1 学习目标

- 了解Thymeleaf的基本使用
- 实现商品详情页的渲染
- 知道页面静态化的作用
- 实现页面静态化功能

2 商品详情

当用户搜索到商品，肯定会点击查看，就会进入商品详情页，接下来我们完成商品详情页的展示，

2.1 thymeleaf

Thymeleaf是跟Velocity、FreeMarker类似的模板引擎，它可以完全替代JSP，相较与其他的模板引擎，它主要有以下几个特点：

1. Thymeleaf在有网络和无网络的环境下皆可运行，即它可以让美工在浏览器查看页面的静态效果，也可以让程序员在服务器查看带数据的动态页面效果。这是由于它支持 html 原型，然后在 html 标签里增加额外的属性来达到模板+数据的展示方式。浏览器解释 html 时会忽略未定义的标签属性，所以thymeleaf的模板可以静态地运行；当有数据返回到页面时，Thymeleaf 标签会动态地替换掉静态内容，使页面动态显示。
2. Thymeleaf开箱即用的特性。它提供标准和spring标准两种方言，可以直接套用模板实现JSTL、OGNL表达式效果，避免每天套模板、改jstl、改标签的困扰。同时开发人员也可以扩展和创建自定义的方言
3. Thymeleaf提供spring标准方言和一个与SpringMVC完美集成的可选模块，可以快速的实现表单绑定、属性编辑器、国际化等功能。
4. Thymeleaf官网：http://www.thymeleaf.org

2.2 模板项目搭建

商品详情浏览量比较大，并发高 一般都是要开启一个独立的微服务去做。

2.2.1 在mingrui-shop-service项目下新建mingrui-shop-service-template

2.2.2 pom.xml

```
    <dependencies>
        <!--模板引擎-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
    </dependencies>
```

2.2.3 application.yml

```
server:
```

```yaml
  port: 8400

spring:
  application:
    name: template-server
  thymeleaf:
    # 配置前缀-->模板文件存储路径
    prefix: classpath:/templates/
    # 是否检查本地模板
    check-template-location: true
    # 配置模板文件后缀
    suffix: .html
    # 编码格式
    encoding: UTF-8
    servlet:
      # 模板类型
      content-type: text/html
    #模板模式
    mode: HTML5
    # 是否启用缓存
    cache: false

# eureka配置
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
```

2.2.4 启动类

```java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.openfeign.EnableFeignClients;

/**
 * @ClassName RunTemplateServer
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/10
 * @Version V1.0
 **/
@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})
@EnableFeignClients
@EnableEurekaClient
public class RunTemplateServerApplication {
    public static void main(String[] args) {
        SpringApplication.run(RunTemplateServerApplication.class);
    }
}
```

2.2.5 在resource目录下新建templates文件夹

2.2.6 新建123.html

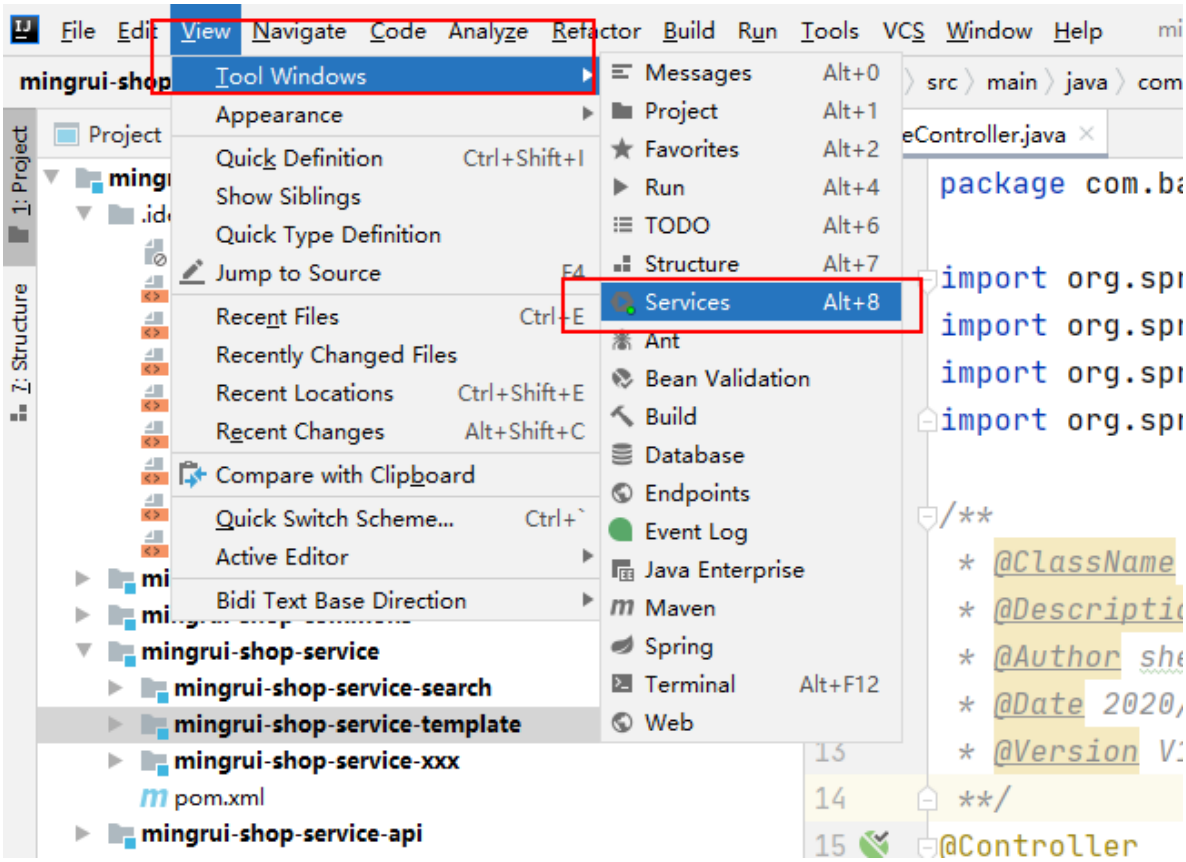
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
明瑞教育,世界第一
</body>
</html>
```

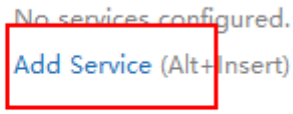
2.2.7 新建PageController

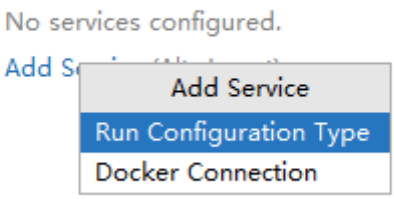
```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

/**
 * @ClassName PageController
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/10
 * @Version V1.0
 **/
@Controller
public class PageController {

    @GetMapping(value = "123.html")
    public String test(){
        return "123";
    }
}
```

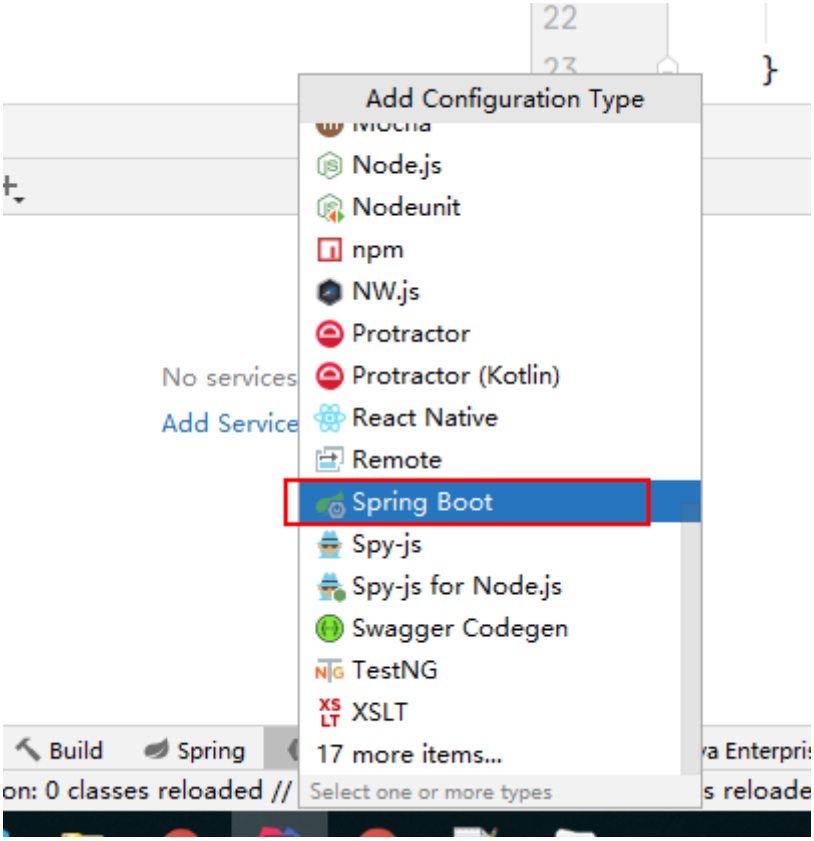
2.2.8 浏览器输入ip:port/123.html能正常访问到页面即可

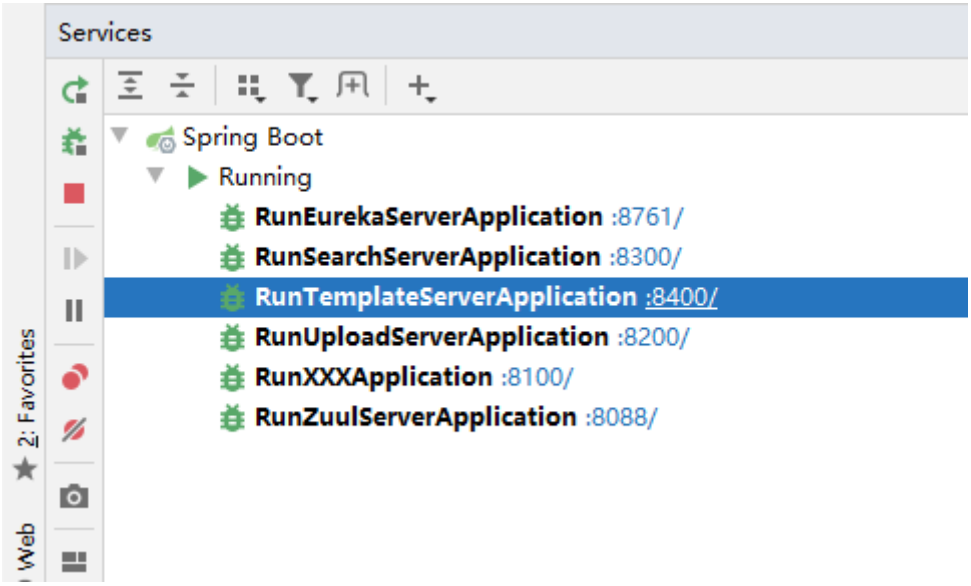
现在项目太多了,启动和停止项目相当费劲


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Tool Windows
Appearance
Quick Definition Ctrl+Shift+I
Show Siblings
Quick Type Definition
Jump to Source F4
Recent Files Ctrl+E
Recently Changed Files
Recent Locations Ctrl+Shift+E
Recent Changes Alt+Shift+C
Compare with Clipboard
Quick Switch Scheme... Ctrl+`
Active Editor
Bidi Text Base Direction
Messages Alt+0
Project Alt+1
Favorites Alt+2
Run Alt+4
TODO Alt+6
Structure Alt+7
Services Alt+8
Ant
Bean Validation
Build
Database
Endpoints
Event Log
Java Enterprise
Maven
Spring
Terminal Alt+F12
Web
1: Project
7: Structure
mingrui-service
mingrui-shop-service-search
mingrui-shop-service-template
mingrui-shop-service-xxx
pom.xml
mingrui-shop-service-api
src main java com
eController.java
package com.ba
import org.spr
import org.spr
import org.spr
import org.spr
/**
* @ClassName
* @Descripti
* @Author she
* @Date 2020/
* @Version V1
**/
@Controller

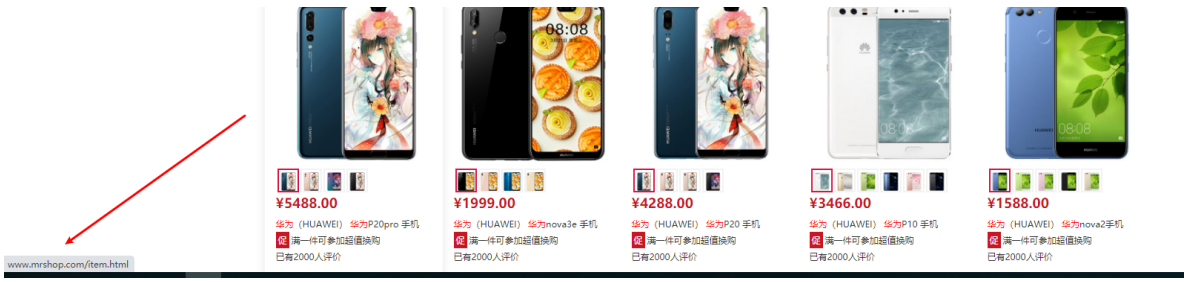

No services configured.
Add Service (Alt+Insert)


No services configured.
Add Service
Run Configuration Type
Docker Connection





2.3 展示item模板



先来看一下问题:

大家这个请求的地址是其实是127.0.0.1:80对吧

但是我们想让他请求我们template项目的地址

所以我们得先处理一下nginx

```
server {
    listen       80;
    server_name  www.mrshop.com;
    #如果请求路径带item我们就让nginx帮我们把请求转发到8400端口
    location /item {
        proxy_pass http://127.0.0.1:8400;
        proxy_connect_timeout 600;
        proxy_read_timeout 600;
    }
    location / {
        proxy_pass http://127.0.0.1:9002;
        proxy_connect_timeout 600;
        proxy_read_timeout 600;
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   html;
    }
}
```

```
#如果请求路径带item我们就让nginx帮我们把请求转发到8400端口
location /item {
    proxy_pass http://127.0.0.1:8400;
    proxy_connect_timeout 600;
    proxy_read_timeout 600;
}
```

既然这样改了,那我们就应该让图片的地址带上item


```
<!--大图片 设置成当前选中的图片-->
<a :href="'/item/' +goods.id + '.html'" target="_blank"><img
:src="goods.selected.image" height="200" /></a>
```

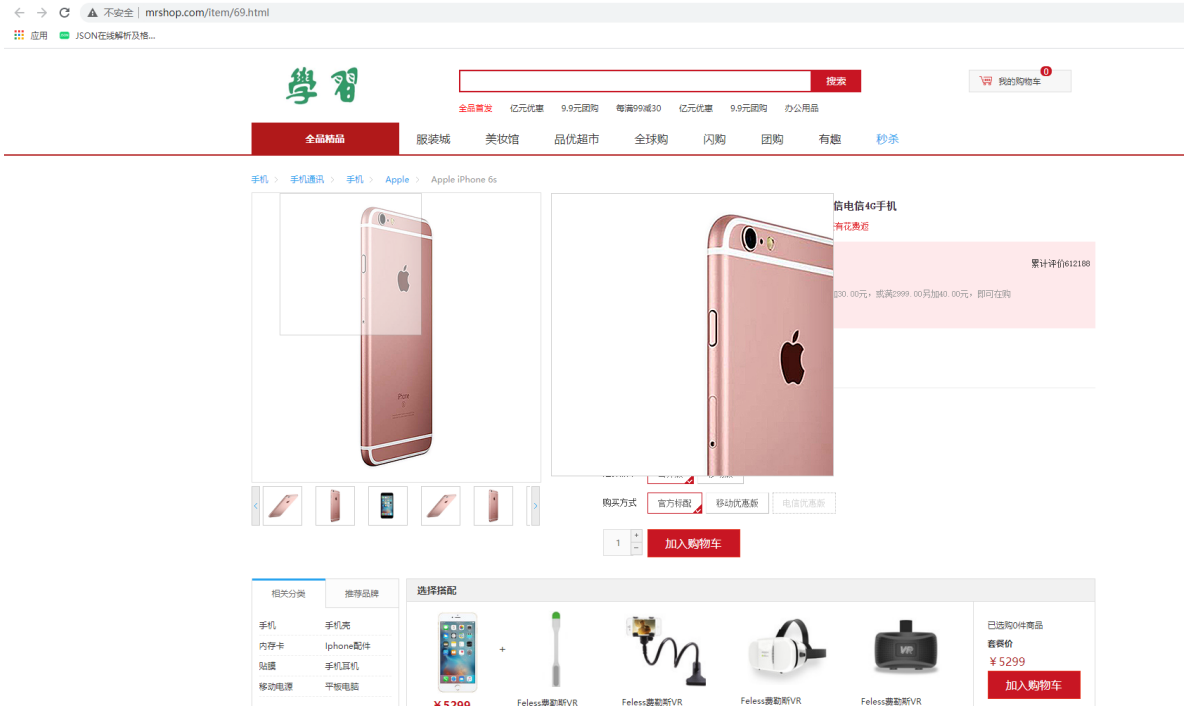
2.3.1 PageController

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;

/**
 * @ClassName PageController
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/10
 * @Version V1.0
 **/
@Controller
@RequestMapping(value = "item")
public class PageController {

    @GetMapping(value = "{spuId}.html")
    public String test(@PathVariable(value = "spuId") Integer spuId){
        //
        return "item";
    }
}
```

2.3.2 复制item.html到templates下

重启项目

点击图片能正常访问页面即可



重启项目报错的话就自己新建item.html,然后复制代码即可

2.4 后台数据准备

```
@Override
public Map<String, Object> getGoodsInfo(Integer spuId) {

    Map<String, Object> map = new HashMap<>();

    //spu信息
    SpuDTO spuDTO = new SpuDTO();
    spuDTO.setId(spuId);
    Result<List<SpuDTO>> spuResult = goodsFeign.getSpuInfo(spuDTO);
    if(spuResult.getCode() == 200){
        SpuDTO spuInfo = spuResult.getData().get(0);

        map.put("spuInfo",spuInfo);
        //spudetail信息
        Result<SpuDetailEntity> spuDetailResult = goodsFeign.getSpuDetailBydSpu(spuId);
        if(spuDetailResult.getCode() == 200){
            SpuDetailEntity spuDetailInfo = spuDetailResult.getData();

            map.put("spuDetailInfo",spuDetailInfo);
        }
```

```
//查询分类信息
Result<List<CategoryEntity>> cateResult = categoryFeign.getCateByIds(
        String.join(
                delimiter: ","
                , Arrays.asList(
                        spuInfo.getCid1() + ""
                        , spuInfo.getCid2() + ""
                        , spuInfo.getCid3() + "")
        )
);

if(cateResult.getCode() == 200){
    map.put("cateList",cateResult.getData());
}
//查询品牌信息
BrandDTO brandDTO = new BrandDTO();
brandDTO.setId(spuInfo.getBrandId());
Result<PageInfo<BrandEntity>> brandResult = brandFeign.getBrandInfo(brandDTO);

if(brandResult.getCode() == 200){

    map.put("brandInfo",brandResult.getData().getList().get(0));
}
```

分类信息

品牌信息

```
//规格组和规格参数
SpecGroupDTO specGroupDTO = new SpecGroupDTO();
specGroupDTO.setCid(spuInfo.getCid3());
Result<List<SpecGroupEntity>> sepcGroupResult = specificationFeign.getSepcGroupInfo(specGroupDTO);

if (sepcGroupResult.getCode() == 200) {
    List<SpecGroupEntity> specGroupInfo = sepcGroupResult.getData();
    //规格组和规格参数
    List<SpecGroupDTO> groupsInParams = specGroupInfo.stream().map(specGroup -> {

        SpecGroupDTO sgd = BaiduBeanUtil.copyProperties(specGroup, SpecGroupDTO.class);
        //规格参数-通用参数
        SpecParamDTO specParamDTO = new SpecParamDTO();
        specParamDTO.setGroupId(specGroup.getId());
        specParamDTO.setGeneric(1);
        Result<List<SpecParamEntity>> specParamResult = specificationFeign.getSpecParamInfo(specParamDTO);
        if (specParamResult.getCode() == 200) {
            sgd.setSpecParams(specParamResult.getData());
        }
        return sgd;
    }).collect(Collectors.toList());

    map.put("groupsInParams",groupsInParams);
}
```

通过分类id查询规格组

遍历规格组

将规格组转换成规格组DTO
规格组DTO中规格组下的规格参数

通过规格组查询通过规格参数

给规格组DTO设置规格参数

```
//特有规格参数
SpecParamDTO specParamDTO = new SpecParamDTO();
specParamDTO.setCid(spuInfo.getCid3());
specParamDTO.setGeneric(0);
Result<List<SpecParamEntity>> specParamResult = specificationFeign.getSpecParamInfo(specParamDTO);

if(specParamResult.getCode() == 200) {
    //需要将数据转换为map方便页面操作!!!!!!!!
    Map<Integer, String> specMap = new HashMap<>();
    specParamResult.getData().stream().forEach(spec -> specMap.put(spec.getId(),spec.getName()));
    map.put("specParamMap",specMap);
}
```

通过分类id查询特殊规格参数

将规格参数封装成map集合
key为id,值为规格参数名

```
//sku
Result<List<SkuDTO>> skuResult = goodsFeign.getSkuBySpuId(spuId);

if(skuResult.getCode() == 200){

    List<SkuDTO> skuList = skuResult.getData();
    map.put("skuList",skuList);
}
```

通过id查询sku集合

```
@Override
public Map<String, Object> getGoodsInfo(Integer spuId) {

    Map<String, Object> map = new HashMap<>();

    //spu信息
    SpuDTO spuDTO = new SpuDTO();
```

```
        spuDTO.setId(spuId);
        Result<List<SpuDTO>> spuResult = goodsFeign.getSpuInfo(spuDTO);
        if(spuResult.getCode() == 200){
            SpuDTO spuInfo = spuResult.getData().get(0);

            map.put("spuInfo",spuInfo);
            //spudetail信息
            Result<SpuDetailEntity> spuDetailResult =
goodsFeign.getSpuDetailBydSpu(spuId);
            if(spuDetailResult.getCode() == 200){
                SpuDetailEntity spuDetailInfo = spuDetailResult.getData();

                map.put("spuDetailInfo",spuDetailInfo);
            }
            //查询分类信息
            Result<List<CategoryEntity>> cateResult =
categoryFeign.getCateByIds(
                    String.join(
                            ","
                            , Arrays.asList(
                                    spuInfo.getCid1() + ""
                                    , spuInfo.getCid2() + ""
                                    , spuInfo.getCid3() + "")
                    )
            );

            if(cateResult.getCode() == 200){
                map.put("cateList",cateResult.getData());
            }
            //查询品牌信息
            BrandDTO brandDTO = new BrandDTO();
            brandDTO.setId(spuInfo.getBrandId());
            Result<PageInfo<BrandEntity>> brandResult =
brandFeign.getBrandInfo(brandDTO);

            if(brandResult.getCode() == 200){

                map.put("brandInfo",brandResult.getData().getList().get(0));
            }

            //规格组和规格参数
            SpecGroupDTO specGroupDTO = new SpecGroupDTO();
            specGroupDTO.setCid(spuInfo.getCid3());
            Result<List<SpecGroupEntity>> sepcGroupResult =
specificationFeign.getSepcGroupInfo(specGroupDTO);

            if (sepcGroupResult.getCode() == 200) {
                List<SpecGroupEntity> specGroupInfo = sepcGroupResult.getData();
                //规格组和规格参数
                List<SpecGroupDTO> groupsInParams =
specGroupInfo.stream().map(specGroup -> {

                    SpecGroupDTO sgd = BaiduBeanUtil.copyProperties(specGroup,
SpecGroupDTO.class);
                    //规格参数-通用参数
                    SpecParamDTO specParamDTO = new SpecParamDTO();
                    specParamDTO.setGroupId(specGroup.getId());
                    specParamDTO.setGeneric(1);
```

```
                    Result<List<SpecParamEntity>> specParamResult =
specificationFeign.getSpecParamInfo(specParamDTO);
                    if (specParamResult.getCode() == 200) {
                        sgd.setSpecParams(specParamResult.getData());
                    }
                    return sgd;
                }).collect(Collectors.toList());

                map.put("groupsInParams",groupsInParams);
            }

            //特有规格参数
            SpecParamDTO specParamDTO = new SpecParamDTO();
            specParamDTO.setCid(spuInfo.getCid3());
            specParamDTO.setGeneric(0);
            Result<List<SpecParamEntity>> specParamResult =
specificationFeign.getSpecParamInfo(specParamDTO);

            if(specParamResult.getCode() == 200) {
                //需要将数据转换为map方便页面操作!!!!!!!!!
                Map<Integer, String> specMap = new HashMap<>();
                specParamResult.getData().stream().forEach(spec ->
specMap.put(spec.getId(),spec.getName()));
                map.put("specParamMap",specMap);
            }

            //sku
            Result<List<SkuDTO>> skuResult = goodsFeign.getSkuBySpuId(spuId);

            if(skuResult.getCode() == 200){

                List<SkuDTO> skuList = skuResult.getData();
                map.put("skuList",skuList);
            }

        }

        return map;
    }
}
```

2.5 展示分类/品牌/标题

```
<div class="py-container">
    <div id="item">
        <div class="crumb-wrap">
            <ul class="sui-breadcrumb">

                <!--分类信息展示-->
                <li th:each="cate : ${ cateList }">
                    <a href="#" th:text="${ cate.name }"></a>
                </li>
                <!--品牌信息展示-->
                <li>
                    <a href="#" th:text="${ brandInfo.name }"></a>
                </li>
                <!--标题展示-->
                <li class="active" th:text="${ spuInfo.title }"></li>
            </ul>
        </div>
        <!--product-info-->
```

```
<!--分类信息展示-->
<li th:each="cate : ${ cateList }">
    <a href="#" th:text="${ cate.name }"></a>
</li>
<!--品牌信息展示-->
<li>
    <a href="#" th:text="${ brandInfo.name }"></a>
</li>
<!--标题展示-->
<li class="active" th:text="${ spuInfo.title }"></li>
```

2.6 展示子标题


```
<div class="fr itemInfo-wrap">
    <div class="sku-name">
        <h4>Apple iPhone 6s（A1700）64G玫瑰金色 移动通信电信4G手机</h4>
    </div>
    <!--子标题展示-->
    <div class="news"><span th:utext="${ spuInfo.subTitle }"></span></div>
    <div class="summary">
        <div class="summary-wrap">
            <div class="fl title"><i>价　　格</i></div>
            <div class="fl price">
                <i>¥</i><em>5299.00</em><span>降价通知</span>
```

```
<!--子标题展示-->
<div class="news"><span th:utext="${ spuInfo.subTitle }">
</span></div>
```

2.7 sku信息展示

```
const specParamMap = /*[[ ${specParamMap} ]]*/[];//得到 1:屏幕尺寸
const specialSpec = JSON.parse(/*[[ ${ spuDetailInfo.specialSpec } ]]*/); //得到 1:5.7
const skus = /*[[ ${ skuList } ]]*/[];
```

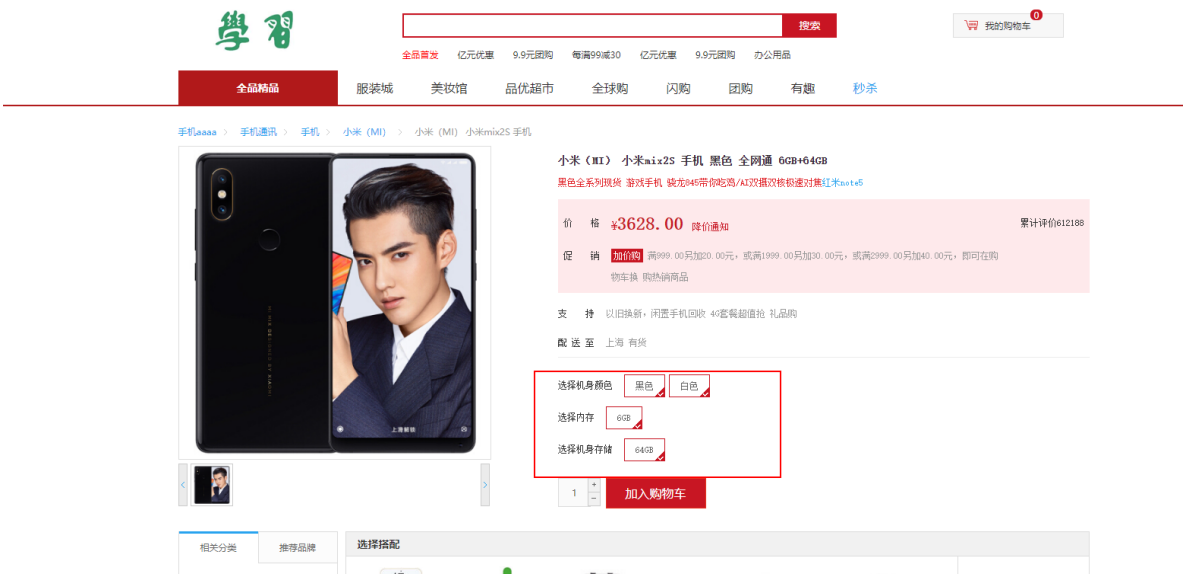

```
<script th:inline="javascript">
    const specParamMap = /*[[ ${specParamMap} ]]*/[];
    const specialSpec = JSON.parse(/*[[ ${ spuDetailInfo.specialSpec } ]]*/);
    const skus = /*[[ ${ skuList } ]]*/[];
</script>
<script>
    var itemVm = new Vue({
        el:"#itemApp",
        data:{
            mrshop,
            specParamMap,
            specialSpec,
            skus,
            indexs
        },
        components:{
            b2cTop: () => import('/js/pages/top.js')
        }
    });
</script>
```

```
<script th:inline="javascript">
    const specParamMap = /*[[ ${specParamMap} ]]*/[];
    const specialSpec = JSON.parse(/*[[ ${ spuDetailInfo.specialSpec } ]]*/);
    const skus = /*[[ ${ skuList } ]]*/[];
</script>
<script>
    var itemVm = new Vue({
        el:"#itemApp",
        data:{
            mrshop,
            specParamMap,
            specialSpec,
            skus,
            indexs
        },
        components:{
            b2cTop: () => import('/js/pages/top.js')
        }
    });
</script>
```

html代码

```
<!--特有规格展示-->
<dl v-for="(value,key) in specialSpec">
    <dt>
        <div class="fl title">
            <!--通过id得到特有规格的名称-->
            <i>选择{{specParamMap[key]}}</i>
```

```
                                    </div>
                                </dt>
                                <dd v-for="(o,index) in value" >
                                    <a href="javascript:;"  class="selected">
                                        {{ o }}<span title="点击取消选择"> 
</span>
                                    </a>
                                </dd>
                            </dl>
```



- 解决特有规格全部被选中的问题

```
<script th:inline="javascript">
    const specParamMap = /*[[ ${specParamMap} ]]*/[];
    const specialSpec = JSON.parse(/*[[ ${ spuDetailInfo.specialSpec } ]]*/);
    const skus = /*[[ ${ skuList } ]]*/[];

    const indexs = {};
    //得到所有特有属性的id所有的key对应的值设置为0
    Object.keys(specialSpec).forEach(k => {
        indexs[k] = 0;
    })

</script>
<script>
    var itemVm = new Vue({
        el:"#itemApp",
        data:{
            mrshop,
            specParamMap,
            specialSpec,
            skus,
            indexs
        },
        components:{
            b2cTop: () => import('/js/pages/top.js')
        },
```

```
    const indexs = {};
    //得到所有特有属性的id所有的key对应的值设置为0
    Object.keys(specialSpec).forEach(k => {
        indexs[k] = 0;
    })
```

```
                                <!--点击某一项规格的时候讲将当前下标设置为被选中的下标--
>
                                <dd v-for="(o,index) in value"
@click="indexs[key] = index">
                                    <!--如果当前下标==设置的下标
                                    那么此项规格被选中
                                    剩下的则不选中-->
                                    <a href="javascript:;"  :class="{selected:
index == indexs[key]}">
                                        {{ o }}<span title="点击取消选择"> 
</span>
                                    </a>
                                </dd>
```

- 显示当前选中规格的图片 价格 和标题

```
        computed:{
            //计算属性得到当前被选中的sku信息
            sku(){
                //获取到indexs中所有的值并且用_拼接起来
                const index = Object.values(indexs).join("_");
                //数组.find方法-->通过条件查找数组中的元素
                return this.skus.find(sku => sku.indexes == index);
            },
            //处理图片
            images(){
                return this.sku.images ? this.sku.images.split(",") : [];
            }
        }
```

```
                    <!--sku标题-->
                    <div class="sku-name">
                        <h4>{{ sku.title }}</h4>
                    </div>
```

```
                            <!--sku价格-->
                            <div class="fl price">
                                <i>¥</i><em>{{ mrshop.formatPrice(sku.price) }}
</em><span>降价通知</span>
                            </div>
```

```
<div class="product-info">
    <div class="fl preview-wrap">
        <!--放大镜效果-->
        <div class="zoom">
            <!--默认第一个预览-->
            <div id="preview" class="spec-preview">
                <span class="jqzoom">
                    <img :jqimg="images[0]" :src="images[0]" width="400px" height="400px"/>
                </span>
            </div>
            <!--下方的缩略图-->
            <div class="spec-scroll">
                <a class="prev"><</a>
                <!--左右按钮-->
                <div class="items">
                    <ul>
                        <li v-for="(image, index) in images">
                            <img :src="image" :bimg="image" onmousemove="preview(this)" />
                        </li>
                    </ul>
                </div>
                <a class="next">></a>
            </div>
        </div>
    </div>
    <div class="fr itemInfo-wrap">
        <!--sku标题-->
```

```
                <div class="fl preview-wrap">
                    <!--放大镜效果-->
                    <div class="zoom">
                        <!--默认第一个预览-->
                        <div id="preview" class="spec-preview">
                            <span class="jqzoom">
                                <img :jqimg="images[0]" :src="images[0]"
width="400px" height="400px"/>
                            </span>
                        </div>
                        <!--下方的缩略图-->
                        <div class="spec-scroll">
                            <a class="prev">&lt;</a>
                            <!--左右按钮-->
                            <div class="items">
                                <ul>
                                    <li v-for="(image, index) in images">
                                        <img :src="image" :bimg="image"
onmousemove="preview(this)" />
                                    </li>
                                </ul>
                            </div>
                            <a class="next">&gt;</a>
                        </div>
                    </div>
                </div>
```

2.8 商品介绍

```
                            <!--商品详情-->
                            <div class="intro-detail" th:utext="${
spuDetailInfo.description }">
                            </div>
```

2.9 规格与包装

2.9.1 规格

```
    const specialSpec = JSON.parse(/*[[ ${spuDetailInfo.specialSpec} ]]*/); //specialSpec 1 : 10.0

    const spuDetailInfo = /*[[ ${spuDetailInfo} ]]*/{}; //detail一并拿过来,后面有可能会用
    const genericSpec = JSON.parse(/*[[ ${ spuDetailInfo.genericSpec } ]]*/);//通过规格参数
    const specGroupAndParam = /*[[ ${specGroupAndParam} ]]*/[];//规格组合规格参数数据

    const indexes = {};
    Object.keys(specialSpec).forEach(key => {
        indexes[key] = 0;
    });

</script>
<script>
    var itemVm = new Vue({
        el:"#itemApp",
        data:{
            skusInfo,
            specParamMap,
            specialSpec,
            indexes,
            spuDetailInfo,
            genericSpec,
            specGroupAndParam
        },
        computed:{
```

image-20210305154219480

```
<!--
    规格组 规格参数展示
    v-if="group.specParams.length > 0" 如果当前规格组下没有规格参数则不展示
-->
<div class="Ptable-item"
     v-for="(group,index) in specGroupAndParam"
     :key="index">
    <h3>{{ group.name }}</h3>
    <dl v-for="(param,index) in group.specParams" :key="index">
        <dt >{{ param.name }}</dt>
        <dd>{{ genericSpec[param.id] || '无' }}</dd>
    </dl>
</div>
```

遍历规格组

遍历规格参数

如果规格参数没有值的话显示'无'

```
    const spuDetailInfo = /*[[ ${spuDetailInfo} ]]*/{}; //detail一并拿过来,后面有可能会用
    const genericSpec = JSON.parse(/*[[ ${ spuDetailInfo.genericSpec } ]]*/);//通过规格参数
    const specGroupAndParam = /*[[ ${specGroupAndParam} ]]*/[];//规格组合规格参数数据
```

```
    <!--
        规格组 规格参数展示
        v-if="group.specParams.length > 0" 如果当前规格组下没有规格参数则不展示
    -->
    <div class="Ptable-item"
         v-for="(group,index) in specGroupAndParam"
         :key="index">
        <h3>{{ group.name }}</h3>
        <dl v-for="(param,index) in group.specParams" :key="index">
            <dt >{{ param.name }}</dt>
            <dd>{{ genericSpec[param.id] || '无' }}</dd>
        </dl>
    </div>
```

2.9.2 包装

```
                                <div class="package-list">
                                    <h3>包装清单</h3>
                                    <p th:text="${ spuDetailInfo.packingList }">
</p>
                                </div>
```

2.10 售后保障

```
19                  <div id="three" class="tab-pane">
20                      <p th:text="${ spuDetailInfo.afterService }"></p>
21                  </div>
```

```
<div id="three" class="tab-pane">
    <p th:text="${ spuDetailInfo.afterService }"></p>
</div>
```

2.11 商品标题/价格/图片展示

> 前置说明：

```
computed:{
    sku () {                                                    计算出当前选中的sku
        /*
        将:
        {
            颜色 : 0,
            内存 : 1,
            机身存储 : 0
        }
        这样数据转换成0_1_0
        Object.values获取json对象的所有value值得到数据
        */
        const indexStr = Object.values(this.indexes).join('_');
        //数组的find函数: 通过条件查找数组内某一个函数
        //我们需要查询sku.indexes == indexStr的元素
        return this.skusInfo.find(sku => sku.indexes == indexStr);
    },
    images () {                                                 获取当前选中的sku中的图片
        //先判断一下当前选中sku的images是否有内容
        //然后再通过,分割字符串.因为有可能一个sku有多个图片
        return this.sku.images ? this.sku.images.split(',') : [];
    }
},
```

```
<div class="fr itemInfo-wrap">
    <div class="sku-name">
        <h4>{{ sku.title }}</h4>                                商品标题
    </div>
    <div class="news"><span th:utext="${spuInfo.subTitle}"></span></div>
    <div class="summary">
        <div class="summary-wrap">
            <div class="fl title"><i>价    格</i></div>
            <div class="fl price">                              当前商品价格,注意需要转换一下价格
                <i>¥</i><em>{{ sku.price }}</em><span>降价通知</span>
            </div>
            <div class="fr remark"><i>累计评价</i><em>612188</em></div>
        </div>
        <div class="summary-wrap">
            <div class="fl title">
                <i>促    销</i>
            </div>
            <div class="fl fix-width">
                <i class="red-bg">加价购</i>
                <em class="t-gray">满999.00另加20.00元，或满1999.00另加30.00元，或满2999.00另加40.00元，即可在购物车换
购商品</em>
            </div>
```


```
computed:{
    sku () {
        /*
        将:
        {
            颜色 : 0,
            内存 : 1,
            机身存储 : 0
        }
        这样数据转换成0_1_0
        Object.values获取json对象的所有value值得到数据
        */
        const indexStr = Object.values(this.indexes).join('_');
        //数组的find函数: 通过条件查找数组内某一个函数
        //我们需要查询sku.indexes == indexStr的元素
        return this.skusInfo.find(sku => sku.indexes == indexStr);
    },
    images () {
        //先判断一下当前选中sku的images是否有内容
        //然后再通过,分割字符串.因为有可能一个sku有多个图片
        return this.sku.images ? this.sku.images.split(',') : [];
    }
},
```

3 页面静态化

3.1 问题分析

现在，我们的页面是通过Thymeleaf模板引擎渲染后返回到客户端

返回页面之前需要在后端查询数据，进行多次与mysql的交互.

而且每次请求数据就会与mysql交互,性能非常的差,如果并发量比较高的话

mysql和后台服务有可能会扛不住,后台服务我们可以做负载均衡,但是数据库没有办法做负载

即使做负载,成本也会非常的高,所以们使用页面静态化技术来解决这个问题

3.2 静态化

静态化是指把动态生成的HTML页面变为静态内容保存，以后用户的请求商品详情，直接访问静态页面，不再经过tomcat controler 的渲染。

而静态的HTML页面可以部署在nginx中，从而提高并发能力，减小tomcat压力。无需占用tomact并发数

3.3 如何实现页面静态化

目前，静态化页面都是通过模板引擎来生成，而后保存到nginx服务器来部署。常用的模板引擎比如：

- Freemarker
- Velocity
- Thymeleaf

我们之前就使用的Thymeleaf，来渲染html返回给用户。

Thymeleaf除了可以把渲染结果写入Response，也可以写到本地文件，从而实现静态化。

3.4 Thymeleaf实现静态化

先说下Thymeleaf中的几个概念：

- Context：运行上下文
- TemplateResolver：模板解析器
- TemplateEngine：模板引擎　processEngine-->JBPM(hibernate) , activity(mybatis) //请假 --> 请假申请 --> 组长审批 --> 经理审批 --> 人事审批 --> 财务审批 --> 老板审批

> Context

上下文： 用来保存模型数据，当模板引擎渲染时，可以从Context上下文中获取数据用于渲染。

当与SpringBoot结合使用时，我们放入Model的数据就会被处理到Context，作为模板渲染的数据使用。

> TemplateResolver

模板解析器：用来读取模板相关的配置，例如：模板存放的位置信息，模板文件名称，模板文件的类型等等。

当与SpringBoot结合时，TemplateResolver已经由其创建完成，并且各种配置也都有默认值，比如模板存放位置，其默认值就是：templates。比如模板文件类型，其默认值就是html。

> TemplateEngine

模板引擎：用来解析模板的引擎，需要使用到上下文、模板解析器。分别从两者中获取模板中需要的数据，模板文件。然后利用内置的语法规则解析，从而输出解析后的文件。来看下模板引起进行处理的函数：

```
templateEngine.process("模板名", context, writer);
```

三个参数：

- 模板的名称
- 上下文：里面包含模型数据
- writer：输出目的地的流

在输出时，我们可以指定输出的目的地，如果目的地是Response的流，那就是网络响应。如果目的地是本地文件，那就实现静态化了。

而在SpringBoot中已经自动配置了模板引擎，因此我们不需要关心这个。现在我们做静态化，就是把输出的目的地改成本地文件即可

3.5 实现

3.5.1 mingrui-shop-service-api

3.5.1.1 新建项目mingrui-shop-service-api-template

3.5.1.2 新建包com.baidu.shop

3.5.1.3 包下新建service和config包

3.5.1.4 service包下新建TemplateService

```
    @GetMapping(value = "template/createStaticHTMLTemplate")
    Result<JSONObject> createStaticHTMLTemplate(Integer spuId);

    @GetMapping(value = "template/initStaticHTMLTemplate")
    Result<JSONObject> initStaticHTMLTemplate();

    @GetMapping(value = "template/clearStaticHTMLTemplate")
    Result<JSONObject> clearStaticHTMLTemplate();

    @GetMapping(value = "template/deleteStaticHTMLTemplate")
    Result<JSONObject> deleteStaticHTMLTemplate(Integer spuId);
```

3.5.1.5 config包下新建MrSwagger2Config

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

/**
 * @ClassName MrSwagger2Config
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/17
 * @Version V1.0
 **/
@Configuration
@EnableSwagger2
public class MrSwagger2Config {

    @Bean
    public Docket createRestApi(){
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(this.apiInfo())
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.baidu"))
                .paths(PathSelectors.any())
                .build();
```

```
    }

    private ApiInfo apiInfo(){
        return new ApiInfoBuilder()
                //标题
                .title("明瑞SWAGGER2标题")
                //条款地址
                .termsOfServiceUrl("http://www.baidu.com")
                //联系方式-->有String参数的方法但是已经过时，所以不推荐使用
                .contact(new
Contact("shenyaqi","baidu.com","shenyaqiii@163.com"))
                //版本
                .version("v1.0")
                //项目描述
                .description("描述")
                //创建API基本信息
                .build();
    }
}
```

3.5.2 mingrui-shop-service-template

3.5.2.1 注释掉或者直接删掉controller和service下的所有注解

我们不用原来的那种方式实现，所以留着他们也没有什么用

```
 **/
//@Controller
//@RequestMapping(value = "item")
public class PageController {

    //@Autowired
    private PageService pageService;

    //@GetMapping(value = "{spuId}.html")
@   public String itemPage(@PathVariable(value = "spuId") Integer spuId, ModelMap map){

        Map<String,Object> goodsMap = pageService.getGoodsInfo(spuId);

        map.putAll(goodsMap);
        return "item";
    }
}
```

```
 * @Date 2020/9/11
 * @Version V1.0
 **/
//@Service
public class PageServiceImpl implements PageService {

    //@Autowired
    private GoodsFeign goodsFeign;

    //@Autowired
    private CategoryFeign categoryFeign;

    //@Autowired
    private SpecificationFeign specificationFeign;

   // @Autowired
    private BrandFeign brandFeign;

    @Override
    public Map<String, Object> getGoodsInfo(Integer spuId) {

        Map<String, Object> map = new HashMap<>();

        //spu信息
        SpuDTO spuDTO = new SpuDTO();
        spuDTO.setId(spuId);
```

这个方法暂时还留着,
一会直接复制粘贴代码

3.5.2.2 application.yml

```yaml
mrshop:
  static:
    html:
      path: E:\static-html\item #生成的html文件存储的路径,注意这个目录需要提前建好
```

3.5.2.3 impl包下新建TemplateServiceImpl

```
    @Autowired
    private TemplateEngine templateEngine;

    @Value(value = "${mrshop.static.html.path}")
    private String filePath;

    @Override
    public Result<JSONObject> createStaticHTMLTemplate(Integer spuId) {
        Map<String, Object> goodsInfo = new HashMap<>();
```

模板引擎

文件位置

```
    //创建上下文
    Context context = new Context();
    context.setVariables(goodsInfo);//设置数据,参数为map集合

    //创建要生成的文件
    File file = new File( pathname: filePath + spuId + ".html");
    //输出流
    PrintWriter printWriter = null;
    try {
        printWriter = new PrintWriter(file, csn: "UTF-8");
        //按照模板写入文件内容
        templateEngine.process( template: "item",context,printWriter);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } finally {
      if(printWriter != null){

          printWriter.close();//关流!!!!
      }
    }
}
return this.setResultSuccess();
```

```
import com.alibaba.fastjson.JSONObject;
import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
import com.baidu.shop.entity.*;
import com.baidu.shop.feign.BrandFeign;
import com.baidu.shop.feign.CategoryFeign;
import com.baidu.shop.feign.GoodsFeign;
import com.baidu.shop.feign.SpecificationFeign;
import com.baidu.shop.response.*;
import com.baidu.shop.service.TemplateService;
import com.baidu.shop.utils.BaiduBeanUtil;
import com.github.pagehelper.PageInfo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.RestController;
import org.thymeleaf.TemplateEngine;
import org.thymeleaf.context.Context;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

/**
 * @ClassName TemplateServiceImpl
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/15
 * @Version V1.0
 **/
@RestController
```

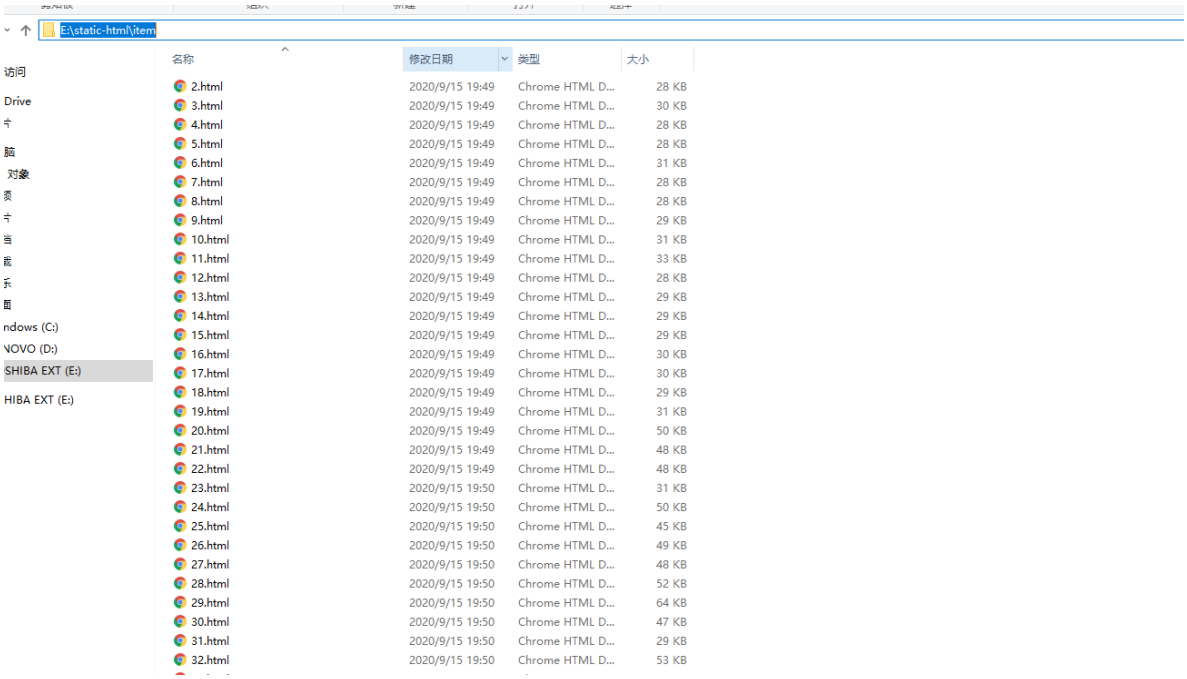
```
public class TemplateServiceImpl extends BaseApiService implements
TemplateService {


    @Autowired
    private GoodsFeign goodsFeign;

    @Autowired
    private CategoryFeign categoryFeign;

    @Autowired
    private SpecificationFeign specificationFeign;

    @Autowired
    private BrandFeign brandFeign;

    //注入静态化模版
    @Autowired
    private TemplateEngine templateEngine;

    //静态文件生成的路径
    @Value(value = "${mrshop.static.html.path}")
    private String staticHTMLPath;

    @Override
    public Result<JSONObject> createStaticHTMLTemplate(Integer spuId) {
        Map<String, Object> map = new HashMap<>();

        //spu信息
        SpuDTO spuDTO = new SpuDTO();
        spuDTO.setId(spuId);
        Result<List<SpuDTO>> spuResult = goodsFeign.getSpuInfo(spuDTO);
        if(spuResult.getCode() == 200){
            SpuDTO spuInfo = spuResult.getData().get(0);

            map.put("spuInfo",spuInfo);
            //spudetail信息
            Result<SpuDetailEntity> spuDetailResult =
goodsFeign.getSpuDetailBydSpu(spuId);
            if(spuDetailResult.getCode() == 200){
                SpuDetailEntity spuDetailInfo = spuDetailResult.getData();

                map.put("spuDetailInfo",spuDetailInfo);
            }
            //查询分类信息
            Result<List<CategoryEntity>> cateResult =
categoryFeign.getCateByIds(
                    String.join(
                            ","
                            , Arrays.asList(
                                    spuInfo.getCid1() + ""
                                    , spuInfo.getCid2() + ""
                                    , spuInfo.getCid3() + "")
                    )
            );

            if(cateResult.getCode() == 200){
                map.put("cateList",cateResult.getData());
```

```
            }
            //查询品牌信息
            BrandDTO brandDTO = new BrandDTO();
            brandDTO.setId(spuInfo.getBrandId());
            Result<PageInfo<BrandEntity>> brandResult =
brandFeign.getBrandInfo(brandDTO);

            if(brandResult.getCode() == 200){

                map.put("brandInfo",brandResult.getData().getList().get(0));
            }

            //规格组和规格参数
            SpecGroupDTO specGroupDTO = new SpecGroupDTO();
            specGroupDTO.setCid(spuInfo.getCid3());
            Result<List<SpecGroupEntity>> sepcGroupResult =
specificationFeign.getSepcGroupInfo(specGroupDTO);

            if (sepcGroupResult.getCode() == 200) {
                List<SpecGroupEntity> specGroupInfo = sepcGroupResult.getData();
                //规格组和规格参数
                List<SpecGroupDTO> groupsInParams =
specGroupInfo.stream().map(specGroup -> {

                    SpecGroupDTO sgd = BaiduBeanUtil.copyProperties(specGroup,
SpecGroupDTO.class);
                    //规格参数-通用参数
                    SpecParamDTO specParamDTO = new SpecParamDTO();
                    specParamDTO.setGroupId(specGroup.getId());
                    specParamDTO.setGeneric(1);
                    Result<List<SpecParamEntity>> specParamResult =
specificationFeign.getSpecParamInfo(specParamDTO);
                    if (specParamResult.getCode() == 200) {
                        sgd.setSpecParams(specParamResult.getData());
                    }
                    return sgd;
                }).collect(Collectors.toList());

                map.put("groupsInParams",groupsInParams);
            }

            //特有规格参数
            SpecParamDTO specParamDTO = new SpecParamDTO();
            specParamDTO.setCid(spuInfo.getCid3());
            specParamDTO.setGeneric(0);
            Result<List<SpecParamEntity>> specParamResult =
specificationFeign.getSpecParamInfo(specParamDTO);

            if(specParamResult.getCode() == 200) {
                //需要将数据转换为map方便页面操作!!!!!!!!!
                Map<Integer, String> specMap = new HashMap<>();
                specParamResult.getData().stream().forEach(spec ->
specMap.put(spec.getId(),spec.getName()));
                map.put("specParamMap",specMap);
            }

            //sku
            Result<List<SkuDTO>> skuResult = goodsFeign.getSkuBySpuId(spuId);
```

```
            if(skuResult.getCode() == 200){

                List<SkuDTO> skuList = skuResult.getData();
                map.put("skuList",skuList);
            }
        }

        //创建模板引擎上下文
        Context context = new Context();
        //将所有准备的数据放到模板中
        context.setVariables(map);

        //创建文件 param1:文件路径 param2:文件名称
        File file = new File(staticHTMLPath, spuId + ".html");
        //构建文件输出流
        PrintWriter writer = null;
        try {
            writer = new PrintWriter(file, "UTF-8");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        //根据模板生成静态文件
        //param1:模板名称 params2:模板上下文[上下文中包含了需要填充的数据],文件输出流
        templateEngine.process("item",context,writer);

        return this.setResultSuccess();
    }

    @Override
    public Result<JSONObject> initStaticHTMLTemplate() {

        //获取所有的spu信息,注意:应该写一个只获取id集合的接口,我只是为了省事
        Result<List<SpuDTO>> spuInfo = goodsFeign.getSpuInfo(new SpuDTO());
        if(spuInfo.getCode() == 200){

            List<SpuDTO> spuList = spuInfo.getData();

            spuList.stream().forEach(spu -> {
                this.createStaticHTMLTemplate(spu.getId());
            });
        }
        return this.setResultSuccess();
    }

        @Override
    public Result<JSONObject> clearStaticHTMLTemplate() {

        Result<List<SpuDTO>> spuInfo = goodsFeign.getSpuInfo(new SpuDTO());
        if(spuInfo.isSuccess()){
            spuInfo.getData().stream().forEach(spu -> {
                this.deleteStaticHTMLTemplate(spu.getId());
            });
        }
        return this.setResultSuccess();
    }
```

```
    @Override
    public Result<JSONObject> deleteStaticHTMLTemplate(Integer spuId) {

        File file = new File(staticHTMLPath + File.separator + spuId + "html");
        if(!file.delete()){
            return this.setResultError("文件删除失败");
        }

        return this.setResultSuccess();
    }
}
```

使用swagger测试看文件是否生成



此时文件已经生成了,但是浏览器请求详情页面的时候还是请求的我们的项目

所以我们需要修改一下nginx的配置

```
30    server {
31        listen       80;
32        server_name  www.mrshop.com;
33        #如果请求路径带item我们就让nginx帮我们把请求转发到8400端口
34        location /item {
35            root E:\static-html;
36        }
37        location / {
38            proxy_pass http://127.0.0.1:9002;
39            proxy_connect_timeout 600;
40            proxy_read_timeout 600;
41        }
42
43        error_page   500 502 503 504  /50x.html;
44        location = /50x.html {
45            root   html;
46        }
47    }
48
```

将所有请求包含item的请求转发到本地
注意:不需要加上item那一层文件夹
因为他只是将包含item的请求转发到了本地的文件夹,他的请求中还是带item的
我们并没有重写url

```
        location /item {
            root E:\static-html;
        }
```

打开浏览器进行测试,展示详情页的时候几乎是秒开,大大加快了查询效率

至此,页面静态化完成,但是service中的方法太大了,所以我们还是需要拆一下方法的....

```
import com.alibaba.fastjson.JSONObject;
import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
import com.baidu.shop.entity.*;
import com.baidu.shop.feign.BrandFeign;
import com.baidu.shop.feign.CategoryFeign;
import com.baidu.shop.feign.GoodsFeign;
import com.baidu.shop.feign.SpecificationFeign;
import com.baidu.shop.response.*;
import com.baidu.shop.service.TemplateService;
import com.baidu.shop.utils.BaiduBeanUtil;
import com.github.pagehelper.PageInfo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.RestController;
import org.thymeleaf.TemplateEngine;
import org.thymeleaf.context.Context;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

/**
 * @ClassName TemplateServiceImpl
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/15
 * @Version V1.0
 **/
@RestController
public class TemplateServiceImpl extends BaseApiService implements
TemplateService {


    @Autowired
    private GoodsFeign goodsFeign;

    @Autowired
    private CategoryFeign categoryFeign;

    @Autowired
    private SpecificationFeign specificationFeign;

    @Autowired
    private BrandFeign brandFeign;

    //注入静态化模版
    @Autowired
    private TemplateEngine templateEngine;
```

```
    //静态文件生成的路径
    @Value(value = "${mrshop.static.html.path}")
    private String staticHTMLPath;

    @Override
    public Result<JSONObject> createStaticHTMLTemplate(Integer spuId) {
        Map<String, Object> map = new HashMap<>();
        //spu信息
        SpuDTO spuDTO = new SpuDTO();
        spuDTO.setId(spuId);
        Result<List<SpuDTO>> spuResult = goodsFeign.getSpuInfo(spuDTO);
        if(spuResult.getCode() == 200){
            SpuDTO spuInfo = spuResult.getData().get(0);

            map.put("spuInfo",spuInfo);
            //spudetail信息
            SpuDetailEntity spuDetailInfo = this.getSpuDetailInfo(spuId);
            map.put("spuDetailInfo",spuDetailInfo);
            //查询分类信息
            List<CategoryEntity> cateList = this.getCateList(spuInfo);
            map.put("cateList",cateList);

            //查询品牌信息
            BrandEntity brandInfo = this.getBrandInfo(spuInfo);
            map.put("brandInfo",brandInfo);

            //规格组和规格参数
            List<SpecGroupDTO> groupsInParams = this.getGroupsInParams(spuInfo);
            map.put("groupsInParams",groupsInParams);

            //特有规格参数
            Map<Integer, String> specParamMap = this.getSpecParamMap(spuInfo);
            map.put("specParamMap",specParamMap);
            //sku
            List<SkuDTO> skuList = this.getSkuList(spuId);
            map.put("skuList",skuList);
        }

        this.createStaticHTML(map,spuId);

        return this.setResultSuccess();
    }

    private SpuDetailEntity getSpuDetailInfo(Integer spuId){
        Result<SpuDetailEntity> spuDetailResult =
goodsFeign.getSpuDetailBydSpu(spuId);
        if(spuDetailResult.getCode() == 200){
            SpuDetailEntity spuDetailInfo = spuDetailResult.getData();
            return spuDetailInfo;
        }
        return null;
    }

    private List<CategoryEntity> getCateList(SpuDTO spuInfo){
        Result<List<CategoryEntity>> cateResult = categoryFeign.getCateByIds(
                String.join(
                        ","
```

```
                        , Arrays.asList(
                                spuInfo.getCid1() + ""
                                , spuInfo.getCid2() + ""
                                , spuInfo.getCid3() + "")
                )
        );

        if(cateResult.getCode() == 200){
            return cateResult.getData();
        }
        return null;
    }

    private BrandEntity getBrandInfo(SpuDTO spuInfo){
        BrandDTO brandDTO = new BrandDTO();
        brandDTO.setId(spuInfo.getBrandId());
        Result<PageInfo<BrandEntity>> brandResult =
brandFeign.getBrandInfo(brandDTO);

        if(brandResult.getCode() == 200){

            return brandResult.getData().getList().get(0);
        }
        return null;
    }

    private List<SpecGroupDTO> getGroupsInParams(SpuDTO spuInfo){
        SpecGroupDTO specGroupDTO = new SpecGroupDTO();
        specGroupDTO.setCid(spuInfo.getCid3());
        Result<List<SpecGroupEntity>> sepcGroupResult =
specificationFeign.getSepcGroupInfo(specGroupDTO);

        if (sepcGroupResult.getCode() == 200) {
            List<SpecGroupEntity> specGroupInfo = sepcGroupResult.getData();
            //规格组和规格参数
            List<SpecGroupDTO> groupsInParams =
specGroupInfo.stream().map(specGroup -> {

                SpecGroupDTO sgd = BaiduBeanUtil.copyProperties(specGroup,
SpecGroupDTO.class);
                //规格参数-通用参数
                SpecParamDTO specParamDTO = new SpecParamDTO();
                specParamDTO.setGroupId(specGroup.getId());
                specParamDTO.setGeneric(1);
                Result<List<SpecParamEntity>> specParamResult =
specificationFeign.getSpecParamInfo(specParamDTO);
                if (specParamResult.getCode() == 200) {
                    sgd.setSpecParams(specParamResult.getData());
                }
                return sgd;
            }).collect(Collectors.toList());

            return groupsInParams;
        }
        return null;
    }

    private Map<Integer, String> getSpecParamMap(SpuDTO spuInfo){
```

```
        SpecParamDTO specParamDTO = new SpecParamDTO();
        specParamDTO.setCid(spuInfo.getCid3());
        specParamDTO.setGeneric(0);
        Result<List<SpecParamEntity>> specParamResult =
specificationFeign.getSpecParamInfo(specParamDTO);

        if(specParamResult.getCode() == 200) {
            //需要将数据转换为map方便页面操作!!!!!!!!!
            Map<Integer, String> specMap = new HashMap<>();
            specParamResult.getData().stream().forEach(spec ->
specMap.put(spec.getId(),spec.getName()));
            return specMap;
        }
        return null;
    }

    private List<SkuDTO> getSkuList(Integer spuId){
        Result<List<SkuDTO>> skuResult = goodsFeign.getSkuBySpuId(spuId);

        if(skuResult.getCode() == 200){

            List<SkuDTO> skuList = skuResult.getData();
            return skuList;
        }
        return null;
    }

    private void createStaticHTML(Map<String, Object> map,Integer spuId){
        //创建模板引擎上下文
        Context context = new Context();
        //将所有准备的数据放到模板中
        context.setVariables(map);

        //创建文件 param1:文件路径 param2:文件名称
        File file = new File(staticHTMLPath, spuId + ".html");
        //构建文件输出流
        PrintWriter writer = null;
        try {
            writer = new PrintWriter(file, "UTF-8");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        //根据模板生成静态文件
        //param1:模板名称 params2:模板上下文[上下文中包含了需要填充的数据],文件输出流
        templateEngine.process("item",context,writer);
    }

    @Override
    public Result<JSONObject> initStaticHTMLTemplate() {

        //获取所有的spu信息,注意:应该写一个只获取id集合的接口,我只是为了省事
        Result<List<SpuDTO>> spuInfo = goodsFeign.getSpuInfo(new SpuDTO());
        if(spuInfo.getCode() == 200){

            List<SpuDTO> spuList = spuInfo.getData();
```

```
            spuList.stream().forEach(spu -> {
                this.createStaticHTMLTemplate(spu.getId());
            });
        }
        return this.setResultSuccess();
    }

    @Override
    public Result<JSONObject> clearStaticHTMLTemplate() {

        Result<List<SpuDTO>> spuInfo = goodsFeign.getSpuInfo(new SpuDTO());
        if(spuInfo.isSuccess()){
            spuInfo.getData().stream().forEach(spu -> {
                this.deleteStaticHTMLTemplate(spu.getId());
            });
        }
        return this.setResultSuccess();
    }

    @Override
    public Result<JSONObject> deleteStaticHTMLTemplate(Integer spuId) {

        File file = new File(staticHTMLPath + File.separator + spuId + ".html");
        if(!file.delete()){
            return this.setResultError("文件删除失败");
        }

        return this.setResultSuccess();
    }
}
```