# 19．模型搜索器和数据集

学习要点：

1.模型搜索器

2.模型数据集

本节课我们来学习模型中的用于封装的搜索器和数据结果集的操作。

## 一．模型搜索器

1. 搜索器是用于封装字段（或搜索标识）的查询表达式；
2. 一个搜索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：searchFieldNameAttr()；
4. 举个例子，我们要封装一个邮箱字符模糊查询，然后封装一个时间限定查询；
5. 在 User 模型端，我创建两个对外的方法，如下：

```php
public function searchEmailAttr($query, $value)
{
    $query->where('email', 'like', $value.'%');
}

public function searchCreateTimeAttr($query, $value)
{
    $query->whereBetweenTime('create_time', $value[0], $value[1]);
}
```

6. 然后，在控制器端，通过 withSearch()静态方法实现模型搜索器的调用；

```php
$result = UserModel::withSearch(['email', 'create_time'],[
    'email'         =>  'xiao',
    'create_time'    =>  ['2014-1-1', '2017-1-1']
])->select();
```

7. withSearch()中第一个数组参数，限定搜索器的字段，第二个则是表达式值；
8. 如果想在搜索器查询的基础上再增加查询条件，直接使用链式查询即可；

```php
UserModel::withSearch(...)->where('gender', '女')->select()
```

9. 如果你想在搜索器添加一个可以排序的功能，具体如下：

```php
public function searchEmailAttr($query, $value, $data)
{
    $query->where('email', 'like', $value.'%');
    if (isset($data['sort'])) {
        $query->order($data['sort']);
    }
}
```

```
$result = UserModel::withSearch(['email', 'create_time'],[
    'email'         =>  'xiao',
    'create_time'   =>  ['2014-1-1', '2017-1-1'],
    'sort'          =>  ['price'=>'desc']
])->select();
```

10. 搜索器的第三个参数$data，可以得到 withSearch()方法第二参数的值；
11. 字段也可以设置别名：'create_time'=>'ctime'

## 二. 模型数据集

1. 数据集由 all()和 select()方法返回数据集对象；
2. 数据集对象和数组操作方法一样，循环遍历、删除元素等；
3. 判断数据集是否为空，我们需要采用 isEmpty()方法；

```
$resut = UserModel::where('id', 111)->select();
if ($resut->isEmpty()) {
    return '没有数据！';
}
```

4. 使用模型方法 hidden()可以隐藏某个字段，使用 visible()显示只某个字段；
5. 使用 append()可以添加某个获取器字段，使用 withAttr()对字段进行函数处理；

```
$result = UserModel::select();
$result->hidden(['password'])->append(['nothing'])->withAttr('em
ail', function ($value) {
    return strtoupper($value);
});
return json($result);
```

6. 使用模型方法 filter()对筛选的数据进行过滤；

```
$result = UserModel::select()->filter(function ($data) {
    return $data['price'] > 100;
});
return json($result);
```

7. 也可以使用数据集之后链接 where()方法来代替 filter()方法；

```
$result = UserModel::select()->where('price', '>', '100');
```

8. 数据集甚至还可以使用 order()方法进行排序；

```
$result = UserModel::select()->order('price', 'desc');
```

9. 使用 diff()和 intersect()方法可以计算两个数据集的差集和交集；

```
$result1 = UserModel::where('price', '>', '80')->select();
$result2 = UserModel::where('price', '<', '100')->select();
```

```
return json($result1->diff($result2));
return json($result2->intersect($result1));
```