# Data Model

TODO: sub model

The data\_model component provides the classes for other components to declare their part of the data model.

As mentioned before, each component must declare its data model (tables, format of columns, links between tables, permissions needed to access to tables or columns…). This will be used, first to ensure security checks so we can be sure that if a user should not access to specific data, even a component *forget* to check the permission, the access will be automatically denied. It will also be used to implement generic functionalities to manipulate data.

Each component can contain a file *datamodel.inc* which declares its part of the data model (the data the component is responsible for). For this it can use the following classes:

D:\_PN\student_soft\repositories\lecousin\doc\system\diagrams\Common_Data Model.wmf

## DataModel class

The DataModel class, together with Table class are described in the chapter “2.2 DataBase”: they allow each component to declare their part of the data model, building the complete data model of the application.

In addition to what has been previously described, *displayable data* can be declared on each table. A displayable data is a column of a table which aims at being displayed to the user (contrariwise the other columns are internal data, like foreign keys, which are not supposed to be displayed to the user). When declaring a displayable data, a localized name is provided, so the description of the data can be displayed.

Those displayable data will allow to implement generic pages, which can display data from one or several tables: the page will automatically adapt to the data model, by analyzing it. Thus a modification of the data model will be automatically reflected in those pages, without any additional code.

### Declaration from each component to build a complete data model

Each component declares its part of the data model, in a file *datamodel.inc*.

The class *DataModel* will then execute each of those files, to finally get the complete data model.

### Specific data for a project

Each PN project may have specific needs for the data. For example, in Cambodia, we would like to store the name of a person in English, but also its Khmer version. This is obviously not needed for Philippines or Vietnam.

In order to keep the databases of all project homogeneous, so all databases will have the same structure, some data may be hidden for some projects.

For this, a component may provide a file *datamodel\_hidden.inc*: it declares, for each project, which data should be hidden. Thus, the Khmer name column will be present in the database of each project, but will never be displayed when we are connected to the Philippines or Vietnam application.

This will allow flexibility for each project to have data which has not meaning or no interest for other projects, while keeping homogeneous database structures among the different projects.

## DataPath and DataPathBuilder

The class DataPathBuilder can automatically find the path from one table to another, or find all possible linked data from a given table, by analyzing the data model.

## Pages

### Entity

A portion of page is provided to display data from a given row of a table. Every displayable data of this table is displayed. If the current used has the permission to modify a data, the page will provide this possibility to the user: the data will be locked when the user starts to edit, to avoid another user attempt to edit the same data at the same time.

Each data will be displayed according to the type of the column: for example, if the column is a date, and the user wants to edit the data, a date picker (with a calendar) will be displayed so the user can easily pick a new date.

### Editable entity list

This page allows the user to edit a list of rows, add new rows, or remove rows.

This can be used typically when a table contains a list of entries, like a list of provinces in a country, so the user can easily edit this list.

In the same way as *Entity*, each data will be displayed/edited according to the type of the column. For instance, if a column is an integer, the user will be able only to enter digits.

### Data list

*DataList* implements a generic screen, where the user can see a table of data, select or not the columns to display, make searches, edit data… (a kind of Excel sheet).

This screen is using the data model defined by the components, to know what data are available, and what the type of each data is.

Then, using this screen will need only few lines: give what is the starting point in the data model, and what are the data displayed by default. For example, if we want a screen with a list of users, the starting point is the table of Users, then we will be able to see any data having relationship with a user (its personal information, through the component *People*, …) just by analyzing the data model. For this, DataList is using DataPathBuilder to know all the possible data which can be linked to the starting table, and how to reach all those data (following links, foreign keys…).

An example of usage, for a list of user:

$list = **new** DataList("Users");

$list->primary\_key("Users.domain","Users.username");

$list->add("Users.domain", **false**);

$list->add("Users.username", **false**);

$list->add("UserPeople.people>first\_name", **false**);

$list->add("UserPeople.people>last\_name", **false**);

$list->add("UserRole.role\_id>name", **false**);

Here, with only few lines, we indicate:

* The starting point is the table “Users”
* To uniquely identify a row (a user), we have to use the columns “domain” and “username” of the table “Users” (primary key)
* Then, by default, the table will contain the following information: “domain” from table “Users”; “username” from table “Users”; “first\_name” and “last\_name” from table “People” which can be reaches through “UserPeople”; “name” from table “Role” which can be reached through “UserRole” table.

The model being:



So for example, we can reach “first\_name” through the table “UserPeople” which is linked to the starting table “Users”.

## Services

Services, used by the previous pages or functionalities, are provided to:

* Lock a row in a table, or an entire table
* Release a lock
* Retrieve data for *DataList* (with columns selected by the user)

Export data from *DataList* in different formats like Excel or PDF.