MODEL

Symfony uses, by default, Propel as the ORM and Creole as the database abstraction layer.
More about propel: http://propel.phpdb.org/docs/user_guide/

DATABASES SUPPORTED (Creole)

- MySQL
- PostgreSQL
- Oracle
- SQLServer
- SQLITE

CONNECT TO DATABASE

propel.ini — /myproject/config

```
propel.targetPackage          = lib.model
propel.packageObjectModel     = true
propel.project                = myproject
propel.database               = mysql
propel.database.createUrl     = mysql://localhost/
propel.database.url       = mysql://localhost/myproject
...
```

databases.yml — /myproject/config

```
prod:
  propel:
    param:
      host:         mydataserver
      username:     myusername
      password:     mypassword

all:
  propel:
    class:          sfPropelDatabase
    param:
      phptype:      mysql
      hostspec:     localhost
      database:     blog
      username:     login
      password:     passwd
      port:         80
      encoding:     utf8
      persistent:   true
```

You can define distinct settings for the prod, dev, and test environments, or any other environment in your application. This configuration can also be overridden per application, in **apps/myapp/config/databases.yml**

TRANSACTIONS

```
public function save($con = null){
  $con = Propel::getConnection();
  try{
    $con->begin();
    $ret = parent::save($con);
     // update interested_users in question table
    $question = $this->getQuestion();
    $interested_users=$question->getInterestedUsers();
    $question->setInterestedUsers($interested_users+1);
    $question->save($con);
    $con->commit();
    return $ret;
  }
  catch (Exception $e){
    $con->rollback();
    throw $e;
  }
}
```

MODEL CLASSES

The schema is used to build the model classes of the ORM layer through the command-line task: **$ symfony propel-build-model**

BASE CLASSES — lib/model/om/

`BaseArticle.php` `BaseComment.php`
`BaseArticlePeer.php` `BaseCommentPeer.php`

Base classes are the ones directly generated from the schema. **Never modify them,** since every new build of the model will completely erase these files.

DATA MODEL CLASSES — lib/model/

`Article.php` `Comment.php`
`ArticlePeer.php` `CommentPeer.php`

Inherit from the Base ones. When the propel-build-model task is called on an existing model, these classes **are not modified**. So this is where you can **add custom methods and properties to the model objects.**

Example: here is the content of the newly created Article.php file:

```
require_once 'model/om/BaseArticle.php';
class Article extends BaseArticle{
}
```

It inherits all the methods of the BaseArticle class, but a modification in the model will not affect it.

OBJECT CLASSES — Article.php, Comment.php

Represent a record in the database. They give access to the columns of a record and to related records.

Object Class Constructor - new

To create a new object:

```
$myobject = new MyTable();
```

PEER CLASSES — ArticlePeer.php, CommentPeer.php

Contain static methods to operate on the tables. Provide a way to retrieve records from the tables. Their methods usually return an object or a collection of objects of the related object class.

The methods of the Peer classes will be called with a :: (*for static method call*) instead of the usual -> (*for instance method call*)

Retrieving Records by Primary Key

```
$myobject=myTablePeer::retrieveByPk(7);
```

retriving by primary key that consist of more than one column:

```
$myobject=myTablePeer::retrieveByPk(1, 12);
```

select multiple objects based on their primary keys:

```
$myobject=myTablePeer::retrieveByPKs($arrayOfPrimaryKeys);
```

Retrieving Records with Criteria

```
$c = new Criteria();
$c->add(CommentPeer::AUTHOR, 'Steve');
$c->addAscendingOrderByColumn(CommentPeer::DATE);
$comments = CommentPeer::doSelect($c);
//$comments is an array of objects of class Comment
```

METADATA CLASSES — lib/model/map/

`ArticleMapBuilder.php` `CommentMapBuilder.php`

Contains metadata information about the table that is needed for the runtime environment.

Creole Column Types

BOOLEAN = 1	VARBINARY = 13
BIGINT = 2	NUMERIC = 14
SMALLINT = 3	BLOB = 15
TINYINT = 4	CLOB = 16
INTEGER = 5	LONGVARCHAR = 17
CHAR = 6	DECIMAL = 18
VARCHAR = 7	REAL = 19
TEXT = 17	BINARY = 20
FLOAT = 8	LONGVARBINARY = 21
DOUBLE = 9	YEAR = 22
DATE = 10	ARR = 23
TIME = 11	OTHER = -1
TIMESTAMP = 12	

Special Date Columns

created_at
store a timestamp of the date when the record was created

updated_at
updated each time the record itself is updated

Object Class Methods

The **accessors** and **mutators** use a **camelCase** variant of the column names, so the ***getTitle() method*** retrieves the value of the ***title column***.

Accessors:

get[MyColumnName]()
Retrieve the column value

```
$myobject->getMyColumn();
```

getByName($name)

Mutators:

set[MyColumnName]($value)
To set one field:

```
$myobject->setMyColumn("value");
```

fromArray($array)
To set several fields at one time:

```
$myobject->fromArray(array(
  'myColumn1' => 'Some content',
  'myColumn2' => 'Some content',
));
```

setByName($name, $value)

save()
save the data into the database

```
$myobject->save();
```

isNew()
check if an object is new

```
$myobject->isNew();
```

isModified()
check if an object has been modified and deserves saving

```
$myobject->isModified();
```

delete()
delete records from the database

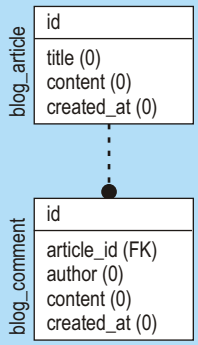
```
$myobject->delete();
```

isDeleted()
check if an object is deleted in database

```
$myobject->isDeleted();
```

DATABASE SCHEMA (sample) — myproject/config/

table structure



schema.yml

```
propel:
  blog_article:
    _attributes: { phpName: Article }
    id:
    title:      varchar(255)
    content:    longvarchar
    created_at:
  blog_comment:
    _attributes: { phpName: Comment }
    id:
    article_id:
    author:     varchar(255)
    content:    longvarchar
    created_at:
```

schema.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<database name="propel" defaultIdMethod="native" noXsd="true" package="lib.model">
  <table name="blog_article" phpName="Article">
    <column name="id" type="integer" required="true" primaryKey="true"autoIncrement="true" />
    <column name="title" type="varchar" size="255" />
    <column name="content" type="longvarchar" />
    <column name="created_at" type="timestamp" />
  </table>
  <table name="blog_comment" phpName="Comment">
    <column name="id" type="integer" required="true" primaryKey="true"autoIncrement="true" />
    <column name="article_id" type="integer" />
    <foreign-key foreignTable="blog_article">
      <reference local="article_id" foreign="id"/>
    </foreign-key>
    <column name="author" type="varchar" size="255" />
    <column name="content" type="longvarchar" />
    <column name="created_at" type="timestamp" />
  </table>
</database>
```