18．模型获取器和修改器

学习要点：

1.模型获取器

2.模型修改器

本节课我们来学习模型中操作比较方便的获取器和修改器。

一．模型获取器

1. 获取器的作用是对模型实例的数据做出自动处理；
2. 一个获取器对应模型的一个特殊方法，该方法为 public；
3. 方法名的命名规范为：getFieldAttr()；
4. 举个例子，数据库表示状态 status 字段采用的是数值；
5. 而页面上，我们需要输出 status 字段希望是中文，就可以使用获取器；
6. 在 User 模型端，我创建一个对外的方法，如下：

```
public function getStatusAttr($value)
{
    $status = [-1=>'删除', 0=>'禁用', 1=>'正常', 2=>'待审核'];
    return $status[$value];
}
```

7. 然后，在控制器端，直接输出数据库字段的值即可得到获取器转换的对应值；

```
$user = UserModel::get(21);
return $user->status;
```

8. 除了 getFieldAttr 中 Field 可以是字段值，也可以是自定义的虚拟字段；

```
public function getNothingAttr($value, $data)
{
    $myGet = [-1=>'删除', 0=>'禁用', 1=>'正常', 2=>'待审核'];
    return $myGet[$data['status']];
}

return $user->nothing;
```

9. Nothing 这个字段不存在，而此时参数$value 只是为了占位，并未使用；
10. 第二个参数$data 得到的是筛选到的数据，然后得到最终值；
11. 如果你定义了获取器，并且想获取原始值，可以使用 getData()方法；

```
return $user->getData('status');
```

12. 直接输出无参数的 getData()，可以得到原始值，而$user 输出是改变后的；

```
dump($user->getData());
dump($user);
```

13. 使用 WithAttr 在控制器端实现动态获取器，比如设置所有 email 为大写；

```
$result = UserModel::WithAttr('email', function ($value) {
    return strtoupper($value);
})->select();
return json($result);
```

14. 使用 WithAttr 在控制器端实现动态获取器，比如设置 status 翻译为中文；

```
$result = UserModel::WithAttr('status', function ($value) {
    $status = [-1=>'删除', 0=>'禁用', 1=>'正常', 2=>'待审核'];
    return $status[$value];
})->select();
return json($result);
```

15. 同时定义了模型获取器和动态获取器，那么模型修改器优先级更高；

二. 模型修改器

1. 模型修改器的作用，就是对模型设置对象的值进行处理；
2. 比如，我们要新增数据的时候，对数据就行格式化、过滤、转换等处理；
3. 模型修改器的命名规则为：setFieldAttr；
4. 我们要设置一个新增，规定邮箱的英文都必须大写，修改器如下：

```
public function setEmailAttr($value)
{
    return strtoupper($value);
}
```

5. 除了新增，会调用修改器，修改更新也会触发修改器；
6. 模型修改器只对模型方法有效，调用数据库的方法是无效的，比如->insert();