Atitit orm对象关系映射

# ORM中的Active Record和Data Mapper

Ruby和Laravel的ORM都采取了Active Record的模式，因此它们ORM可能像下面这样：

$user = new User;$user->username = 'philipbrown';$user->save();

领域对象直接对应着数据库中的一个表：

+----+-------------+| id | username |

+----+-------------+| 1 | philipbrown |

+----+-------------+

我们再来看使用Data Mapper的ORM是怎样的：

$user = new User;$user->username = 'philipbrown';

EntityManager::persist($user);

现在我们察看到了它们最基本的区别：在Active Record中，领域对象有一个save()方法，领域对象通常会继承一个ActiveRecord的基类来实现。而在Data Mapper模式中，领域对象**不存在**save()方法，持久化操作由一个中间类来实现。

如果使用领域模型，还会有更多的选择。在简单应用中，领域模型是一种和数据库结构一致的简单结构，对应每个数据库表都有一个领域类，在这种情况下，有必要让每个对象负责数据库的存取过程，这也就是Active Record（活动记录）。

领域对象直接鱼数据库表进行交互，这带来了一个问题，随着领域逻辑变的更加复杂，它就慢慢转变成一个大的领域模型，简单的Active Record已经不能满足要求了，领域类和表一对一匹配也开始随着把领域逻辑放入更小的类而失效。关系数据库无法处理继承，因此使用策略模式等面相对象模式非常困难。

一种更好的办法是把数据库和数据库完全独立，让间接层完全领域对象和数据库表之间的映射，这个映射类也称作Data Mapper（数据映射器）。这个映射类处理数据库和领域模型之间所有的存取操作，并且允许双方都能独立变化。

如果领域逻辑非常简单并且类和表十分一致，使用简单的Active Record就足够了。如果领域逻辑比较复杂，Data Mapper则可能是需要的。

# 结构映射

## 关系映射

在ORM中，通常还要关注关系映射的问题，譬如回到上面的例子，一个User可能有一些Post，而且它们往往在其他的表中，这就需要处理跨表的映射关系。

$posts = $user->posts;

+----+---------+-------+| id | user\_id | title |

+----+---------+-------+| 1 | 2 | "..." |

+----+---------+-------+

就上面这种一对多的关系来说，我们可以通过对象中的一个标识域来保持对象之间的关系特性，这种标识域在数据表中直接的体现就是外键，通过外键来得到对象间的引用。

而在多对多的关系中，因为两边都存在着集合，就用学生－课程来说，一个学生可能有多门课程，而一个课程可能被多个学生选择，这个时候，用一个关联对象抽取出原来的外键则是更好的选择，这种关联对象在数据库中扮演着一个**关联表**。

## 继承

在SQL里没有用于继承的标准方法，所以处理领域对象的继承时要十分留意，对于任何继承结构一般有三种选择：**单表继承**：为一个层次所有的类建立一个表，**具体表继承**：为每个具体类建立一个表，**类表继承**：为这个层次的每一个类建立一个表。