ShopNC B2B2C 商城开发手册

天津市网城天创科技有限责任公司

2015 年 1 月

1. 简介

ShopNC B2B2C 商城是 ShopNC 于 2014 年最新推出的【B2B2C】电商平台系统，采用 PHP5+MySQL 技术为基础，采用 OOP（面向对象）方式进行核心框架搭建，结合 MVC 模式进行开发，可以支持 Windows/Unix 服务器环境，需 PHP5.3 及以上版本支持，可运行于包括 Apache、IIS 和 Nginx 在内的多种 WEB 服务器。

2. 架构介绍

2.1 系统特性

ShopNC 借鉴了国内外优秀的开源程序与开发模式，使用面向对象的开发架构、MVC 模式开发，主要特性如下。

- **MVC 设计**

ShopNC B2B2C 商城融合了 MVC 模式进行开发，系统开发高效，各个节点结构更加清晰。

模型(M)：模型的定义由 Model 类来完成。

控制器(C)：由框架核心和 Action 共同完成。

视图(V)：由 Tpl 类和模板文件组成。

MVC 作为一种模式只是提供了一种敏捷开发的手段，ShopNC 系统融入 MVC 模式但不拘泥于 MVC 本身。

- **缓存机制**

B2B2C 商城支持使用 Redis 存储数据，使用响应速度提高。

- **查询机制**

系统内建丰富的查询机制，包括单表查询、多表查询、区间查询、统计查询、定位查询和原生查询等，使用数据查询简洁高效。

- **动态模型**

系统中无需创建对应的模型类、即可轻松完成 CURD 操作，使数据库操作更加简洁。

- **搜索机制**

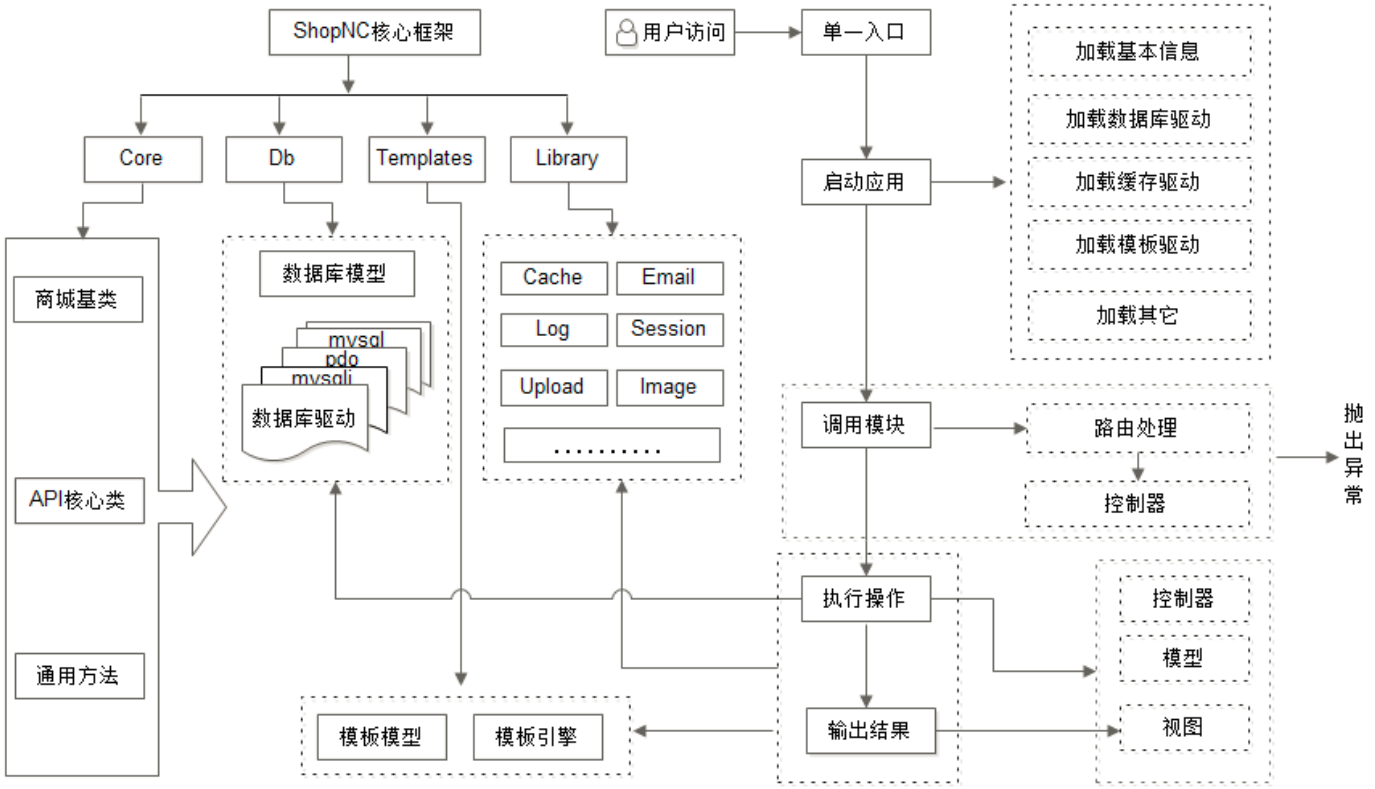
系统引入第三方搜索引擎，最大化减少搜索给系统带来的影响。

- **易用性**

LAMP 架构保证跨平台、MVC 模块化开发保证易维护与扩展、系统架构保证集群部署与扩展。

2.2 执行流程

系统采用单一入口模式，系统框架执行流程如下：



2.3 目录结构

系统主要目录：

admin	后台管理目录
chat	IM 目录
circle	圈子目录
cms	CMS 目录
microshop	微商城目录
shop	商城目录
core	框架目录
data	公共资源目录
mobile	手机客户端 API
wap	wap 商城
crontab	定时任务执行目录

delivery 自提服务站后台管理

2.4 MVC 设计

ShopNC B2B2C 商城融入 MVC 模式进行开发，系统开发高效，各个节点结构更加清晰。

模型（ M ）：模型的定义由 Model 类来完成。

控制器（ C ）：由框架核心和 Action 共同完成。

视图（ V ）：由 Tpl 类和模板文件组成。

MVC 作为一种模式只是提供了一种敏捷开发的手段，ShopNC 系统应用 MVC 但不拘泥于 MVC 本身。

2.5 控制器

ShopNC B2B2C 商城控制器类位于 control 目录，控制器调度由框架依据 act 和 op 参数完成，如果 act 或 op 参数为空，系统会自动赋值“index”。

控制器类文件名一般为业务名称，类名称一般为 业务名称 + “Control”，如品牌控制器类文件命名为 control/brand.php，类名为 brandControl。

http://<siteurl>/index.php

将会执行 control/index.php 中的 indexOp()方法

http://<siteurl>/index.php?act=brand&op=list

将会执行 control/brand.php 中的 listOp()方法

2.6 模型

ShopNC B2B2C 商城 Model 的使用灵活，可以无需进行任何模型定义，就可以完成对相关数据表的 CURD 操作，这就是动态模型处理，不需要重复进行模型实例化即可实现对不同表的操作。新模型的处理支持向下兼容。

使用 Model()方法创建模型，可以创建一个空模型实例，然后使用该实例完成对不同表的操作，如：

```
$model = Model();
```

也可以实例化一个非空模型实例，如：

```
$model = Model('member');
```

系统首先会查找 model/member.model.php 文件及内部的 memberModel 类是否存在：

a) 如果存在 将实例化 member.model.php 中的 memberModel 类 如果需使用框架已封装的方法(select、find、delete、insert 等)，需要在 memberModel 类中继承 Model 类并在构造方法中触发父类构造方法，

```
class memberModel extends Model{
```

```
    public function __construct(){
            parent::__construct('member');
    }
    //除使用系统提供方法以外，还可以自定义方法
    //public function myfuc(){
        //添加业务逻辑
    }
```

b) 如果不存在，将实例化 framework/core/model.php 中的 Model 类，也就是只实例化框架提供的模型类（动态模型）

```
$model = Model('member');
```

2.7 数据库抽象层

ShopNC B2B2C 商城支持使用 mysql、mysqli 访问 MySQL 数据库，支持主从架构部署。主从配置请详见安装手册。

2.8 视图

视图功能主要由 Tpl 类和模板文件组成（位于 templates 目录下），Tpl 类完成控制器和模板文件的沟通，控制器通过 Tpl 类将数据输送到模板，然后由模板输出数据，ShopNC B2B2C 商城未使用特定的模板语言，而是使用原生的 PHP 语法，省去了解析模板语言的时间，加快响应速度。

3. 开发指南

3.1 控制器

系统控制器类位于 control 目录，控制器调度由 framework/core/base.php 中 control()方法依据 act 和 op 参数完成，如果 act 或 op 参数为空，系统会自动赋值“index”。

控制器类文件名一般为业务名称，类名称一般为 业务名称 + “Control”，例如系统的品牌控制器类文件为 control/brand.php，类名为 brandControl。

跟据商城业务需要，系统内置多个控制器父级类，分别适用于前台展示、下单、会员中心、商家中心等控制器，品牌展示需要继承 BaseHomeControl 类。

```
<?php
```

```php
/**
* 品牌展示
*
* 版权声明...
*/
defined('InShopNC') or exit('Access Invalid!');
class brandControl extends BaseHomeControl {
    public function indexOp(){
        //读取语言包
        Language::read('home_brand_index');
        //使用模型获得品牌列表
        $model_brand = Model('brand');
        $brand_list = $model_brand->getBrandList();
        //向模板抛出内容
        Tpl::output('brand_list',$brand_list);
        //设置页面标题
        Tpl::output('html_title',Language::get('brand_index_brand_list'));
        //输出 SEO 设置信息
        Model('seo')->type('brand')->show();
        //调用模板展示
        Tpl::showpage('brand');
    }
    public function searchOp(){
        /**
        * 内容略...
        */
    }
}
?>
```

访问 http://<siteurl>/index.php?act=brand

将会执行 brandControl 类的 indexOp 方法

访问 http://<siteurl>/index.php?act=brand&op=search

将会执行 brandControl 类的 searchOp 方法

3.2 模型

3.2.1 实例化

使用 Model()方法创建模型，可以创建一个空模型实例，然后全用该实例完成对不同表的操作，如：

```
$model = Model();
```

也可以实例化一个非空模型实例，如：

```
$model = Model('member');
```

系统首先会查找 model/member.model.php 文件及内部的 memberModel 类是否存在，实例化该类，如果不存在，则实例化框架提供的模型类，实例化模型更详细的信息可查看 2.6 模型

3.2.2 内置方法

系统模型提供了一系列快捷操作的方法 ,可以大幅提高开发效率。目前已提供的方法主要有 select、find、limit、table、order、where、field、on、join、count、page、showpage、insert、insertAll、delete、update、group、clear、query、execute、sum、avg、max、min、setInc、setDec、和动态方法 getby_、getfby_。

1. **Select** 方法：取得查询信息，返回结果为数组，如果未找到数据返回 null，select 一般在 where,order,tabale 等方法的后面，作为最后一个方法来使用。如:

```
$model = Model('member');
// 查询会员表全部信息
$model->select();
//取得性别为 1 的会员列表信息, 注意：select 方法需要在连贯操作中的最后一步出现
$model->where(array('member_sex'=>1))->select();
```

2. **Find** 方法：取得一条记录信息,find 同 select 一样，一般作为最后一个方法来使用，如：

```
$model = Model('member');
// 查询 ID 为 5 的会员信息
$model->where(array('member_id'=>5))->find();
```

3. **Limit** 方法：指定返回多少条记录数，

```
$model = Model('member');
$model->limit(4)->select();        // 等同于 SELECT * FROM member LIMIT 4;
$model->limit('4,10')->select();   // 等同于 SELECT * FROM member LIMIT 4,10;
```

4. **Table** 方法：指定要操作的数据表名称，返回模型实例本身，如：

```
$model = Model();
```

多表联合查询时，可以传入多个表名称，如：

```
// 内链接查询 member 和 store 表,并返回前两条记录
$on = 'store.member_id=member.member_id';
$model->table('member,store')->join('inner')->on($on)->limit(2)->select();
```

如果实例化时指定了表名，则可以不使用 table 方法指定表名，如：

```
$model = Model('member');
$model ->limit(4)->select();  // 查询前条 4 会员记录
```

5. **Order** 方法：指定排序的参数，返回模型实例本身，如：

```
$model->table('member')->order('member_id desc')->limit(4)->select();
```

也可指定多个字段进行排序，如：

```
$model->table('member')->order('member_id desc,member_sex asc')->select();
```

6. **Where** 方法：指定 sql 执行的条件，返回模型实例本身，入可传入数组或字段串，如：

```
//传入数组条件
$model->where(array('member_id'=>5))->find();
//传入多表关联条件
$model->table('member,store');
$model->where('store.store_id=member.store_id and store.store_id=2')->find();
```

7. **Field** 方法：指定要查询的字段，不使用 field 方法时，默认查询所有字段，如：

```
$model->field('member_id,member_name')->select();
```

8. **On** 方法：指定多表联查时的字段关系。
9. **Join** 方法：指定多表联查时的链接类型， 支持内链接、左链接(默认)、右链接。On 与 join 方法需要一起使用，如：

```
$model = Model();
//内链接查询 member 和 store 表,返回会员 ID 为 6 的记录信息
$field = 'member.member_name,store.store_name';
$on = 'store.member_id=member.member_id';
$model->table('member,store')->field($field);
$model->join('inner')->on($on)->where(array('member.member_id'=>6))->find();
```

三表关联查询如下：

```
$model = Model();
//内链接查询 member 和 store,然后左链接 store_class,查询会员 ID 为 6 的记录信息
$field = 'member.member_name,store.store_name,store_class.sc_name';
$on = 'store.member_id=member.member_id,store.sc_id=store_class.sc_id';
```

```
$model->table('member,store,store_class')->field($field);
$model->join('inner,left')->on($on)->where('member.member_id=6')->find();
```

10. Count 方法：返回记录总数量，如：

```
$model = Model('member');
//返回会员表总行数
$model->count();
$model->where(array('member_sex'=>0))->count();
```

11. Page 方法：实现记录分页，格式为 page(每页显示数，总记录数)，总记录数可以人为指定，也可以为空让系统自动去计算，如：

```
//每页显示 10 条数据
 $model = Model('member');
 //系统会跟据每页显示数和已知属性自动计算总记录数
 $model->page(10)->order('member_id desc')->select();
 //每页显示 10 条数据，指定总记录为 1000 条，系统将不再计算总记录数
$model->page(10, 1000)->order('member_id desc')->select();
```

注意：如果同时使用 where 和 page 方法时，where 方法要在 page 方法前面使用，如：

```
$model->where(array('member_id'=>5))->page(10)->select();   //正确
$model->page(10)->where(array('member_id'=>5))->select();   //错误
```

12. Showpage 方法：返回分页超链接，结合 page 方法完成分页，如：

```
//显示上一页下一下链接
$model->showpage();
```

13. Insert 方法：插入单行数据，并返回最新插入的主键 ID，如果插入失败返回 false,如：

```
//向 link 表插入数据，并返回最新主键 ID
$model = Model('table');
$data = array(
  'link_title'=>'ShopNC',
  'link_url'=>'http://www.shopnc.net',
 'link_sort'=>32,
 );
$model->insert($data);
```

Insert 方法同样支持 replace 操作，将第二个参数设置为 true 即可，如：

```
$model = Model();
$data = array(
```

```
    'link_title'=>'ShopNC',
    'link_url'=>'http://www.shopnc.net',
    'link_sort'=>32,
    'link_id'=>30
    );
    $model->table('link')->insert($data,true);
```

14. InsertAll 方法：实现批量插入数据，如：

```
    $model = Model('link');
    $data = array(
     array(
            'link_title'=>'新浪',
            'link_url'=>'http://www.sina.com',
           'link_sort'=>32,
     ),
     array(
           'link_title'=>'百度',
           'link_url'=>'http://www.baidu.com',
           'link_sort'=>30,
      )
     );
    $model->insertAll($data);
```

15. Delete 方法：删除记录，如：

```
    $model = Model('link');
    //删除主键为 5 的记录
    $model->where(array('link_id'=>5))->delete();
```

16. Update 方法：数据更新，如果更新内容含有主键下标，自动以该主键为更新条件，如：

```
    $model = Model();
    //更新主键(link_id)为 37 的记录信息
    $data = array(
      'link_title'=>'ShopNC',
      'link_url'=>'http://www.shponc.net',
      'link_sort'=>32,
     'link_id'=>37
```

```
);
$model->table('link')->update($data);
//指定更新条件
$data = array(
 'link_title'=>'ShopNC',
 'link_url'=>'http://www.shponc.net',
 'link_sort'=>32
);
$model->table('link')->where(array('link_id'=>37))->update($data);
```

17. Group 方法：实现分组功能，如：

```
//查询每个店铺发布商品的数量
$model = Model('goods');
$model->field('store_id,count(*) as count')->group('store_id')->select();
```

18. Clear 方法：清空单个表中的内容，返回 true/false，如：

```
//清空 link 表
$model = Model();
$model->table('link')->clear();
```

19. Query/execute 方法，两个方法均用于直接执行 SQL 语句，query 方法用于查询，execute 方法用于更新、写入和删除操作，如：

```
Model()->query('SELECT * FROM `shopnc_member` LIMIT 10');
Model()->execute('UPDATE `shopnc_goods` SET goods_click=1000 WHERE goods_id=2');
```

20. Sum/Avg/Max/Min 方法：求和、求平均值、取最大值、取最小值，如：

```
$model = Model();
//返回所有商品总价格之和
$model->table('goods')->sum('price');
//上面等同于 SQL：SELECT SUM(price) AS nc_sum FROM `goods`

//取商品表中所有商品的平均价格
$model->table('goods')->avg('price');
//以上等同于 SQL：SELECT AVG(price) AS nc_avg FROM `goods` LIMIT 1

//取商品的最高价
$model->table('goods')->max('price');
```

```
//以上等同于 SQL：SELECT MAX(price) AS nc_max FROM `goods` LIMIT 1

//取商品的最低价
$model->table('goods')->min('price');
//以上等同于 SQL：SELECT MIN(price) AS nc_min FROM `goods` LIMIT 1
```

21. 自增/自减：系统使用 setInc 和 setDec 完成自增和自减，示例如下：

```
$model = Model();
//使主键值为 2 的商品点击量加 1000
$model->table('goods')->where(array('goods_id'=>2))->setInc('goods_click',1000);
//等同于：UPDATE `goods` SET goods_click=goods_click+3 WHERE ( goods_id = '2' )

//结合 exp 参数，使用该商品点击量减 1000
$model = Model('goods');
$data = array(
'goods_id' => 2,
'goods_click' =>array('exp','goods_click-1000'));
$model->update($data);
//等同于：UPDATE `goods` SET goods_click=goods_click-1000 WHERE ( goods_id = '2' )
```

22. 动态方法：系统内置 getby_和 getfby_两个动态方法，格式如下：

getby_ + 字段名（字段值）

getfby_ + 条件字段名（条件字段值，返回字段名）

结合示例来说明动态方法的使用

```
$model = Model('member');
//使用 getby_动态方法，取得 member_name 为 kevin 的会员信息
$model->getby_member_name('kevin');
//等同于 SQL：SELECT * FROM `member` WHERE ( member_name = 'kevin' ) LIMIT 1

//使用 getfby_方法，取得 member_id 为 6 的会员名
$a = $model->getfby_member_id(6,'member_name');    //返回 kevin
//等同于 SQL：SELECT member_name FROM `shopnc_member` WHERE ( member_id = '6' ) LIMIT 1
```

23. 设置 SQL 执行优先级：系统支持使用 SQL 关键字 LOW_PRIORITY、DELAYED、HIGH_PRIORITY，格式如下：

attr（关键字）

结合示例来说明动态方法的使用

```
$model = Model('goods');
$model->where(array('goods_id' => 100))->attr('LOW_PRIORITY');
$model->update(array('goods_click' =>array('exp','goods_click+1')));
//等同于
//UPDATE LOW_PRIORITY `shopnc_goods` SET goods_click=goods_click+1 WHERE ( goods_id = '100' )
```

3.2.3 CURD 操作

3.2.3.1 读取数据

系统中可使用 select、find、query 方法完成查询操作。

使用 select 方法查询信息：

```
$model = Model('member');
// 查询会员表全部信息
$model->select();
//取得性别为 1 的会员列表信息，注意：select 方法需要在连贯操作中的最后一步出现
$model->where(array('member_sex'=>1))->select();
```

使用 find 方法查询信息：

```
$model = Model('member');
// 查询 ID 为 5 的会员信息
$model->where(array('member_id'=>5))->find();
```

使用 query 方法取得查询信息：

```
Model()->query('SELECT * FROM `shopnc_member` LIMIT 10');
```

使用动态方法取得查询信息：

```
$model = Model('member');
//使用 getby_动态方法，取得 member_name 为 kevin 的会员信息
$model->getby_member_name('kevin');
//等同于 SQL：SELECT * FROM `member` WHERE ( member_name = 'kevin' ) LIMIT 1

//使用 getfby_方法，取得 member_id 为 6 的会员名
$a = $model->getfby_member_id(6,'member_name');   //返回 kevin
//等同于 SQL：SELECT member_name FROM `shopnc_member` WHERE ( member_id = '6' ) LIMIT 1
```

3.2.3.2 更新数据

系统可使用 update、execute 方法完成更新操作。

```
$model = Model();
//更新主键(link_id)为 37 的记录信息
$data = array(
    'link_title'=>'ShopNC',
    'link_url'=>'http://www.shponc.net',
    'link_sort'=>32,
    'link_id'=>37
);
$model->table('link')->update($data);  // 系统自动以主键 link_id 为更新条件
//指定更新条件
$data = array(
    'link_title'=>'ShopNC',
    'link_url'=>'http://www.shponc.net',
    'link_sort'=>32
);
$model->table('link')->where(array('link_id'=>37))->update($data);
//使用 execute 方法执行更新
Model()->execute('UPDATE `shopnc_goods` SET goods_click=1000 WHERE goods_id=2');
```

3.2.3.3 插入数据

系统可使用 insert、insertAll、execute 方法完成插入操作。

使用 insert 方法插入单行数据：

```
//向 link 表插入数据，并返回最新生成的主键 ID
$model = Model('table');
$data = array(
    'link_title'=>'ShopNC',
    'link_url'=>'http://www.shopnc.net',
    'link_sort'=>32,
);
```

```
$model->insert($data);
```

使用 Insert 方法执行 replace 操作，将第二个参数设置为 true，如：

```
$model = Model();
$data = array(
'link_title'=>'ShopNC',
'link_url'=>'http://www.shopnc.net',
'link_sort'=>32,
'link_id'=>30
);
$model->table('link')->insert($data,true);
```

使用 insertAll 方法：实现批量插入数据：

```
$model = Model('link');
$data = array(
array(
    'link_title'=>'新浪',
    'link_url'=>'http://www.sina.com',
    'link_sort'=>32,
),
array(
    'link_title'=>'百度',
    'link_url'=>'http://www.baidu.com',
    'link_sort'=>30,
  )
);
$model->insertAll($data);
```

也可以使用 execute 直接执行 SQL 语句来插入数据。

3.2.3.4 删除数据

系统可使用 delete、clear、execute 方法完成删除操作。

使用 delete 方法删除数据：

```
$model = Model('link');
//或者
```

```
$model->where(array('link_id'=>5))->delete();
```

使用 clear 方法清空数据：

```
//清空 link 表
$model = Model();
$model->table('link')->clear();
```

也可以使用 execute 直接执行 SQL 语句来删除数据。

3.2.4 运算符

系统对常用运算符的使用进行了二次封装，使用方便、快捷。

gt ： 大于（>）

egt ： 大于等于（>=）

lt ： 小于（<）

elt ： 小于等于（<=）

eq ： 等于（=）

neq ： 不等于（!=）

notlike ： NOT LIKE

like ： 同 sql 中的 LIKE

between：同 sql 中的 BETWEEN

[not] in：同 sql 中的 [NOT] IN

示例：

```
//为便于演示，这里将所有运算符的使用均罗列出来，以下代码不可直接运行
$condition=array()
// uid > 5
$condition['uid'] = array('gt',5);
 // uid < 5
 $condition['uid'] = array('lt',5);
 // uid = 5
 $condition['uid'] = array('eq',5);
 // uid >= 5
 $condition['uid'] = array('egt',5);
 // uid <= 5
 $condition['uid'] = array('elt',5);
```

```
// uid 在 3,5,19 之间一个或多个
$condition['uid'] = array('in','3,5,19');
// uid 是 3,5,19 中的任何值
$condition['uid'] = array('not in','3,5,19');
// 5 <= uid <= 19
$condition['uid'] = array('between','5,19');
//product_name like 'a%'
$condition['product_name'] = array(array('like','a%'));
// product_name like 'a%' or product_name like 'b%'
$condition['product_name'] = array(array('like','a%'),array('like','b%'),'or');
//会员昵称或姓名有一个含有 shopnc 字样的即可满足
$condition['member_name|member_trname'] = array(array('like','%shopnc%'));
//会员昵称或姓名都必须含有 shopnc 字样的才可满足
$condition['member_name&member_trname'] = array(array('like','%shopnc%'));
//以上各条件默认均是 "AND" 关系，即每个条件都需要满足，如果想满足一个即可（ "OR" 关系），可增加以下条件
$condition['_op'] = 'or';
//最后将以上条件传入 where 方法
$list = Model(TABLE)->where($condition)->select();
```

3.3 视图

系统采用 MVC 模式，由视图类 Tpl 将变量抛到模板并进行输出，使用 setDir 设置模板目录，使用 output 方法抛出变量，使用 showpage 显示模板，抛出的变量会赋值到模板的 $output 数组中。

```
Tpl::setDir('home');    // 指定模板位于 templates/default/home
Tpl::output('name','shopnc');  // 向模板抛出变量
Tpl::showpage('index');  // 显示模板 /templates/default/home/index.php
echo $output['name'];  // 在模板中使用$output[$var]直接输出
```

3.4 调试

开启调试模式可以看到更加详细的系统运行信息，调试模式的作用在于显示更多的运行日志信息，以便在项目开发过程中快速定位和解决问题。开启调试模式需要在 config.ini.php 中设置：

```
$config['debug'] = true;
```

3.5 安全

系统为保护目录及文件安全，在所有敏感的目录中放置一个 1 字节的 index.html 文件，内容为一个空格，以避免当 http 服务器的 Directory Listing 打开时，服务器文件被索引和列表。

为防止系统内文件被非法调用，系统会在所使用的.php 文件头部增加有合法性验证：

```
defined('InShopNC') or exit('Access Invalid!');
```

3.6 开发规范约定

3.6.1 全局约定

文件编码

调整编辑器文件编码为 UTF-8，并关闭 UTF-8 BOM 的功能。不要使用 windows 自带的记事本编辑项目文件。

文件规范

如果为纯 php 文件(没有嵌套 HTML)，请不要用?>符号结尾，保持最后一行留空即可。

```
<?php
//this is a test file
echo 'hello';
<---这行留空
```

缩进

代码缩进使用 4 个空格(space)，而不是制表符(tab)，务必统一调整编辑器设置。

3.6.2 命名

文件命名

文件名必须使用小写字母和下划线，不能使用大写字母

文件夹不能出现点

Model 文件命名

模型名 + “.model.php”，模型名使用小写，如果多个词，之间使用下划线隔开
例：order.model.php

Control 文件命名

Control 名+ “.php”，control 名小写，多个词之间用下划线隔开
例：member_order.php

Template 文件命名

功能名称+“.”+op 业务描述，多个词之间用下划线隔开
例：order.list.php
order.add.php

类名

Control 类名

Control 名+ “Control” control 名使用小写，多词之间用下划线隔开
例：member_orderControl

Model 类名

Model 名+ “Model” model 名使用小写，多词之间用下划线隔开
例：orderModel

函数命名

Control 类函数

公用的:名字 +“Op”,名字使用小写(多词之间用下划线)
例：order_listOp

私有的：使用”_” + 名字,名字使用小写(多词之间用下划线)
例：_get_order_list

其它函数

其它方法的命名都使用驼峰（首字母小写）

例：getOrderList

命名约定

对常用增删改等操作约定如下

- 增加 add
- 编辑 edit
- 删除 del
- 列表 getList
- 详细 getInfo

Model 函数命名使用驼峰方式

取列表类时开头”get”+中间内容+尾缀”List”

取一条时开头”get”+中间内容+尾缀”Info”

增加时开头”add”+内容

修改时开头”edit”+内容

删除时开头”del”+内容

例：

```
getGoodsOlineList
getGoodsOffList
getGoodsList
addGoods
delGoods
editGoods
```

变量命名

Pulic:小写下划线

例：$goods_id

Private:下划线开头

例：$_goods_id

常量

全大写字母，不同词用下划线隔开

例：SITE_URL

3.6.3 编码规范

大括号位置

```
class memberControl {
    //code
}
```

逗号

函数中用逗号来分隔参数，所有的参数与前面的逗号之间要空格(第一个参数除外)。

```
public function connect($host, $port, $db, $user, $password, $charset = NULL)
```

关键字统一使用小写

```
    if
    for
    forearh
    switch
    true
    false
    null
    return
    empty
```

函数返回值

如果有内容返回内容

查询结果为空时返回 null

如果中间出现错误，返回 false

空格使用

除了参数之间要使用空格外，所有操作符之间都要使用空格，包括字符连接符(.)。

```
$host . ':' . $port

if

if ($a > $b) {
    //code
} elseif ($b > $c) {
    //code
} else {
    //code
}

if ($a && $b) {
    //code
} elseif ($d && $e) || $f) {
    //code
} else {
    //code
}

foreach

foreach($list as $key => $value) {
    //code
}

for

for ($i = 0; $i < 100; $i++) {
    //code
}

switch

switch ($key) {
    case 'value':
        //code
        break;
    default:
        //code
```

```
        break;
}

array

$list = array(
    'name' => 'xiaming',
    'age' => 17
);
```

注释

头部注释

```
<?php
/**
 * 文件说明
 *
 * @copyright  Copyright (c) 2007-2013 ShopNC Inc. (http://www.shopnc.net)
 * @license    http://www.shopnc.net
 * @link       http://www.shopnc.net
 * @since      File available since Release v1.1
 */
```

函数注释

```
/**
 * 函数说明
 *
 * @param array $input 更新内容
 * @param array $condition 更新条件
 * @return bool
 */
```

代码块注释

```
//声明一个变量
$goods_id = 10;
```

3.6.4 数据库设计原则

数据库表名不超过 30 个字符

表引擎统一使用 INNODB