# Sexy ORM --- 全面革新的 ORM 系统

**作者：**廖宇雷

**前言：**

QeePHP 提供了一个全功能的 ORM 系统，能够完善处理各种对象间的复杂关系，为开发者采用面向对象设计来解决业务问题提供了有力的支持。不过实践中，我们也发现 QeePHP 现有的 ORM 系统存在各种不足。因此，全面革新 ORM 系统成为了一项重要的任务。该任务名为：Sexy ORM。

Sexy ORM 是 QeePHP 下一个版本的重点之一，目标是为 QeePHP 提供一个更容易理解和使用的 ORM 架构。在 QeePHP 现有 ORM 架构的基础之上，Sexy ORM 更符合面向对象开发的思想，并且为团队协作和高效开发提供了更多的支持。

### 清晰的模型定义

QeePHP 目前使用的 ORM 中，模型的属性、关联等是在模型类的 \_\_define() 静态方法中定义的。

这种定义方式不够直观，而且还有下列弊端：

* 违背了面向对象设计的初衷，强迫开发者回到以数据库为中心的设计方式；
* 模型代码中的属性定义不完整，如果不查看数据表无法知道模型到底有哪些属性；
* 当模型或数据表改变时，很可能出现模型与数据表不一致的情况；
* 无法通过反射等机制完成模型的自动化维护，给团队协作带来了困难。

在 Sexy ORM 中，将采用更清晰的模型定义方式：

<?php

/\*\*

\* 作者

\*

\* @table authors

\*/

class Author extends QModel

{

/\*\*

\* ID

\*

\* @var int

\* @pk

\*/

public $id;

/\*\*

\* 名

\*

\* @var string

\* @size 60

\*/

public $first\_name;

/\*\*

\* 姓

\*

\* @var string

\* @size 60

\*/

public $last\_name;

/\*\*

\* 电子邮件

\*

\* @var string

\* @optional

\* @size 80

\*

\* @validate\_is\_email '必须是有效的电子邮件地址'

\*/

public $email;

/\*\*

\* 该作者参与撰写的书籍

\*

\* @var coll of Book

\* @has\_many :through writings # 通过 writing 属性关联到书籍

\*/

public $books;

/\*\*

\* 该作者的写作记录

\*

\* @var coll of Writing

\* @has\_many :cascade destroy # 删除作者时，删除作者的撰写记录

\* :order 'finished\_date DESC' # 按照撰写完成时间排序

\* :limit 5 # 默认只读取作者最近 5 个撰写记录

\*/

public $writings;

/\*\*

\* 全名

\*

\* @var string

\* @getter get\_full\_name

\*/

public $full\_name;

/\*\*

\* 返回作者的全名

\*

\* @return string

\*/

function get\_full\_name()

{

return $this->first\_name . ', ' . $this->last\_name;

}

.... 其他领域逻辑方法 ....

}

新的定义方式将模型的所有属性都直接写了出来，而且模型间的关联也通过属性来直接定义。

定义属性时，每一个属性的特性都使用注解（@ 开头的注释文字）来描述。示例中用 @var 指定了属性的类型，@size 则指定了属性值的最大长度。除了属性，模型类的特性也使用注解来描述，例如 @table 指定了存储该模型的数据表名称。

新的模型定义具有下列特点：

* 属性由对象成员变量直接定义；
* 基于注解的属性描述；
* 基于注解的类定义描述。

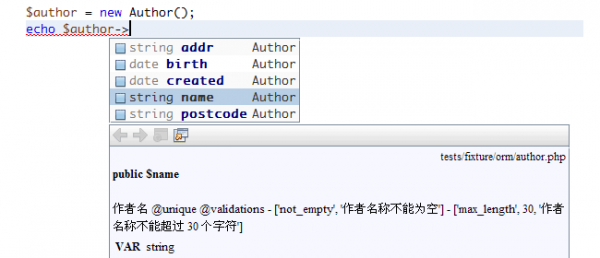
由于采用了与现在完全不同的模型定义方式，因此在 Sexy ORM 中，模型不再是从数据表创建，而是由开发人员来创建。创建好的模型以最自然的方式来勾勒出模型的概貌，并且通过注解明确模型的方方面面。

### 新模型定义带来的好处

新的模型定义方式，带来了一些显著的好处：

* 模型的定义完全以面向对象为出发点，在设计阶段排除了数据库；
* 所有的属性通过阅读源代码即获知，实现设计上的“所见即所得”；
* 模型的变化可以自动映射到数据表之上，配合数据库迁移脚本，可以消除模型与数据库不一致的情况；
* 基于注解的描述，允许我们使用自动化工具完成模型的维护工作；
* 模型的属性与关联定义更直观，易于阅读和理解。

而且新的模型定义方式可以让 IDE 自动提示对象属性，进一步提高了开发效率：



### 新的模型创建流程

目前，我们的创建一个模型的流程是：做初步设计、建立数据表、从数据表生成模型文件。

这种流程最大的问题就在于设计数据表变成了设计模型的一部分，而模型原本应该是完全从面向对象角度来思考的，不适宜过早和数据表发生关系。

改善后的模型创建流程是：

1. 设计时，以模型为关注焦点；
2. 创建模型的类文件；
3. 自动从模型定义生成数据库迁移脚本；
4. 通过数据库迁移脚本来创建数据表。

这里提到的数据库迁移脚本是一种自动化工具，具备下列功能：

* 分析模型类，确定模型的所有属性及属性类型等信息；
* 生成创建数据表的脚本代码；
* 当模型发生变化后，可以自动比对模型与数据表的结构，并生成更新数据表的脚本代码。

数据库迁移脚本的引入，使得团队协作时，模型的维护工作得到了显著简化。更新模型后，开发者可以通过自动化工具生成一个新的数据库迁移脚本。

由于团队成员之间利用 Subversion 等源代码版本控制系统进行代码的更新，因此一个开发者对模型修改后产生的源文件和数据库迁移脚本都能够同步到其他团队成员那里。最终，只需要执行数据库迁移脚本，模型更新的同步工作即可轻松完成。

### 基于 PHP 5.3

Sexy ORM 是为 PHP 5.3 及更新版本 PHP 所设计的，充分利用了新版本 PHP 的功能。这些新功能包括延迟静态绑定、静态魔法调用、名字空间等。这些新功能的应用，让我们的模型类变得更整洁，而且使用时具备更佳的灵活性。

Sexy ORM 相对于 QeePHP 现有 ORM 的新特征：

* 基于 PHP 5.3；
* 所见即所得风格的模型定义；
* 全面的自动化工具；
* 更完善的查询支持；
* 更多的动态查找方法；
* 更好的关联处理；
* 更简洁的模型类定义；
* 更好的 IDE 支持；
* 完善的反射支持；
* 性能改进。

虽然 Sexy ORM 的许多新特征依赖于 PHP 5.3 版本，但由于在基础架构和实现上和现有 ORM 的传承，Sexy ORM 的一些特性也将陆续出现在 QeePHP 现有的 ORM 系统中。这些即将出现的新特征包括：

* 数据库迁移脚本支持；
* 更完善的查询支持；
* 更好的关联处理；
* 性能改进。

### 路线图

虽然 PHP 5.3 近期已经发布了 RC1 版本。但根据以往的经验，PHP 每一次重大升级的版本都需要两个较小版本的更新后才能稳定下来，因此 PHP 5.3.2 或更新的版本将会步入成熟期。

我们预计在 2010 年上半年，PHP 5.3 系列会开始普及，此时也将是 Sexy ORM 正式推出的时机。

### QeePHP介绍

回顾一下曾经开发过的 PHP 应用，大部分开发者都会发现这些应用中，数据的创建、读取、更新和删除操作是重复最多次的操作。但是不管我们如何简化这些 CRUD（创建、读取、更新、删除）操作，面对客户不断变化的需求，应用程序的内在结构总是逐渐变得凌乱。

而造成这种情况的根本原因就是我们没有正确使用面向对象的技术来设计和实现这些应用程序。由于业务逻辑固有的复杂性被所谓的 CRUD 快速开发能力所掩盖。本应是内聚的业务逻辑却拆散为一个个 CRUD 操作，分散到了应用程序的不同部分。如此一来，业务逻辑的变化必然会导致应用程序内部产生连锁反应式的改动，应用程序内在结构的逐渐腐朽变成了不可逆转的趋势。

QeePHP 是一个遵循“领域驱动设计”思想设计和实现的 PHP 开发框架。在保持易学易用、快速稳定的同时，QeePHP 为创建具有复杂内在逻辑的 Web 应用提供了全面的支持。

“领域驱动设计”是一种设计和架构思想，以“模型”为核心，将业务逻辑封装到一个个内聚的模型之中，从而避免业务逻辑被分散到应用程序的各个部分。这样一来，不管是增加了新的业务需求还是已有需求的改变，我们都能够将修改局限在一个相对较小的范围。从根本上避免了需求快速更新导致应用程序内在结构逐渐腐朽的问题。

更多有关 QeePHP 的信息，请访问： <http://qeephp.com/>