1. 系统功能
   1. 需求

品牌管理：

商品分类管理：(无限极分类)

商品管理：

账号管理：

权限管理：

菜单管理：

订单管理：

* 1. 流程

自动登录流程

购物车流程

订单流程

1. 品牌功能
   1. 需求

品牌管理功能涉及品牌的列表展示、品牌添加、修改、删除功能。

品牌需要保存缩略图和简介。（注意图片上传）

品牌删除使用逻辑删除。（主要是通过改变状态值来删除）

2.2要点难点及解决方案

* + - 1. 删除使用逻辑删除,只改变status属性,不删除记录，最后彻底删除
      2. 使用uploadify插件,提升用户体验
      3. 使用composer下载和安装uploadify
      4. //关联品牌分类
      5. public function getBrand()
      6. {
      7. /\*\*
      8. \* hasOne参数 1. 关联对象的类名
      9. \* 参数2 [key=>value] key 表示关联对象的主键 value 表示关联对象在当前对象的字段
      10. \*/
      11. return $this->hasOne(Brand::className(),['id'=>'brand\_id']);
      12. }
      13. //定义分类
      14. public static function getBrandOptions()
      15. {
      16. //第一种找到分类id和分类name
      17. $brands = Brand::find()->asArray()->all();
      18. return ArrayHelper::map($brands,'id','name');
      19. }

1. 文章管理
   1. 需求

文章的增删改查

文章分类的增删改查（和文章是一对多的关系）

* 1. 要点

文章模型和文章详情模型建立1对1关系

* 1. 要点难点及解决方案
     + 1. 文章分类不能重复,通过添加验证规则unique解决
       2. 因为文章内容过多，所以需要分表

1. 商品分类
   1. 需求
      * 1. 商品分类的增删改查
        2. 商品分类支持无限级分类（主要是parent\_id）
        3. 方便直观的显示分类层级
        4. 分类层级允许修改
   2. 设计要点
      * 1. 使用嵌套集合模型
        2. 使用ztree插件显示分类层级
        3. 商品分类数据表设计

4.3难点

* + - 1. 理解无限级分类的原理
      2. nestedset行为的配置和使用
      3. ztree插件的使用

1. 商品管理
   1. 需求
      * 1. 保存每天创建多少商品,创建商品的时候,更新当天创建商品数量
        2. 商品增删改查
        3. 商品列表页可以进行搜索(商品名,商品状态,售价范围
        4. 新增商品自动生成sn,规则为年月日+今天的第几个商品,比如201704010001
        5. 商品详情使用ueditor插件
   2. 要点难点及解决方案
      * 1. 商品分类只能选择第三级分类
        2. 品牌使用下拉选择
        3. 商品相册,添加完商品后,跳转到添加商品相册页面,允许多图片上传
        4. 创建goods\_day\_count数据迁移时,如何创建date类型主

1. 管理员
   1. 需求
      * 1. 管理员增删改查（

1.加密

$password=\Yii::$app->security->generatePasswordHash('123456');

2.验证密码

$password=\Yii::$app->security->validatePassword();）

* + - 1. 管理员登录和注销
      2. 自动登录(基于cookie)
      3. 促销管理(选做)
  1. 要点
     + 1. 创建admin表(在user表基础上添加最后登录时间和最后登录ip)

1. RBAC权限控制
   1. 需求
      * 1. 权限的增删改查
        2. 角色的增删改查
        3. 角色和权限关联
        4. 用户和角色关联
        5. 菜单的增删改查
        6. 菜单和权限关联
   2. 设计要点
      * 1. 配置RBAC（在common配置authManager组件，执行rbac数据迁移）
        2. 根据authManager相应方法进行开发
        3. 添加角色的时候需要关联权限（addchild）
        4. 修改角色（先清空该角色的权限，然后再关联）

自定义验证方法 public function validate+字段名($attribute,$params){}

$attribute->属性

$params->条件

1.首先在表单模型中定义一个方法,把方法名写在验证规则中

2.只处理错误的情况

1. 菜单管理
   1. 需求
      * 1. 菜单增删改查
        2. 菜单和权限关联（一一对应）
        3. 根据当前用户的权限显示菜单
   2. 数据表设计
   3. 要点难点
      * 1. 菜单分为一级菜单和二级菜单
        2. 建立一级菜单和二级菜单的关系（1对多）
        3. 根据用户权限显示菜单（使用user组件的can(权限名称)方法来判断是否有权限）
        4. 有权限才显示一级菜单（判断是否有二级菜单权限，如果没有，不显示一级菜单）
        5. 实现RBAC权限控制，配合过滤器（在控制器添加过滤器）
2. 用户模块
   1. 需求
      * 1. 用户注册
        2. 用户登录（自动登录）（用户最后一次登录ip使用int类型保存）使用ip2long将IP保存成int类型.需要显示的是够,使用long2ip转化成ip格式的数据
        3. 收货地址管理（收货地址增删改查）,根据用户ID显示出该用户下面的所有地址,添加的时候,(难点,收货地址的三级联动,默认地址的设置)
   2. 设计要点
      * 1. 发送短信验证码，
        2. 验证手机号码（提交表单后，将手机号码和短信验证码从session中取出对比）
        3. 默认地址（设置当前地址为默认地址，让其他地址变成非默认地址。）
        4. 省市县三级联动
        5. 阿里大于插件地址
3. 前台模块
   1. 需求（使用小部件，添加Asset，拼接HTML）
      * 1. **前台首页**（商品分类，帮助信息，促销信息选做）查找出一级分类,然后循环找出同一棵树下的二级,三级分类
        2. **商品列表**（左侧分类）根据用户点击的分类ID,获取该分类所在同一棵树的所有分类,并且通过一对多的关系,显示该分类下面的所有商品
        3. **商品详情**（相关分类，商品信息，商品介绍）
        4. **加入购物车**，如果用户未登录,就把商品信息保存到cookie中,如果已经登录就保存到数据库中.
        5. **下订单**, 循环购物车中每一个商品,判断库存是否充足,如果库存不足,开启事务,回滚! 如果充足,就保存减少商品表中的数量,并且保存到订单表和订单详情表中,!(注意:商品表数量的扣除,购物车表数据清空.)
4. 微信支付
   1. 生成订单
   2. 调统一下单API
   3. 返回预支付交易链接(code\_url)
   4. 将链接生成二维码图片
   5. 微信扫码支付
   6. 异步通知商家支付结果

11.7 支付完成后改变订单状态

1. 性能优化

**1 清除超时未支付的订单**(状态为:未支付,and超过规定时间),1.找到超时订单ID 2.修改订单状态 3.返库存(商品数量返回的多少根据订单详情表中的数量)

**set\_time\_limit=()设置脚本可执行最大时间**

**死循环 while(true)**

**间隔时间 sleep(int))**

在命令行中执行: yii 路由 [参数] 比如 yii task/clear

执行后,清除订单和订单详情表中对应的数据,最后返回商品表中开始扣除的库存

**2 防止刷短信获取验证码 :**

//每天每个手机号码只能发送3条短信

//验证码数据表字段:tel code times day send\_time

// 13312345678 1234 3 2017-04-17 123454545

//根据电话号码查询发送记录表

// 1 没有记录，可以发送短信 生成一条记录 13312345678 1 2017-04-17

//2 有记录

// 判断日期是否是今天，如果不是 可以发送 修改日期为今天 修改次数为1

//如果是今天 判断次数是否大于等于3 ，如果大于等于3 不能发送短信，提示短信发送超出限制，请明天再试

//次数小于3 则可以发送 修改次数为 +1

//网络不好，导致短信不一致（短时间内，发送同样的验证码）

//网站发送的验证码 1 1234 2 1234 (5分钟之内)

//1234 1234

//>1 检查上次发送的时间戳距离当前时间戳是否大于5分钟

//检查发送记录表 没有记录 重新生成验证码

//有记录 看发送时间戳time() - send\_time > 60\*5 重新生成验证码，否则 使用上次的验证码

**3 Session入redis**(提高性能)

**4 中文分词查询**(用处:搜索) sphinx coreseek

1. .解压
2. .配置 php.ini->csft.conf
3. 建立索引 indexer.exe --all
4. 测试分词查询
5. 使用api查询(先开启searched服务,命令行中执行searched)

**5 常见的场景设计方案 redis**

**秒杀: lpop list**

**抽奖: spop jiangpin**

1. API接口开发(只处理数据)
   1. 安装安卓模拟器
   2. 安装jd.apk
   3. 修改服务器地址(清除缓存)
2. 笔记：

验证码

1.模型里面定义一个属性code,用来显示验证码

2. 给验证码定义验证规则['code','captcha']

3. 表单里面显示验证码 $form->field($model,'code')->widget(\yii\captcha\Captcha::className());

4. 验证码只能验证一次,不能重复验证. save方法需要关闭自动验证 save(false)

分页

1.计算 总条数

2. 设置每页显示多少条

3.实例化分页工具类 指定总条数 每页多少条

4. 设置limit参数,获取当页数据

5.视图使用 LinkPager 显示分页工具条

echo \yii\widgets\LinkPager::widget(['pagination'=>$pager]);

文件上传

1. model中需要单独定义一个属性(img\_file),用来保存上传文件对象

2. 表单中需要指定文件上传输入框,属性名使用定义的属性(img\_file)

3. 给属性(img\_file)设置验证规则

4. 在验证之前,实例化上传文件对象

5. 移动上传文件

//关联品牌分类

public function getBrand()

{

/\*\*

\* hasOne参数 1. 关联对象的类名

\* 参数2 [key=>value] key 表示关联对象的主键 value 表示关联对象在当前对象的字段

\*/

return $this->hasOne(Brand::className(),['id'=>'brand\_id']);

}

//定义分类

public static function getBrandOptions()

{

//第一种找到分类id和分类name

$brands = Brand::find()->asArray()->all();

return ArrayHelper::map($brands,'id','name');

}

1.数据迁移

yii migrate/create create\_表名\_table

yii migrate

2.建立空的数据表

3.在GII中建立模型

4.在GII中建立控制器

5.写基础Add方法, 建立add视图 注意:radio类的

6.完善add方法 添加几条数据

7.完善index方法 完善index视图

二丶 用户 角色 权限

Yii 提供了两套授权管理器： yii\rbac\PhpManager 和 yii\rbac\DbManager。前者使用 PHP 脚本存放授权数据， 而后者使用数据库存放授权数据。

DbManager 使用4个数据库表存放它的数据：

itemTable： 该表存放授权条目（译者注：即角色和权限）。默认表名为 "auth\_item" 。

itemChildTable： 该表存放授权条目的层次关系。默认表名为 "auth\_item\_child"。

assignmentTable： 该表存放授权条目对用户的指派情况。默认表名为 "auth\_assignment"。

ruleTable： 该表存放规则。默认表名为 "auth\_rule"。

继续之前，你需要在数据库中创建这些表。你可以使用存放在 @yii/rbac/migrations 目录中的数据库迁移文件来做这件事：yii migrate --migrationPath=@yii/rbac/migrations

现在可以通过 \Yii::$app->authManager 访问 authManager 。

创建权限 createPermission()

$createPost = $auth->createPermission('createPost');

获取所有权限 getPermissions()

获取一个权限 getPermission()

创建角色 createRole()

$author = $auth->createRole('author');

获取所有角色 getRoles()

获取一个角色 getRole()

给角色添加权限 addChild('角色','权限')

$auth->addChild($admin, $updatePost);

给用户指派角色 assign('角色','ID') IdentityInterface::getId() 返回的id

自定义验证方法 public function validate+字段名($attribute,$params){}

$attribute->属性

$params->条件

1.首先在表单模型中定义一个方法,把方法名写在验证规则中

2.只处理错误的情况

App Key:

23746798

App Secret:

2c718f8d2b2b6d3ceb87360fdea8ce0

array\_key\_exists() 判断数组中是否有给定的键名

1. 面试
2. 介绍项目 (项目目的,开发周期,开发人员)
3. 项目中有什么难点?怎么解决?(中文分词查询,找到sphinx;)
4. 角色权限体系(前台:ACF存取控制过滤器,后台:RBAC基于角色的存取控制)
5. 防止暴力破解用户的密码(1.验证码(登录时错误次数过多,开启验证码);2.通过session控制登录次数)
6. 购物流程 加入购物车(未登录:cookie;登录:数据表)->下单(需要登录;把cookie中的数据加到数据库 先判断库存,事务回滚)
7. 防止恶意短信刷取 (同一号码每天规定发送多少次短信)
8. Get和post的区别

参数传递:get地址栏 post请求头

传输量:get少 post多

1. cookie session

保存:c 在浏览器 s 服务器

s基于c

安全 :c里的不安全 s不能修改