



The symfony Cookbook

symfony 1.2

This PDF is brought to you by
SENSIOLABS 

License: Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 Unported License
Version: cookbook-1.2-en-2010-02-09

Table of Contents

How to write a Propel schema with the alternative syntax	3
Overview	3
Base example	3
Alternative syntax	4
Base example, with alternative syntax	4
Connection settings	5
Classes	6
Mixed schemas	6

Chapter 1

How to write a Propel schema with the alternative syntax

Overview

As of symfony 1.1, you can choose to describe your model's relational structure in a new YAML syntax. Symfony recognizes `schema.yml` files written either with the syntax described in Chapter 8 of the symfony Book, or with the syntax described below. The alternative syntax is more object-oriented and makes the merging process of several schemas easier to understand.

Base example

Consider the following schema, using the current syntax:

```
propel:
  _attributes:      { noXsd: false, defaultIdMethod: none, package:
lib.model }
  ab_group:
    _attributes:    { phpName: Group, package: foo.bar.lib.model }
    id:
    name:           varchar(50)

  cd_user:
    _attributes:    { phpName: User, isI18N: true, i18nTable: cd_user_i18n
}
    first_name:    { type: varchar, size: 255, default: "Anonymous" }
    last_name:     varchar(50)
    age:           { type: integer, required: true, index: true }
    ab_group_id:
    created_at:

  cd_user_i18n:
    description:   longvarchar

  ef_article:
    title:         { type: longvarchar, required: true, index: unique }
    stripped_title: { type: longvarchar, required: true, primaryKey: true,
sequence: my_custom_sequence_name }
```

*Listing
1-1*

```

    user_id:
    my_group:      { type: integer, foreignTable: ab_group,
foreignReference: id, onDelete: setnull }
    created_at:   timestamp
    updated_at:

ij_article:
  _attributes:   { phpName: Article }
  title:        varchar(50)
  user_id:      { type: integer }
  _foreignKeys:
    -
      foreignTable: cd_user
      onDelete:    cascade
      references:
        - { local: user_id, foreign: id }
  created_at:
  _indexes:
    my_index:    [title, user_id]
  _uniques:
    my_other_index: [created_at]
  _behaviors:
    paranoid:    { column: deleted_at }

ab_group_i18n:
  motto:        longvarchar

```

Alternative syntax

Base example, with alternative syntax

Here is how to write exactly the same structure as the one listed above with the alternative syntax:

```

Listing 1-2
connection:      propel
noXsd:          false
defaultIdMethod: none
package:        lib.model

classes:
  Group:
    tableName:   ab_group
    package:     foo.bar.lib.model
    columns:
      id:
      name:      varchar(50)

  User:
    tableName:   cd_user
    isI18N:     true
    i18nTable:  cd_user_i18n
    columns:
      first_name: { type: varchar, size: 255, default: "Anonymous" }
      last_name:  varchar(50)
      age:        { type: integer, required: true, index: true }
      ab_group_id:

```

```

    created_at:

CdUserI18n:
  columns:
    description:    longvarchar

EfArticle:
  columns:
    title:          { type: longvarchar, required: true, index: unique }
    stripped_title: { type: longvarchar, required: true, primaryKey:
true, sequence: my_custom_sequence_name }
    user_id:
    my_group:       { type: integer, foreignClass: Group,
foreignReference: id, onDelete: setnull }
    created_at:     timestamp
    updated_at:

Article:
  tableName:       ij_article
  columns:
    title:          varchar(50)
    user_id:        { type: integer }
    created_at:
  foreignKeys:
    -
      foreignTable: cd_user
      onDelete:     cascade
      references:
        - { local: user_id, foreign: id }
  indexes:
    my_index:       [title, user_id]
  uniques:
    my_other_index: [created_at]
  behaviors:
    paranoid:       { column: deleted_at }

AbGroupI18n:
  columns:
    motto:          longvarchar

```

The main difference is that you declare classes, not tables, using the table `phpName` as a key.

This alternative syntax is also more explicit, since you must create entries for **classes** and **columns**. But it gets rid of the ugly `_attributes` hack of the current syntax, so a `schema.yml` doesn't try to mimick an XML syntax.

Last but not least, all the 'magic' of the classic syntax is still there (auto definition of primary keys, foreign keys, i18n tables, etc.).

Connection settings

Instead of being defined as `_attributes` of the connection, the connection settings, together with the connection name, are all level-1 keys:

```

connection:        propel
noXsd:             false
defaultIdMethod:   none
package:           lib.model

```

*Listing
1-3*

All these keys are optional, including the `connection` one. If it is not set, symfony will take `propel` as a default value.

Classes

A class definition lists the table name in the database, the columns, foreign keys, indexes and behaviors in a natural key/values syntax:

```
Listing 1-4 Article:
  tableName:      ij_article
  columns:
    title:        varchar(50)
    user_id:      { type: integer }
    created_at:
  foreignKeys:
    -
      foreignTable: cd_user
      onDelete:    cascade
      references:
        - { local: user_id, foreign: id }
  indexes:
    my_index:     [title, user_id]
  uniques:
    my_other_index: [created_at]
  behaviors:
    paranoid:     { column: deleted_at }
```

Note that you can define foreign keys either with the usual `foreignTable` attribute, which expects a table name, or via the new `foreignClass` attribute, which expects a class name.

Mixed schemas

You can have, in a project, schemas with mixed current and alternative syntax.

The schema extension system, described in Chapter 17 of the symfony book, works whatever the original schema syntax and whatever the custom schemas syntax. This means that you can customize an existing schema with the classic syntax using a custom schema with the alternative syntax, and vice-versa. Symfony will do the conversion internally so that the merge is always possible.

Note that the schema merge is easier to understand when considering the alternative syntax for both the original and the custom schema. In fact, this is the internal format used by symfony for the merge. The following listing shows how schemas are merged:

```
Listing 1-5 # Original schema, in plugins/myPlugin/config/schema.yml
classes:
  User:
    tableName:      cd_user
    columns:
      first_name:   { type: varchar, size: 255, default: "Anonymous" }
      last_name:    varchar(50)
      age:          { type: integer, required: true, index: true }
      created_at:
  Article:
    tableName:      ij_article
    columns:
```

```

    title:          varchar(50)
    user_id:        { type: integer }
    created_at:
    foreignKeys:
      -
        foreignTable: cd_user
        onDelete:     cascade
        references:
          - { local: user_id, foreign: id }

# Custom schema, in myPlugin_schema.custom.yml
connection: myConnection
classes:
  Group:
    tableName:      ab_group
    package:         foo.bar.lib.model
    behaviors:       [paranoid]
    columns:
      id:
        name:        varchar(50)

  User:
    tableName:       ef_user
    isI18N:          true
    i18nTable:       cd_user_i18n
    columns:
      ab_group_id:

  Article:
    columns:
      updated_at:

# Resulting schema, merged internally and used for model and sql generation
connection: myConnection
classes:
  Group:
    tableName:      ab_group
    package:         foo.bar.lib.model
    behaviors:       [paranoid]
    columns:
      id:
        name:        varchar(50)

  User:
    tableName:       cd_user
    isI18N:          true
    i18nTable:       cd_user_i18n
    columns:
      first_name:    { type: varchar, size: 255, default: "Anonymous" }
      last_name:     varchar(50)
      age:           { type: integer, required: true, index: true }
      ab_group_id:
      created_at:

  Article:
    tableName:       ij_article
    columns:
      title:         varchar(50)

```

```
user_id:      { type: integer }
created_at:
updated_at:
foreignKeys:
-
  foreignTable: cd_user
  onDelete:    cascade
  references:
    - { local: user_id, foreign: id }
```

For clarity, it is recommended to use the alternative schema syntax as much as possible.

