一个例子

1. [准备数据库](http://www.yiichina.com/doc/guide/2.0/start-databases#preparing-database)
2. [配置数据库连接](http://www.yiichina.com/doc/guide/2.0/start-databases#configuring-db-connection)
3. [创建活动记录](http://www.yiichina.com/doc/guide/2.0/start-databases#creating-active-record)
4. [创建动作](http://www.yiichina.com/doc/guide/2.0/start-databases#creating-action)
5. [创建视图](http://www.yiichina.com/doc/guide/2.0/start-databases#creating-view)
6. [试运行](http://www.yiichina.com/doc/guide/2.0/start-databases#trying-it-out)
7. [总结](http://www.yiichina.com/doc/guide/2.0/start-databases#summary)

本节将介绍如何如何创建一个从数据表 country 中读取国家数据并显示出来的页面。 为了实现这个目标，你将会配置一个数据库连接， 创建一个[活动记录](http://www.yiichina.com/doc/guide/2.0/db-active-record)类， 并且创建一个[操作](http://www.yiichina.com/doc/guide/2.0/structure-controllers)及一个[视图](http://www.yiichina.com/doc/guide/2.0/structure-views)。

准备数据库：

首先创建一个名为 yii2basic 的数据库，应用将从这个数据库中读取数据。 你可以创建 SQLite，MySQL，PostregSQL，MSSQL 或 Oracle 数据库，Yii 内置多种数据库支持。简单起见，后面的内容将以 MySQL 为例做演示。

然后在数据库中创建一个名为 country 的表并插入简单的数据。可以执行下面的语句：

**CREATE** **TABLE** `country` (

`code` CHAR(2) **NOT** NULL PRIMARY **KEY**,

`name` CHAR(52) **NOT** NULL,

`population` INT(11) **NOT** NULL **DEFAULT** '0'

) **ENGINE**=**InnoDB** **DEFAULT** **CHARSET**=utf8;

**INSERT** **INTO** `country` **VALUES** ('AU','Australia',18886000);

**INSERT** **INTO** `country` **VALUES** ('BR','Brazil',170115000);

**INSERT** **INTO** `country` **VALUES** ('CA','Canada',1147000);

**INSERT** **INTO** `country` **VALUES** ('CN','China',1277558000);

**INSERT** **INTO** `country` **VALUES** ('DE','Germany',82164700);

**INSERT** **INTO** `country` **VALUES** ('FR','France',59225700);

**INSERT** **INTO** `country` **VALUES** ('GB','United Kingdom',59623400);

**INSERT** **INTO** `country` **VALUES** ('IN','India',1013662000);

**INSERT** **INTO** `country` **VALUES** ('RU','Russia',146934000);

**INSERT** **INTO** `country` **VALUES** ('US','United States',278357000);

此时便有了一个名为 yii2basic 的数据库，在这个数据库中有一个包含三个字段的数据表 country，表中有十行数据。

配置数据库连接：

开始之前，请确保你已经安装了 PHP [PDO](http://www.php.net/manual/en/book.pdo.php) 扩展和你所使用的数据库的 PDO 驱动（例如 MySQL 的 pdo\_mysql）。 对于使用关系型数据库来讲，这是基本要求。

驱动和扩展安装可用后，打开 config/db.php 修改里面的配置参数对应你的数据库配置。 该文件默认包含这些内容：

<?php

**return** [

'class' => 'yii\db\Connection',

'dsn' => 'mysql:host=localhost;dbname=yii2basic',

'username' => 'root',

'password' => '',

'charset' => 'utf8',

];

config/db.php 是一个典型的基于文件的[配置](http://www.yiichina.com/doc/guide/2.0/concept-configurations)工具。 这个文件配置了数据库连接 [yii\db\Connection](http://www.yiichina.com/doc/api/2.0/yii-db-connection) 的创建和初始化参数， 应用的 SQL 查询正是基于这个数据库。

上面配置的数据库连接可以在应用中通过 Yii::$app->db 表达式访问。

**信息：**config/db.php 将被包含在应用配置文件 config/web.php 中， 后者指定了整个[应用](http://www.yiichina.com/doc/guide/2.0/structure-applications)如何初始化。 请参考[配置](http://www.yiichina.com/doc/guide/2.0/concept-configurations)章节了解更多信息。

创建活动记录（models下面）：

创建一个继承自[活动记录](http://www.yiichina.com/doc/guide/2.0/db-active-record)类的类 Country， 把它放在 models/Country.php 文件，去代表和读取 country 表的数据。

<?php

**namespace** **app**\**models**;

**use** **yii**\**db**\**ActiveRecord**;

**class** **Country** **extends** **ActiveRecord**

{

}

这个 Country 类继承自 [yii\db\ActiveRecord](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord)。你不用在里面写任何代码。 只需要像现在这样，Yii 就能根据类名去猜测对应的数据表名。

**信息：**如果类名和数据表名不能直接对应， 可以覆写 [tableName()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord" \l "tableName()-detail) 方法去显式指定相关表名。

使用 Country 类可以很容易地操作 country 表数据，就像这段代码：

**use** **app**\**models**\**Country**;

// 获取 country 表的所有行并以 name 排序

$countries = Country::find()->orderBy('name')->all();

// 获取主键为 “US” 的行

$country = Country::findOne('US');

// 输出 “United States”

**echo** $country[‘name’];

// 修改 name 为 “U.S.A.” 并在数据库中保存更改

$country->name = 'U.S.A.';

$country->save();

**信息：**活动记录是面向对象、功能强大的访问和操作数据库数据的方式。你可以在[活动记录](http://www.yiichina.com/doc/guide/2.0/db-active-record)章节了解更多信息。 除此之外你还可以使用另一种更原生的被称做[数据访问对象](http://www.yiichina.com/doc/guide/2.0/db-dao)的方法操作数据库数据。

创建动作（controllers下面）：

为了向最终用户显示国家数据，你需要创建一个操作。相比之前小节掌握的在 site 控制器中创建操作， 在这里为所有和国家有关的数据新建一个控制器更加合理。 新控制器名为 CountryController，并在其中创建一个 index 操作， 如下：

<?php

**namespace** **app**\**controllers**;

**use** **yii**\**web**\**Controller**;

**use** **yii**\**data**\**Pagination**;

**use** **app**\**models**\**Country**;

**class** **CountryController** **extends** **Controller**

{

**public** **function** **actionIndex**()

{

$query = Country::find();

$pagination = **new** Pagination([

'defaultPageSize' => 5,

'totalCount' => $query->count(),

]);

$countries = $query->orderBy('name')

->offset($pagination->offset)

->limit($pagination->limit)

->all();

**return** $this->render('index', [

'countries' => $countries,

'pagination' => $pagination,

]);

}

}

把上面的代码保存在 controllers/CountryController.php 文件中。

index 操作调用了活动记录 Country::find() 方法，去生成查询语句并从 country 表中取回所有数据。 为了限定每个请求所返回的国家数量，查询在 [yii\data\Pagination](http://www.yiichina.com/doc/api/2.0/yii-data-pagination) 对象的帮助下进行分页。 Pagination 对象的使命主要有两点：

* 为 SQL 查询语句设置 offset 和 limit 从句， 确保每个请求只需返回一页数据（本例中每页是 5 行）。
* 在视图中显示一个由页码列表组成的分页器， 这点将在后面的段落中解释。

在代码末尾，index 操作渲染一个名为 index 的视图， 并传递国家数据和分页信息进去。

创建视图（views下面）：

在 views 目录下先创建一个名为 country 的子目录。 这个目录存储所有由 country 控制器渲染的视图。在 views/country 目录下 创建一个名为 index.php 的视图文件，内容如下：

<?php

**use** **yii**\**helpers**\**Html**;

**use** **yii**\**widgets**\**LinkPager**;

?>

<h1>Countries</h1>

<ul>

<?php **foreach** ($countries **as** $country): ?>

<li>

<?= Html::encode("{$country->name} ({$country->code})") ?>:

<?= $country->population ?>

</li>

<?php **endforeach**; ?>

</ul>

<?= LinkPager::widget(['pagination' => $pagination]) ?>

这个视图包含两部分用以显示国家数据。第一部分遍历国家数据并以无序 HTML 列表渲染出来。 第二部分使用 [yii\widgets\LinkPager](http://www.yiichina.com/doc/api/2.0/yii-widgets-linkpager) 去渲染从操作中传来的分页信息。 小部件 LinkPager 显示一个分页按钮的列表。 点击任何一个按钮都会跳转到对应的分页。

Active Record（活动记录）

1. [声明 AR 类](http://www.yiichina.com/doc/guide/2.0/db-active-record#sheng-ming-ar-lei)
2. [访问列数据](http://www.yiichina.com/doc/guide/2.0/db-active-record#fang-wen-lie-shu-ju)
3. [建立数据库连接](http://www.yiichina.com/doc/guide/2.0/db-active-record#jian-li-shu-ju-ku-lian-jie)
4. [查询数据](http://www.yiichina.com/doc/guide/2.0/db-active-record#cha-xun-shu-ju)
5. [操作数据](http://www.yiichina.com/doc/guide/2.0/db-active-record#cao-zuo-shu-ju)
6. [AR的生命周期](http://www.yiichina.com/doc/guide/2.0/db-active-record#ar-de-sheng-ming-zhou-qi)
7. [查询关联的数据](http://www.yiichina.com/doc/guide/2.0/db-active-record#cha-xun-guan-lian-de-shu-ju)
8. [中间关联表](http://www.yiichina.com/doc/guide/2.0/db-active-record#zhong-jian-guan-lian-biao)
9. [延迟加载和即时加载（又称惰性加载与贪婪加载）](http://www.yiichina.com/doc/guide/2.0/db-active-record#yan-chi-jia-zai-he-ji-shi-jia-zai-you-cheng-duo-xing-jia-zai-yu-tan-lan-jia-zai)
10. [逆关系](http://www.yiichina.com/doc/guide/2.0/db-active-record#ni-guan-xi)
11. [JOIN 类型关联查询](http://www.yiichina.com/doc/guide/2.0/db-active-record#join-lei-xing-guan-lian-cha-xun)
12. [关联表操作](http://www.yiichina.com/doc/guide/2.0/db-active-record#guan-lian-biao-cao-zuo)
13. [作用域](http://www.yiichina.com/doc/guide/2.0/db-active-record#zuo-yong-yu)
14. [事务操作](http://www.yiichina.com/doc/guide/2.0/db-active-record#shi-wu-cao-zuo)
15. [乐观锁（Optimistic Locks）](http://www.yiichina.com/doc/guide/2.0/db-active-record#le-guan-suo-optimistic-locks)
16. [被污染属性](http://www.yiichina.com/doc/guide/2.0/db-active-record#bei-wu-ran-shu-xing)
17. [另见](http://www.yiichina.com/doc/guide/2.0/db-active-record#ling-jian)

[Active Record](http://zh.wikipedia.org/wiki/Active_Record) （活动记录，以下简称AR）提供了一个面向对象的接口， 用以访问数据库中的数据。一个 AR 类关联一张数据表， 每个 AR 对象对应表中的一行，对象的属性（即 AR 的特性Attribute）映射到数据行的对应列。 一条活动记录（AR对象）对应数据表的一行，AR对象的属性则映射该行的相应列。 您可以直接以面向对象的方式来操纵数据表中的数据， 再不用担心需要写原生 SQL 语句啦。

例如，假定 Customer AR 类关联着 customer 表， 且该类的 name 属性代表 customer 表的 name 列。 你可以写以下代码来向 customer 表里插入一行新的记录:

$customer = **new** Customer();

$customer->name = 'Qiang';

$customer->save(); // 一行新数据插入 customer 表

上面的代码和使用下面的原生 SQL 语句是等效的，但显然前者更直观， 更不易出错，并且面对不同的数据库系统（DBMS, Database Management System）时更不容易产生兼容性问题。

下面是所有目前被 Yii 的 AR 功能所支持的数据库列表：

* MySQL 4.1 及以上：通过 [yii\db\ActiveRecord](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord)
* PostgreSQL 7.3 及以上：通过 [yii\db\ActiveRecord](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord)
* SQLite 2 和 3：通过 [yii\db\ActiveRecord](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord)
* Microsoft SQL Server 2010 及以上：通过 [yii\db\ActiveRecord](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord)
* Oracle: 通过 [yii\db\ActiveRecord](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord)
* CUBRID 9.1 及以上：通过 [yii\db\ActiveRecord](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord) (Note that due to a [bug](http://jira.cubrid.org/browse/APIS-658) in the cubrid PDO extension, quoting of values will not work, so you need CUBRID 9.3 as the client as well as the server)
* Sphinx：通过 yii\sphinx\ActiveRecord，需求 yii2-sphinx 扩展
* ElasticSearch：通过 yii\elasticsearch\ActiveRecord，需求 yii2-elasticsearch 扩展

Additionally, Yii also supports using Active Record with the following NoSQL databases:

* Redis 2.6.12 及以上：通过 yii\redis\ActiveRecord，需求 yii2-redis 扩展
* MongoDB 1.3.0 及以上：通过 yii\mongodb\ActiveRecord，需求 yii2-mongodb 扩展

如你所见，Yii 不仅提供了对关系型数据库的 AR 支持，还提供了 NoSQL 数据库的支持。 在这个教程中，我们会主要描述对关系型数据库的 AR 用法。 然而，绝大多数的内容在 NoSQL 的 AR 里同样适用。

### 声明AR类(models下面)

要想声明一个 AR 类，你需要扩展 [yii\db\ActiveRecord](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord) 基类， 并实现 tableName 方法，返回与之相关联的的数据表的名称：

**namespace** **app**\**models**;

**use** **yii**\**db**\**ActiveRecord**;

**class** **Customer** **extends** **ActiveRecord**

{

/\*\*

\* **@return** string 返回该AR类关联的数据表名

\*/

**public** **static** **function** **tableName**()

{

**return** 'customer';

}

}

### 访问列数据

AR 把相应数据行的每一个字段映射为 AR 对象的一个个特性变量（Attribute） 一个特性就好像一个普通对象的公共属性一样（public property）。 特性变量的名称和对应字段的名称是一样的，且大小姓名。

使用以下语法读取列的值：

// "id" 和 "mail" 是 $customer 对象所关联的数据表的对应字段名

$id = $customer->id;

$email = $customer->email;

要改变列值，只要给关联属性赋新值并保存对象即可：

$customer->email = 'james@example.com';

$customer->save();

### 建立数据库连接

AR 用一个 [DB connection](http://www.yiichina.com/doc/api/2.0/yii-db-connection) 对象与数据库交换数据。 默认的，它使用 db 组件作为其连接对象。详见[数据库基础](http://www.yiichina.com/doc/guide/2.0/database-basics)章节， 你可以在应用程序配置文件中设置下 db 组件，就像这样，

**return** [

'components' => [

'db' => [

'class' => 'yii\db\Connection',

'dsn' => 'mysql:host=localhost;dbname=testdb',

'username' => 'demo',

'password' => 'demo',

],

],

];

如果在你的应用中应用了不止一个数据库，且你需要给你的 AR 类使用不同的数据库链接（DB connection） ，你可以覆盖掉 [getDb()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord" \l "getDb()-detail) 方法：

**class** **Customer** **extends** **ActiveRecord**

{

// ...

**public** **static** **function** **getDb**()

{

**return** \Yii::$app->db2; // 使用名为 "db2" 的应用组件

}

}

### 查询数据

AR 提供了两种方法来构建 DB 查询并向 AR 实例里填充数据：

* [yii\db\ActiveRecord::find()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord#find()-detail)
* [yii\db\ActiveRecord::findBySql()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord#findBySql()-detail)

以上两个方法都会返回 [yii\db\ActiveQuery](http://www.yiichina.com/doc/api/2.0/yii-db-activequery) 实例，该类继承自[yii\db\Query](http://www.yiichina.com/doc/api/2.0/yii-db-query)， 因此，他们都支持同一套灵活且强大的 DB 查询方法，如 where()，join()，orderBy()，等等。 下面的这些案例展示了一些可能的玩法：

// 取回所有活跃客户(状态为 \*active\* 的客户）并以他们的 ID 排序：

$customers = Customer::find()

->where(['status' => Customer::STATUS\_ACTIVE])

->orderBy('id')

->all();

// 返回ID为1的客户：

$customer = Customer::find()

->where(['id' => 1])

->one();

// 取回活跃客户的数量：

$count = Customer::find()

->where(['status' => Customer::STATUS\_ACTIVE])

->count();

// 以客户ID索引结果集：

$customers = Customer::find()->indexBy('id')->all();

// $customers 数组以 ID 为索引

// 用原生 SQL 语句检索客户：

$sql = 'SELECT \* FROM customer';

$customers = Customer::findBySql($sql)->all();

**提示：**在上面的代码中，Customer::STATUS\_ACTIVE 是一个在 Customer 类里定义的常量。（译注：这种常量的值一般都是tinyint）相较于直接在代码中写死字符串或数字，使用一个更有意义的常量名称是一种更好的编程习惯。

### 以数组形式获取数据

有时候，我们需要处理很大量的数据，这时可能需要用一个数组来存储取到的数据， 从而节省内存。你可以用 asArray() 函数做到这一点：

// 以数组而不是对象形式取回客户信息：

$customers = Customer::find()

->asArray()

->all();

// $customers 的每个元素都是键值对数组

### 批量获取数据

在 [Query Builder（查询构造器）](http://www.yiichina.com/doc/guide/2.0/query-builder) 里，我们已经解释了当需要从数据库中查询大量数据时，你可以用 batch query（批量查询）来限制内存的占用。 你可能也想在 AR 里使用相同的技巧，比如这样……

// 一次提取 10 个客户信息

**foreach** (Customer::find()->batch(10) **as** $customers) {

// $customers 是 10 个或更少的客户对象的数组

}

// 一次提取 10 个客户并一个一个地遍历处理

**foreach** (Customer::find()->each(10) **as** $customer) {

// $customer 是一个 ”Customer“ 对象

}

// 贪婪加载模式的批处理查询

**foreach** (Customer::find()->with('orders')->each() **as** $customer) {

}

### 操作数据

AR 提供以下方法插入、更新和删除与 AR 对象关联的那张表中的某一行：

* yii\db\ActiveRecord::save()
* [insert()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord#insert()-detail)
* [update()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord#update()-detail)
* [delete()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord#delete()-detail)

AR 同时提供了一下静态方法，可以应用在与某 AR 类所关联的整张表上。 用这些方法的时候千万要小心，因为他们作用于整张表！ 比如，deleteAll() 会删除掉表里**所有**的记录。

* yii\db\ActiveRecord::updateCounters()
* [updateAll()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord#updateAll()-detail)
* [updateAllCounters()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord#updateAllCounters()-detail)
* [deleteAll()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord#deleteAll()-detail)

下面的这些例子里，详细展现了如何使用这些方法：

// 插入新客户的记录

$customer = **new** Customer();

$customer->name = 'James';

$customer->email = 'james@example.com';

$customer->save(); // 等同于 $customer->insert();

// 更新现有客户记录

$customer = Customer::findOne($id);

$customer->email = 'james@example.com';

$customer->save(); // 等同于 $customer->update();

// 删除已有客户记录

$customer = Customer::findOne($id);

$customer->delete();

// 删除多个年龄大于20，性别为男（Male）的客户记录

Customer::deleteAll('age > :age AND gender = :gender', [':age' => 20, ':gender' => 'M']);

// 所有客户的age（年龄）字段加1：

Customer::updateAllCounters(['age' => 1]);

须知：save() 方法会调用 insert() 和 update() 中的一个， 用哪个取决于当前 AR 对象是不是新对象（在函数内部，他会检查 yii\db\ActiveRecord::isNewRecord 的值）。 若 AR 对象是由 new 操作符 初始化出来的，save() 方法会在表里插入一条数据； 如果一个 AR 是由 find() 方法获取来的， 则 save() 会更新表里的对应行记录。

### 数据输入与有效性验证

由于AR继承自[yii\base\Model](http://www.yiichina.com/doc/api/2.0/yii-base-model)，所以它同样也支持[Model](http://www.yiichina.com/doc/guide/2.0/model)的数据输入、验证等特性。例如，你可以声明一个rules方法用来覆盖掉[rules()](http://www.yiichina.com/doc/api/2.0/yii-base-model" \l "rules()-detail)里的；你也可以给AR实例批量赋值；你也可以通过调用[validate()](http://www.yiichina.com/doc/api/2.0/yii-base-model" \l "validate()-detail)执行数据验证。

当你调用 save()、insert()、update() 这三个方法时，会自动调用[validate()](http://www.yiichina.com/doc/api/2.0/yii-base-model#validate()-detail)方法。如果验证失败，数据将不会保存进数据库。

下面的例子演示了如何使用AR 获取/验证用户输入的数据并将他们保存进数据库：

// 新建一条记录

$model = **new** Customer;

**if** ($model->load(Yii::$app->request->post()) && $model->save()) {

// 获取用户输入的数据，验证并保存

}

// 更新主键为$id的AR

$model = Customer::findOne($id);

**if** ($model === **null**) {

**throw** **new** NotFoundHttpException;

}

**if** ($model->load(Yii::$app->request->post()) && $model->save()) {

// 获取用户输入的数据，验证并保存

}

### 读取默认值

你的表列也许定义了默认值。有时候，你可能需要在使用web表单的时候给AR预设一些值。如果你需要这样做，可以在显示表单内容前通过调用loadDefaultValues()方法来实现：

$customer = **new** Customer();

$customer->loadDefaultValues();

// ... 渲染 $customer 的 HTML 表单 ...

### AR的生命周期

理解AR的生命周期对于你操作数据库非常重要。生命周期通常都会有些典型的事件存在。对于开发AR的behaviors来说非常有用。

实例化生命周期（New Instance Life Cycle）：

1. 类的构造函数调用
2. yii\db\ActiveRecord::init(): 会触发一个 yii\db\ActiveRecord::EVENT\_INIT 事件

查询数据生命周期（Querying Data Life Cycle）：

1. constructor
2. yii\db\ActiveRecord::init(): 会触发一个 yii\db\ActiveRecord::EVENT\_INIT 事件
3. yii\db\ActiveRecord::afterFind(): 会触发一个 yii\db\ActiveRecord::EVENT\_AFTER\_FIND 事件

保存数据生命周期（Saving Data Life Cycle）：

1. yii\db\ActiveRecord::beforeValidate(): 会触发一个 yii\db\ActiveRecord::EVENT\_BEFORE\_VALIDATE 事件
2. yii\db\ActiveRecord::afterValidate(): 会触发一个 yii\db\ActiveRecord::EVENT\_AFTER\_VALIDATE 事件
3. yii\db\ActiveRecord::beforeSave(): 会触发一个 yii\db\ActiveRecord::EVENT\_BEFORE\_INSERT 或 yii\db\ActiveRecord::EVENT\_BEFORE\_UPDATE 事件
4. 执行实际的数据写入或更新
5. yii\db\ActiveRecord::afterSave(): 会触发一个 yii\db\ActiveRecord::EVENT\_AFTER\_INSERT 或 yii\db\ActiveRecord::EVENT\_AFTER\_UPDATE 事件

删除数据生命周期（Deleting Data Life Cycle）：

1. yii\db\ActiveRecord::beforeDelete(): 会触发一个 yii\db\ActiveRecord::EVENT\_BEFORE\_DELETE 事件
2. 执行实际的数据删除
3. yii\db\ActiveRecord::afterDelete(): 会触发一个 yii\db\ActiveRecord::EVENT\_AFTER\_DELETE 事件

**Tip：**调用以下方法则不会启动上述的任何生命周期， 因为这些方法直接操作数据库，而不是基于 Active Record 模型：

* [yii\db\ActiveRecord::updateAll()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord#updateAll()-detail)
* [yii\db\ActiveRecord::deleteAll()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord#deleteAll()-detail)
* yii\db\ActiveRecord::updateCounters()
* [yii\db\ActiveRecord::updateAllCounters()](http://www.yiichina.com/doc/api/2.0/yii-db-activerecord#updateAllCounters()-detail)

刷新数据生命周期（Refreshing Data Life Cycle）：

当通过 yii\db\ActiveRecord::refresh() 刷新 Active Record 实例时， 如刷新成功方法返回 true，那么 yii\db\ActiveRecord::EVENT\_AFTER\_REFRESH 事件将被触发。