11. view层：

<link rel="stylesheet" href="css/sort.css" />

<?php

$brand\_id=Yii::$app->session['brand\_id'];

$category\_id=Yii::$app->session['category\_id'];

use yii\widgets\LinkPager;

?>

<div id="div">

<center>

<input type="text" id="goods" value="<?php echo @$goods ?>"><input type="button" value="查询" id="search"><br>

<!-- 品牌处理 -->

品牌：<a href="index.php?r=site/keyword&qb=0">全部</a>&nbsp;&nbsp;&nbsp;&nbsp;

<?php foreach($brand as $v) {?>

<?php if(@$brand\_id==$v['brand\_id']){ ?>

<a href="index.php?r=site/index&brand\_id=<?php echo $v['brand\_id'] ?>">

<font color='red'><?php echo $v['brand\_name'] ?></font>

</a>

<?php }else { ?>

<a href="index.php?r=site/index&brand\_id=<?php echo $v['brand\_id'] ?>"><?php echo $v['brand\_name'] ?></a>

<?php }?>

<?php }?><br>

<!-- 品牌完事 -->

<!-- 分类处理 -->

分类：<a href="index.php?r=site/keyword&qb=0">全部</a>&nbsp;&nbsp;&nbsp;&nbsp;

<?php foreach(@$category as $v) {?>

<?php if(@$category\_id==$v['cat\_id']){ ?>

<a href="index.php?r=site/index&category\_id=<?php echo $v['cat\_id'] ?>">

<font color="red"><?php echo $v['cat\_name'] ?></font>

</a>

<?php }else{ ?>

<a href="index.php?r=site/index&category\_id=<?php echo $v['cat\_id'] ?>">

<?php echo $v['cat\_name'] ?>

<?php }?>

<?php }?><br><br><br>

<!-- 分类完事 -->

<?php foreach($model as $v) {?>

<table>

<tr><td>商品名称：</td><td><?php echo $v['goods\_name'] ?></td></tr>

<?php } ?>

</table>

<?= LinkPager::widget(['pagination' => $pages]); ?>

</div>

<script type="text/javascript" src="jquery-2.1.0.min.js"></script>

<script type="text/javascript">

$("#search").click(function(){

var goods=$("#goods").val();

//alert(goods);die;

$.ajax({

type: "GET",

url: "index.php?r=site/keyword",

data: "goods="+goods,

success: function(msg){

$("#div").replaceWith(msg);

}

});

})

</script>

控制器层：

public function actionIndex()

{

@$brand\_id=$\_GET['brand\_id'];

@$category\_id=$\_GET['category\_id'];

if($brand\_id==""){

Yii::$app->session['category\_id']=$category\_id;

if(Yii::$app->session['brand\_id']==""){

$data = Goods::find()->andwhere(['cat\_id'=>$category\_id]);

}else{

$data = Goods::find()->andwhere([

'cat\_id'=>Yii::$app->session['category\_id'],

'brand\_id'=>Yii::$app->session['brand\_id'],

]);

}

}elseif($category\_id==""){

Yii::$app->session['brand\_id']=$brand\_id;

if(Yii::$app->session['category\_id']==""){

$data = Goods::find()->andwhere([

'brand\_id'=>$brand\_id,

]);

}else{

$data = Goods::find()->andwhere([

'brand\_id'=>Yii::$app->session['brand\_id'],

'cat\_id'=>Yii::$app->session['category\_id'],

]);

}

}

$brand = Brand::find()->all();

$category=Category::find()->all();

$pages = new Pagination(['totalCount' =>$data->count(), 'pageSize' => '6']);

$model = $data->offset($pages->offset)->limit($pages->limit)->all();

return $this->renderPartial('index',[

'model' => $model,

'pages' => $pages,

'brand' => $brand,

'category' => $category,

]);

}

public function actionKeyword(){

@$goods=$\_GET['goods'];

//$goods=trim(' ',$goods);

//echo $goods;die;

@$brand\_id=$\_GET['brand\_id'];

@$category\_id=$\_GET['category\_id'];

@$qb=$\_GET['qb'];

if($qb==0){

unset(Yii::$app->session['category\_id']);

unset(Yii::$app->session['brand\_id']);

$data = Goods::find();

}

elseif($goods==""){

$data = Goods::find();

}else{

unset(Yii::$app->session['category\_id']);

unset(Yii::$app->session['brand\_id']);

$data = Goods::find()->where("goods\_name like '%$goods%'");

}

$brand = Brand::find()->all();

$category = Category::find()->all();

$pages = new Pagination(['totalCount' =>$data->count(), 'pageSize' => '6']);

$model = $data->offset($pages->offset)->limit($pages->limit)->all();

return $this->renderPartial('index',[

'model' => $model,

'pages' => $pages,

'goods' => $goods,

'brand' => $brand,

'category' => $category,

]);

}

12.

应用场景：项目中文章的信息内容因为持续有新增，而文章总量的基数又比较大，所以做搜索的时候，用了主索引+增量索引这种方式来实现索引的实时更新。

实现方式：

实现原理：

1. 新建一张表，记录一下上一次已经创建好索引的最后一条记录的ID

2. 当索引时，然后从数据库中取出所有ID大于上面那个sphinx中的那个ID的数据， 这些就是新的数据，然后创建一个小的索引文件

3. 把上边我们创建的增量索引文件合并到主索引文件上去

4. 把最后一条记录的ID更新到第一步创建的表中

值得注意的两点：

1）当合并索引的时候，只是把增量的索引合并进主索引中，增量索引本身并不会变化，也不会被删除；

2）当重建主索引的时候，增量索引就会被删除；

具体操作实现流程：

1. 新建一张表，用于存储已经建过索引的最大的doc\_id

CREATE TABLE `sph\_counter` (

`counter\_id` int(11) NOT NULL COMMENT '标识不同的数据表',

`max\_doc\_id` int(11) NOT NULL COMMENT '每个索引表的最大ID,会实时更新',

PRIMARY KEY (`counter\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8

2. 配置索引文件

复制代码

#主索引数据源定义

source article\_main

{

type = mysql

sql\_host =xxx.xxx.xxx.xx

sql\_user =

sql\_pass =

sql\_db =

sql\_port = 3306

sql\_query\_pre = SET NAMES utf8

sql\_query\_pre = REPLACE INTO sph\_counter SELECT 1, MAX(id) FROM documents

sql\_query\_range =

sql\_range\_step = 10000

sql\_query = \

SELECT \*\

FROM documents WHERE id>=$start AND id<=$end

sql\_attr\_timestamp = pubtime #从SQL读取到的值必须为整数，作为时间属性

sql\_query\_info\_pre = SET NAMES utf8 #命令行查询时，设置正确的字符集

sql\_query\_info = SELECT \* FROM documents WHERE id=$id #命令行查询时，从数据库读取原始数据信息

}

复制代码

复制代码

# 增量索引数据源定义

source article\_delta : article\_main

{

sql\_query\_pre = SET NAMES utf8

sql\_query\_range =

sql\_range\_step = 10000

sql\_query = \

SELECT \*\

FROM documents WHERE id>=$start AND id<=$end

sql\_attr\_timestamp = pubtime #从SQL读取到的值必须为整数，作为时间属性

# 增量索引创建完成之后，更新最大的doc\_id

sql\_query\_post = UPDATE sph\_counter SET max\_doc\_id=(SELECT MAX(id) FROM documents) where counter\_id=1

# REPLACE INTO sph\_counter SELECT 1, MAX(id) FROM documents

sql\_query\_info\_pre = SET NAMES utf8 #命令行查询时，设置正确的字符集

sql\_query\_info = SELECT \* FROM article\_info WHERE id=$id #命令行查询时，从数据库读取原始数据信息

}

复制代码

复制代码

# 主索引index定义

index article\_main

{

source = article\_main #对应的source名称

path = /data/... #请修改为实际使用的绝对路径，例如：/usr/local/coreseek/var/...

docinfo = extern

mlock = 0

morphology = none

min\_word\_len = 1

html\_strip = 0

#中文分词配置，详情请查看：http://www.coreseek.cn/products-install/coreseek\_mmseg/

charset\_dictpath = /usr/local/mmseg3/etc/ #BSD、Linux环境下设置，/符号结尾

#charset\_dictpath = etc/ #Windows环境下设置，/符号结尾，最好给出绝对路径，例如：C:/usr/local/coreseek/etc/...

charset\_type = zh\_cn.utf-8

}

# 增量索引index定义

index article\_delta : article\_main

{

source = article\_delta

path = /data/....

docinfo = extern

mlock = 0

morphology = none

min\_word\_len = 1

html\_strip = 0

#中文分词配置，详情请查看：http://www.coreseek.cn/products-install/coreseek\_mmseg/

charset\_dictpath = /usr/local/mmseg3/etc/ #BSD、Linux环境下设置，/符号结尾

#charset\_dictpath = etc/ #Windows环境下设置，/符号结尾，最好给出绝对路径，例如：C:/usr/local/coreseek/etc/...

charset\_type = zh\_cn.utf-8

}

复制代码

配置文件中修改好本部分之后，需要重新启动一下searchd进程，让其加载新的配置文件

sudo /usr/local/coreseek4/bin/searchd -c /usr/local/coreseek4/etc/xxxx.conf --stop

sudo nohup /usr/local/coreseek4/bin/searchd -c /usr/local/coreseek4/etc/xxxx.conf &

新建主索引 ：

sudo /usr/local/coreseek4/bin/indexer -c /usr/local/coreseek4/etc/xxxx.conf article\_main --rotate

过一段时间再新建增量索引(需要将此命令放到定时任务中，跑的频率按照实际需求来定)

sudo /usr/local/coreseek4/bin/indexer -c /usr/local/coreseek4/etc/xxxx.conf article\_delta --rotate

下一步，自己可以用命令行的search来查询增量索引的内容

/usr/local/coreseek4/bin/search -c /usr/local/coreseek4/etc/xxxx.conf