

From model to module

The following shows with the example module „AutoCustomer“ for management of customers and their orders how to realize a small module with ModuleStudio. Thereby the most important terms and coherences are explained.

ModuleStudio stores each model in two files. The semantic information is being stored in the domain model *.msmodule, the optical diagram details in *.msmain.

After the application has been started, a wizard can be started with the menu entry „File > New > Module Diagram“, which creates both files according indication of their filenames. Then the user is located in the empty main editor (see Figure 1).

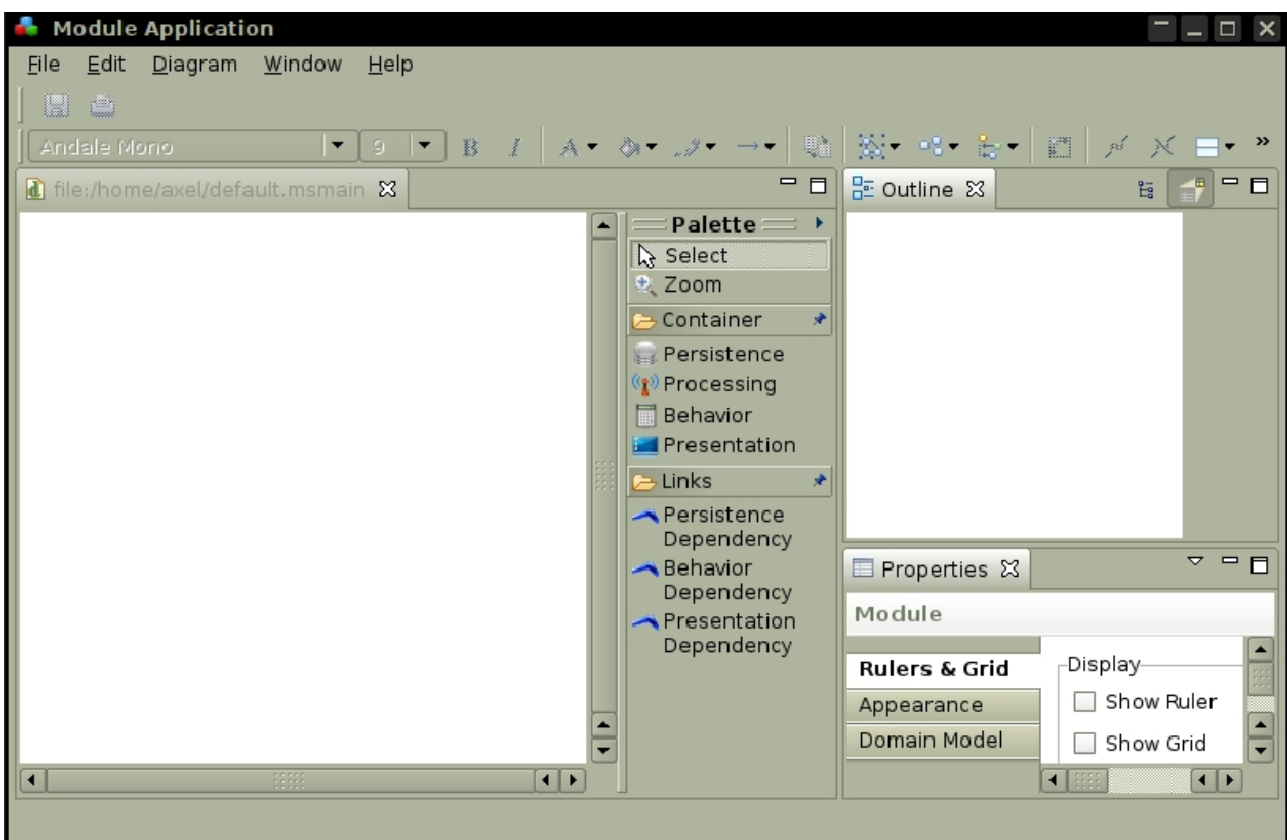


Figure 1: Structure of the application

Beside main menu and toolbars the environment is separated into two sections. The window arrangement can be changed according to your desires. At the left side the editor window resides containing a canvas and a palette with the available tools. There happens the real modeling.

The main editor's palette categorizes nodes and edges in two groups „Container“ and „Links“. The four container types represent the different technical areas of the model.

- Persistence Data layer for definition of tables and relations
- Processing Process layer for definition of use cases
- Presentation Presentation layer for customizing cosmetics
- Behaviour Behaviour layer for definition of business logic

Multiple elements of each container type can exist at the same time in a model. This can be helpful to divide up the single layers according functional criterias. But at the moment this is only relevant for the data layer to realize connections to external data sources.

On the right side there is an Outline View with a preview of the editor window and a Properties View for editing properties. In the latter the tab „Domain Model“ is important in which is a section named „View > Element“. There you can edit the attributes of model elements. Since the canvas complies with the module to be generated, at first some central settings should be set there. Da die Zeichenfläche selbst dem zu generierenden Modul entspricht, sollten dort zunächst einige zentrale Einstellungen gesetzt werden. This affects the module name as well as the author's name (see Figure 2).



Figure 2: Properties View

At this stage the model should be saved one time already. In the right upper edge of the editor window there is a red icon located (not shown in Figure 1) indicating that errors exist in the model. If the user hovers the mouse over this symbol according hints are displayed giving information about what is needed to do.

First of all a container element of each type is needed. These elements can be selected in the palette with a mouse click and then be dran onto the canvas. Alternatively popups can be used appearing in the canvas as soon as the mouse retires. Each container has to get named. Figure 3 shows how the model is looking now approximately.

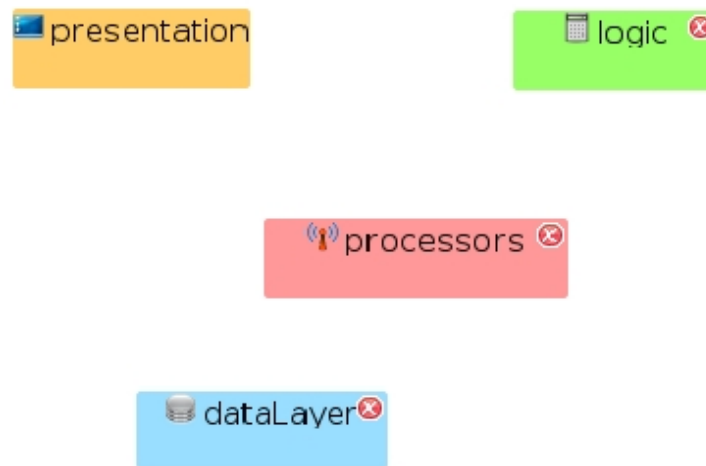


Figure 3: Container elements

Also at single elements now context-sensitive references to unfulfilled validation rules. The user thus anytime sees which parts he needs to touch up. For example the container element for business logic must be dependent from a data source, that is a persistence container.

The specification of dependencies between containers is being done with edge elements. The palette can be used also here for creation. A faster and more intuitive option is hovering an element with the mouse and using the appearing arrows as starting point for drag n drop. Figure 4 shows the model after adding the necessary dependencies.

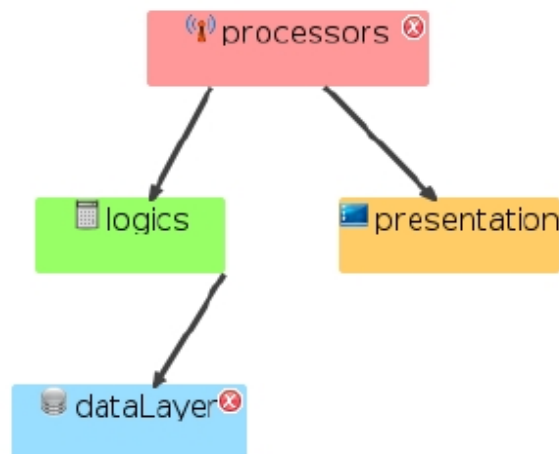


Figure 4: Containers with dependencies

The error message of the element „myData“ says, that the persistence container has to contain at least one table. To define the data layer you can open the according sub editor by double-clicking the element.

The persistence editor's palette contains many different entries. The main element is the table which can take up different fields and indexes. As soon as a table exists on the canvas a table-specific popup appears to add new sub elements easily.

Both tables and fields have multiple attributes which are managed in the Properties View at the moment. For future individual dialogs for changing properties are intended.

After multiple tables have been specified the relations between them have to be defined. This happens the same way as in the main editor with the help of edge elements. Figure 5 shows a possible structure for a first draft of a customer management.

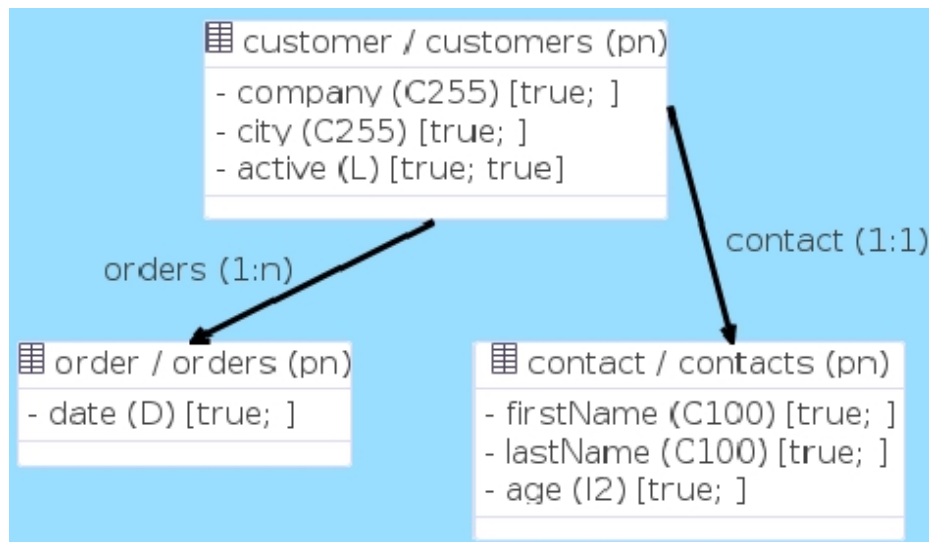


Figure 5: Persistence elements

Tables as well as fields have an attribute named „isLeading“ which tells that the corresponding element is primary. In this model this is true for the customer table and the fields „company“, „date“ and „lastName“. Beside modeling of table schemes the persistence editor further supports a container element for configuration vars. Defining these is on the lines of table fields, for a default module this is often not needed.

After saving and closing the sub editor the main editor is reloaded to synchronize the changes being performed. Now the data layer should be fine and only the process container should contain an error, because at least one processor is needed. Again a double click on the element opens a sub editor with the corresponding context.

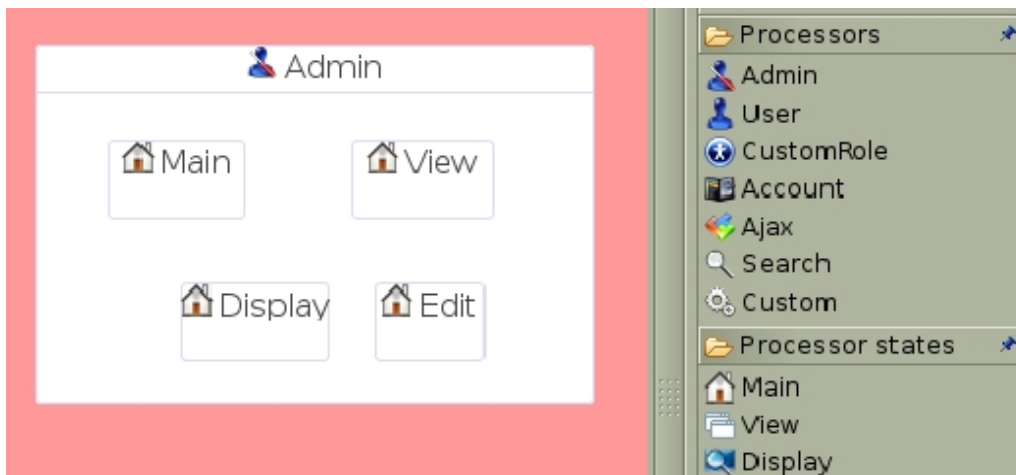


Figure 6: Process elements

In Figure 6 a processor and different states in it have been created. Normally relational connections are also used between states, for example to define redirections. Since the generator is not evaluating these transitions yet, they can be omitted. Also the process editor is saved and closed.

After the synchronization there shouldn't be any errors in the whole model. In this case the menu entry „File > Generate module“ is being activated taking care for executing the generator workflow. The user chooses the desired output directory and gets a confirmation about the generation progress.

Further information about ModuleStudio you can find at <http://modulestudio.de> – there is also a support forum.