Model类

WHERE条件查询

- where子句的例子。

```
// 1. WHERE `id` = 1;
$student->where('id',1)->get();
// 2. WHERE `id` > 1;
$student->where('id','>',1)->get();
// 3. WHERE `id` = 1 AND `name`=a;
$student->where(['id'=>1,'name'=>'a'])->get();
$student->where('id',1)->where('name','a')->get();
// 4. WHERE `id` = 1 OR `name`=a;
$student->where('id',1)->orWhere('name','a')->get();
5. WHERE `id` > 1 ORDER BY `id` DESC LIMIT 1,2;
$student->where('id','>',1)->orderBy('id','DESC')->limit(1,2)->get();
```

- 目的：根据这些例子，写出一些较为通用的函数。
    - 输入：
        - 分析：
            - 字符串、运算符、数字。
                - 参数的个数：
                    - 可能2个。(默认的意思是运算符为等于)
                        - ‘表的字段名’，‘值’。
                    - 可能3个。
                        - ‘表的字段名’，‘运算符’，‘值’。
                - 参数的顺序：
                    - 经过进一步分析，发现，参数的顺序依次为：‘表的字段名’，（运算符，）值。
                    - 当然也可以这么写：‘表的字段名’，‘值’，（‘运算符’）。因为调用方法时必须依次传入参数，使用频率的参数尽可能放在后面。但是这样写不符合我们的习惯。即把运算符写在最后面。
            - 数组。
                - 数组的形式：表的字段对应值。
        - 确定参数个数：
            - 3个。
                - 原则是只能多，不能少。
                    - 只有两个形参，却想传入三个实参肯定不行。
                    - 有三个形参，可以传入两个实参，因为可以为形参设置默认值。
                        - ($field,$op='=',$value=null)

- 输出：
  - 拼接出完整的where子句。
    - 是一个较为独立的功能，所以可以用函数来封装这个功能。
      - 判断传入的实参$field是否为数组。
        - 如果是：循环遍历，为每对键值对拼接出完整的where子句。即：递归。

```
if (is_array($field)) {
    foreach ($field as $k => $v) {
        $this->buildWhere($k, $op, $v, $join);
    }
    return ;
    <!-- 必须要有return,但没有返回值。 只起终止程序的作用。-->
}
```

        - 如果不是：
          - 而且只传了两个实参，即$value为默认值null。
            - 将传入的$op的值赋给$value，
            - 并且给$op赋值，值为'=';

```
elseif (is_null($value)) {
  $value = $op;
  $op = '=';
}
```

      - 判断是否为第一次使用where条件。即$this->options['where']是否为空。
        - 如果是：
          - $join = 'WHERE';
        - 否则：
          - $join的值就为传入的值。

```
if (empty($this->options['where'])) {
  $join = 'WHERE';
}
```

      - 最后拼接where子句。
  - $this;
    - 这个好理解，因为要实现连续操作，肯定要把这个实例传出去。
- 完整代码。

```
public function where($field, $op='=', $value = null)
{
    $this->buildWhere($field, $op, $value, 'AND');
    return $this;
}
public function orWhere($field, $op='=', $value = null)
{
    $this->buildWhere($field, $op, $value, 'OR');
    return $this;
}
public function buildWhere($field, $op, $value, $join = 'AND')
{
    if (is_array($field)) {
        foreach ($field as $k => $v) {
            $this->buildWhere($k, $op, $v, $join);
        }
        return ;
    } elseif (is_null($value)) {
        $value = $op;
        $op = '=';
    }
    if (empty($this->options['where'])) {
        $join = 'WHERE';
    }
    $this->options['where'] .= "$join $field $op ?";
    $this->options['data'][] = $value;
}
```

insert方法

- 示例
  - 添加一条数据：
    - ['name' => '小红','gender' => 0];
    - insert into student(name,gender) values(?,?)
  - 添加多条数据：

    [
    ['name' => '小红','gender' => 0],
    ['name' => '小明','gender' => 1]
    ]
    insert into student(name,gender) values(?,?),(?,?)
- 输入：数组。
- 过程：
  - 分流：
    - 判断输入的数组是一维的还是多维的。
  - 准备SQL语句。

        - 单独的功能，用方法封装这个功能。
            - 输入：字段数组，多维数组元素的个数。默认值为1；（如果是一维数组就不需要传这个参数）
            - 步骤：
                - 根据字段的个数生成'?'的个数。
                - 拼接成(?,...,?);
                - 根据多维数组元素的个数生成(?,...,?),...,(?,...,?);
                - 拼接字段数组成字符串。
            - 输出：SQL语句。
                - 即：
                    - insert into student(name,gender) values(?,?)
                    - insert into student(name,gender) values(?,?),(?,?)
    - 执行SQL语句。
    - 清空条件。
- 输出：受影响行。