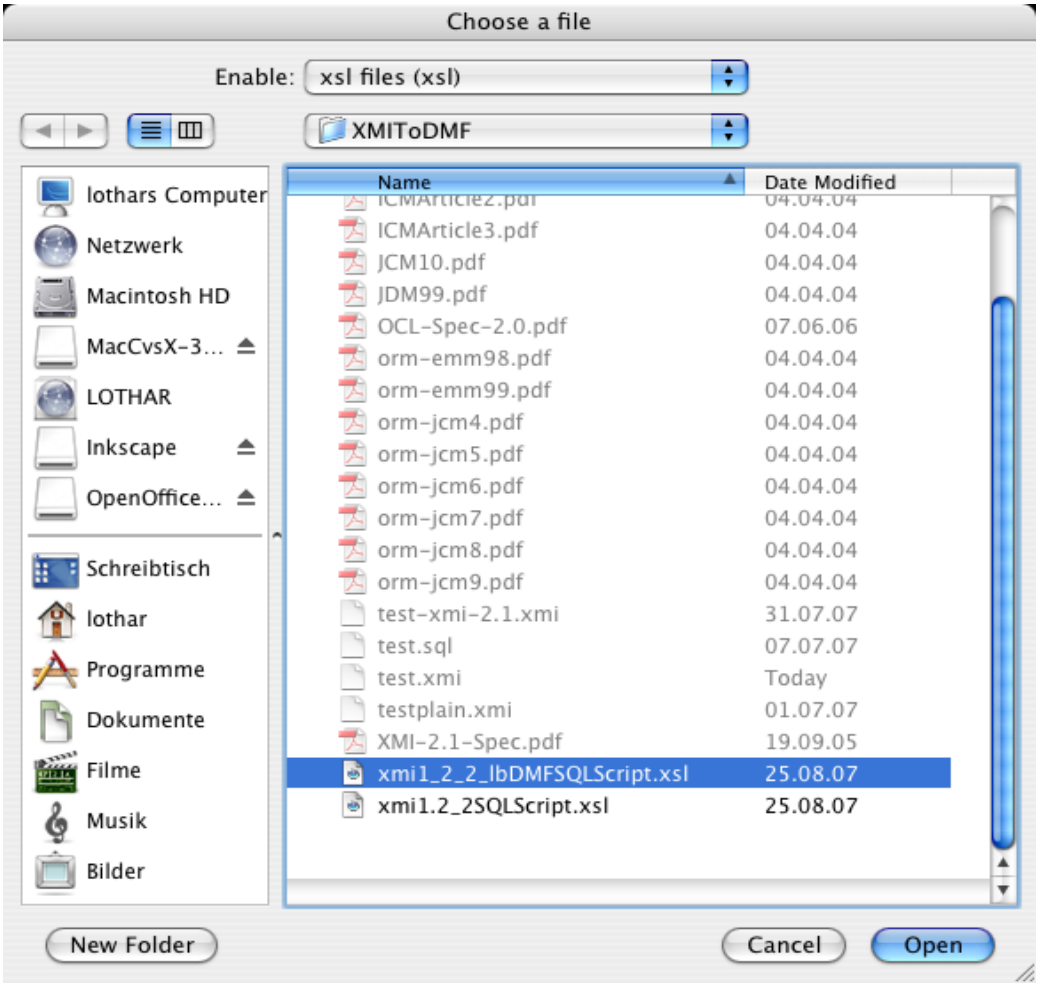
If 'Yes' was selected, you must choose my predefined template as of the picture 12:



If 'No' was pressed, proceed with Step 2. Creating the application model definition

Step 2. Creating the application model definition

You will be asked to import the application model definition. Do that by selecting the file shown in Picture 13:



Now you have a new application prototype. To test it, you need to uncheck the menu entry 'Autoload application' in the Edit menu. This is required to enable manual logon to a different application.

Deleting an application definition

If you have an application that was imported from an XMI file and you like to reimport it, you need to delete the affected application definition.

Deleting an application definition does not delete the database schema of that application. It only deletes it from the configuration database.

To delete an application definition, start wxWrapper, login with the default user 'user' using 'TestUser' as password. Then select the application 'lbDMF Manager' and proceed with opening the form 'Anwendungen' in menu 'lbDMF Manager' or clicking the yellow box (third from left). Select the intented application and press 'Delete'.

Picture 11 shows how it looks like: