# 简介

模型是 [MVC](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) 模式中的一部分， 是代表业务数据、规则和逻辑的对象。

可通过继承 [yii\base\Model](https://www.yiichina.com/doc/api/2.0/yii-base-model) 或它的子类定义模型类， 基类[yii\base\Model](https://www.yiichina.com/doc/api/2.0/yii-base-model)支持许多实用的特性：

* [属性](https://www.yiichina.com/doc/guide/2.0/structure-models" \l "attributes): 代表可像普通类属性或数组 一样被访问的业务数据;
* [属性标签](https://www.yiichina.com/doc/guide/2.0/structure-models" \l "attribute-labels): 指定属性显示出来的标签;
* [块赋值](https://www.yiichina.com/doc/guide/2.0/structure-models" \l "massive-assignment): 支持一步给许多属性赋值;——注意，在更新表数据的时候块赋值别乱用，以免被篡改数据
* [验证规则](https://www.yiichina.com/doc/guide/2.0/structure-models" \l "validation-rules): 确保输入数据符合所申明的验证规则;
* [数据导出](https://www.yiichina.com/doc/guide/2.0/structure-models" \l "data-exporting): 允许模型数据导出为自定义格式的数组。

# 属性

模型通过 属性 来代表业务数据，每个属性像是模型的公有可访问属性，

[yii\base\Model::attributes()](https://www.yiichina.com/doc/api/2.0/yii-base-model" \l "attributes()-detail) 指定模型所拥有的属性。



## 属性标签

当属性显示或获取输入时，经常要显示属性相关标签， 例如假定一个属性名为firstName， 在某些地方如表单输入或错误信息处，你可能想显示对终端用户来说更友好的 First Name 标签。

可以调用 [yii\base\Model::getAttributeLabel()](https://www.yiichina.com/doc/api/2.0/yii-base-model" \l "getAttributeLabel()-detail) 获取属性的标签，例如：

$model = **new** \app\models\ContactForm;

// 显示为 "Name"

**echo** $model->getAttributeLabel('name');

可以覆盖 [yii\base\Model::attributeLabels()](https://www.yiichina.com/doc/api/2.0/yii-base-model" \l "attributeLabels()-detail) 方法明确指定属性标签， 例如：



# 场景

Yii的场景主要是应用于不同场景属性的 块赋值和验证规则。

个人认为通过为每一个场景增加一个继承于表模型的场景模型会更灵活和便于维护。

如下：



# 验证规则

当模型接收到终端用户输入的数据， 数据应当满足某种规则(称为 验证规则, 也称为 业务规则)。

可调用 [yii\base\Model::validate()](https://www.yiichina.com/doc/api/2.0/yii-base-model" \l "validate()-detail) 来验证接收到的数据， 该方法使用[yii\base\Model::rules()](https://www.yiichina.com/doc/api/2.0/yii-base-model" \l "rules()-detail)申明的验证规则来验证每个相关属性， 如果没有找到错误，会返回 true， 否则它会将错误保存在 [yii\base\Model::$errors](https://www.yiichina.com/doc/api/2.0/yii-base-model" \l "$errors-detail) 属性中并返回false，例如：



下述例子显示ContactForm模型申明的验证规则:



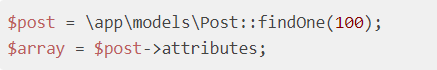
# 数据导出

模型通常要导出成不同格式，例如，你可能想将模型的一个集合转成JSON或Excel格式， 导出过程可分解为两个步骤：

* 模型转换成数组；
* 数组转换成所需要的格式。

你只需要关注第一步，因为第二步可被通用的 数据转换器如[yii\web\JsonResponseFormatter](https://www.yiichina.com/doc/api/2.0/yii-web-jsonresponseformatter)来完成。

将模型转换为数组最简单的方式是使用 [yii\base\Model::attributes()](https://www.yiichina.com/doc/api/2.0/yii-base-model" \l "attributes()-detail) 属性， 例如：



## 字段



# 最佳实践

为确保模型好维护，最好使用以下策略：

* 定义可被多个 [应用主体](https://www.yiichina.com/doc/guide/2.0/structure-applications) 或 [模块](https://www.yiichina.com/doc/guide/2.0/structure-modules) 共享的模型基类集合。 这些模型类应包含通用的最小规则集合和逻辑。
* 在每个使用模型的 [应用主体](https://www.yiichina.com/doc/guide/2.0/structure-applications) 或 [模块](https://www.yiichina.com/doc/guide/2.0/structure-modules)中， 通过继承对应的模型基类来定义具体的模型类， 具体模型类包含应用主体或模块指定的规则和逻辑。

例如，在[高级应用模板](https://www.yiichina.com/doc/guide/2.0/tutorial-advanced-app)， 你可以定义一个模型基类common\models\Post， 然后在前台应用中，定义并使用一个继承common\models\Post的具体模型类frontend\models\Post， 在后台应用中可以类似地定义backend\models\Post。 通过这种策略，你清楚frontend\models\Post只对应前台应用，如果你修改它， 就无需担忧修改会影响后台应用。