

商品详情页

版本: V 1.0

www.atguigu.com

一 业务介绍

商品详情页，简单说就是以购物者的角度展现一个 sku 的详情信息。

这个页面不同于传统的 crud 的详情页，使用者并不是管理员，需要对信息进行查删改查，取而代之的是点击购买、放入购物车、切换颜色等等。

另外一个特点就是该页面的高访问量，虽然只是一个查询操作，但是由于频繁的访问所以我们必须对其性能进行最大程度的优化。

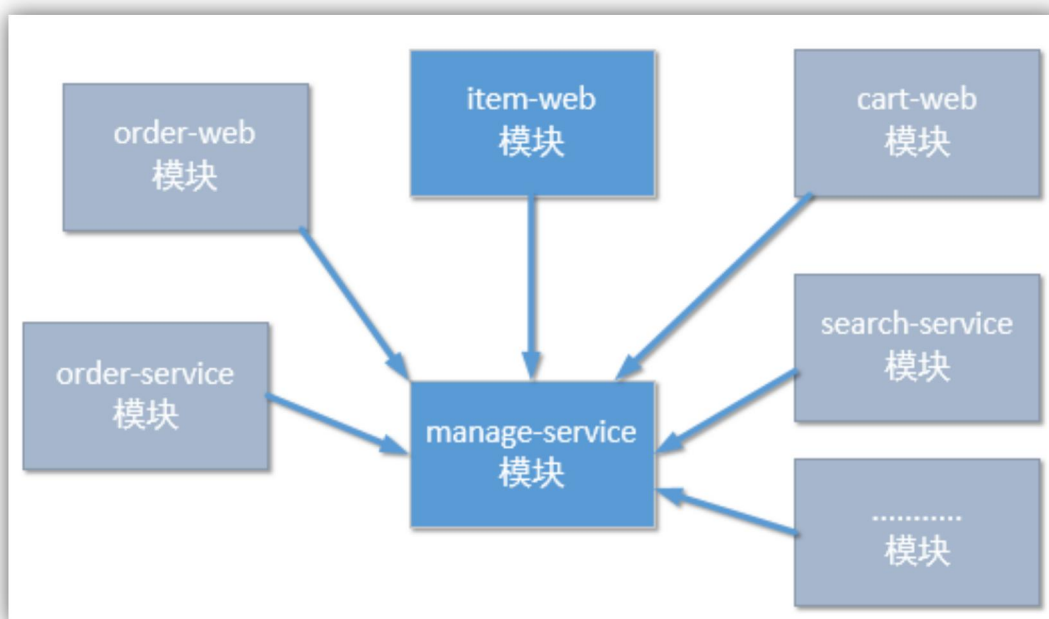
二 难点分析

1 光从功能角度上来说，并没有太多难点，唯一实现起来麻烦的就是用户对于不同销售属性的切换操作。

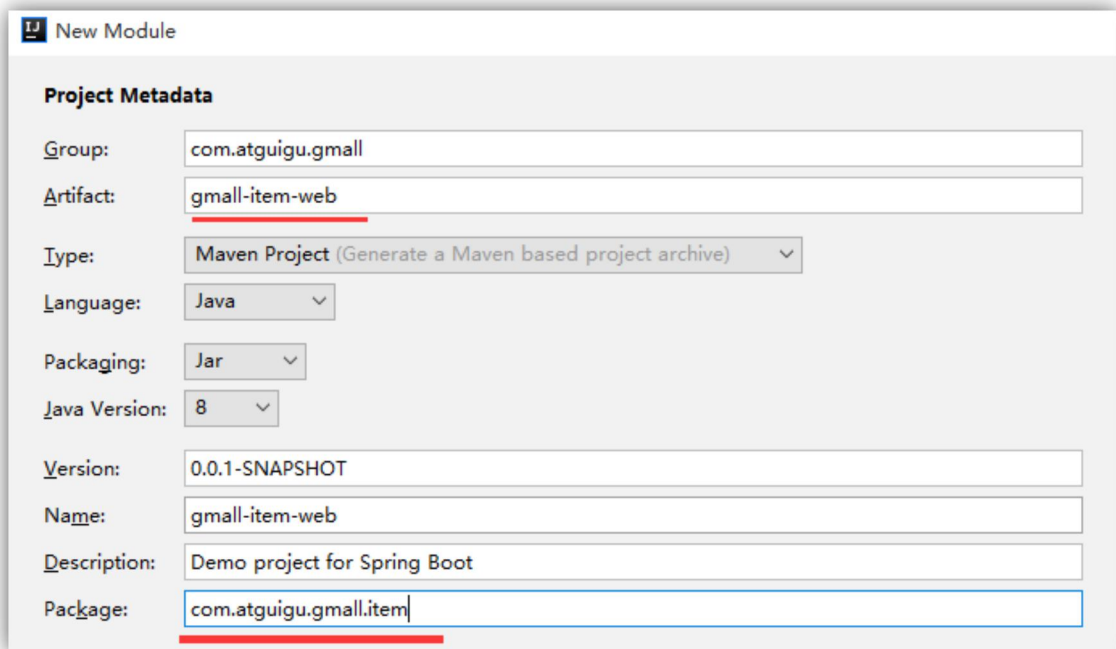


2 从性能角度来看，需要最大程度的提升页面的访问速度。

三 功能开发



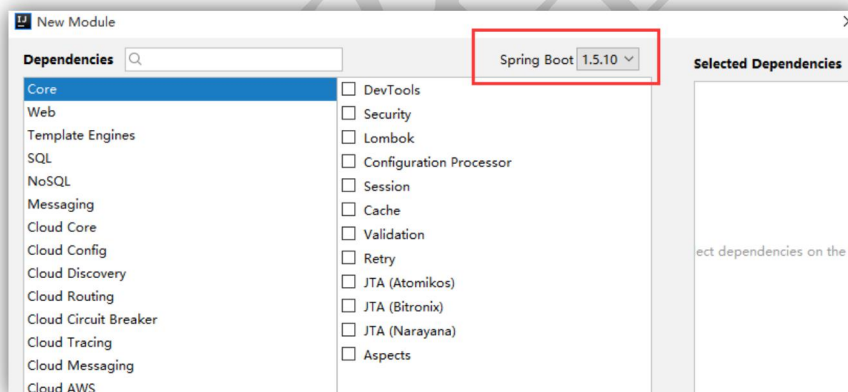
详情页功能，只增加一个 web 模块，后台调用商品管理的模块 `manage-service`
`item-web` 模块负责前端的页面渲染和控制层(controller)。



The 'New Module' dialog box shows the 'Project Metadata' section. The fields are filled as follows:

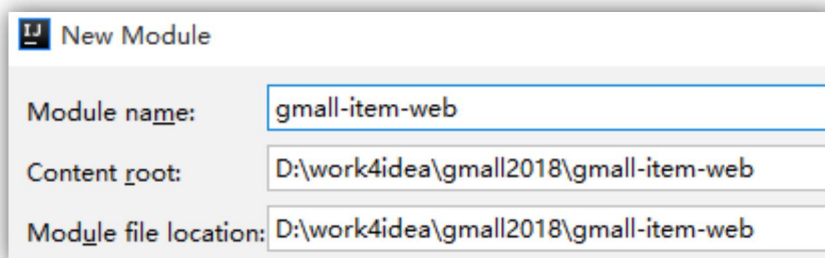
- Group: com.atguigu.gmall
- Artifact: gmall-item-web
- Type: Maven Project (Generate a Maven based project archive)
- Language: Java
- Packaging: Jar
- Java Version: 8
- Version: 0.0.1-SNAPSHOT
- Name: gmall-item-web
- Description: Demo project for Spring Boot
- Package: com.atguigu.gmall.item

依赖包全都不选，SpringBoot 版本选 1.5.10



The 'New Module' dialog box shows the 'Dependencies' section. The 'Spring Boot' version is selected as 1.5.10. The 'Selected Dependencies' section is empty.

Dependencies	Selected Dependencies
<input type="checkbox"/> Core	
<input type="checkbox"/> Web	
<input type="checkbox"/> Template Engines	
<input type="checkbox"/> SQL	
<input type="checkbox"/> NoSQL	
<input type="checkbox"/> Messaging	
<input type="checkbox"/> Cloud Core	
<input type="checkbox"/> Cloud Config	
<input type="checkbox"/> Cloud Discovery	
<input type="checkbox"/> Cloud Routing	
<input type="checkbox"/> Cloud Circuit Breaker	
<input type="checkbox"/> Cloud Tracing	
<input type="checkbox"/> Cloud Messaging	
<input type="checkbox"/> Cloud AWS	
<input type="checkbox"/> DevTools	
<input type="checkbox"/> Security	
<input type="checkbox"/> Lombok	
<input type="checkbox"/> Configuration Processor	
<input type="checkbox"/> Session	
<input type="checkbox"/> Cache	
<input type="checkbox"/> Validation	
<input type="checkbox"/> Retry	
<input type="checkbox"/> JTA (Atomikos)	
<input type="checkbox"/> JTA (Bitronix)	
<input type="checkbox"/> JTA (Narayana)	
<input type="checkbox"/> Aspects	



The 'New Module' dialog box shows the 'Module Information' section. The fields are filled as follows:

- Module name: gmall-item-web
- Content root: D:\work4idea\gmall2018\gmall-item-web
- Module file location: D:\work4idea\gmall2018\gmall-item-web

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.atguigu.gmall</groupId>
  <artifactId>gmall-item-web</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>gmall-item-web</name>
  <description>Demo project for Spring Boot</description>

  <parent>
    <groupId>com.atguigu.gmall</groupId>
    <artifactId>gmall-parent</artifactId>
    <version>1.0-SNAPSHOT</version>
  </parent>

  <dependencies>

    <dependency>
      <groupId>com.atguigu.gmall</groupId>
      <artifactId>gmall-interface</artifactId>
      <version>1.0-SNAPSHOT</version>
    </dependency>

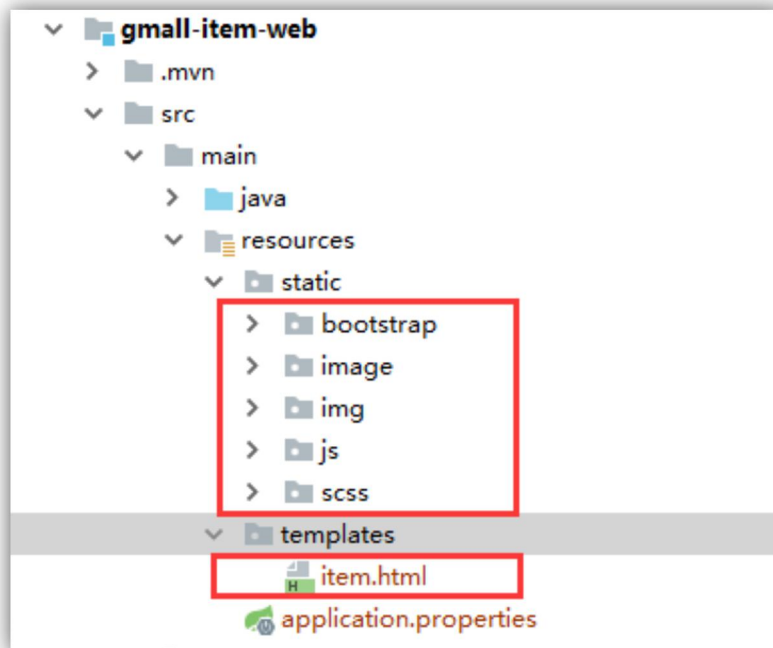
    <dependency>
      <groupId>com.atguigu.gmall</groupId>
      <artifactId>gmall-web-util</artifactId>
      <version>1.0-SNAPSHOT</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>

</project>
```

搭建完成后。

首先导入前端页面



静态页资源全部拷贝到 static 目录中，如果没有该目录请手工创建
动态的 html 文件拷贝到 templates 目录中

编写 Controller 类（入口方法）

```
@Controller
public class ItemController {

    @RequestMapping("/{skuId}.html")
    public String getSkuInfo(@PathVariable("skuId") String skuId){
        return "item";
    }

}
```

application.properties

```
server.port=8084

spring.thymeleaf.cache=false

spring.thymeleaf.mode=LEGACYHTML5
```

修改 item.html 改为绝对路径

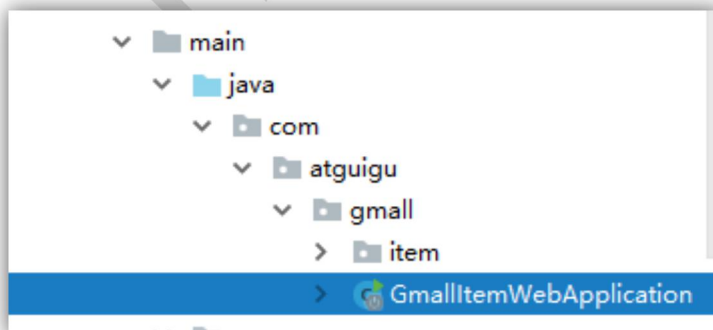
把 item.html 所有 `src="/` 替换成 `src=""`

`<link>` 标签的路径也改为 `/` 开头

```
<head>
  <meta charset="UTF-8">
  <title></title>
  <link rel="stylesheet" type="text/css" href="/scss/shop.css" />
  <link rel="stylesheet" type="text/css" href="/scss/main.css"/>
  <link rel="stylesheet" href="/scss/header.css" />
  <link rel="stylesheet" type="text/css"
href="/bootstrap/css/bootstrap.css"/>
</head>
```

把启动类 `GmallItemWebApplication` 提到和 `item` 平级的目录中。

或者增加 `@ComponentScan(basePackages = "com.atguigu.gmall")`



然后启动服务

启动测试



可以看到商品详情页的静态页面。

后台实现在 `gmall-item-web` 模块中

增加 `ItemController`

```
@RequestMapping("/{skuId}.html")
public String getSkuInfo(@PathVariable("skuId") String skuId, Model model){
    SkuInfo skuInfo = manageService.getSkuInfo(skuId);
    model.addAttribute("skuInfo", skuInfo);
}
```

`gmall-manage-service` 中增加

后台实现类

```
public SkuInfo getSkuInfo(String skuld){

    SkuInfo skuInfo = skuInfoMapper.selectByPrimaryKey(skuld);
    if(skuInfo==null){
        return null;
    }
    SkuImage skuImage=new SkuImage();
    skuImage.setSkuld(skuld);
    List<SkuImage> skuImageList = skuImageMapper.select(skuImage);
    skuInfo.setSkuImageList(skuImageList);
}
```

7

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载, 可访问百度: [尚硅谷官网](#)

```
SkuSaleAttrValue skuSaleAttrValue=new SkuSaleAttrValue();
skuSaleAttrValue.setSkuld(skuld);
List<SkuSaleAttrValue> skuSaleAttrValueList = skuSaleAttrValueMapper.select(skuSaleAttrValue);
skuInfo.setSkuSaleAttrValueList(skuSaleAttrValueList);

return skuInfo;
}
```

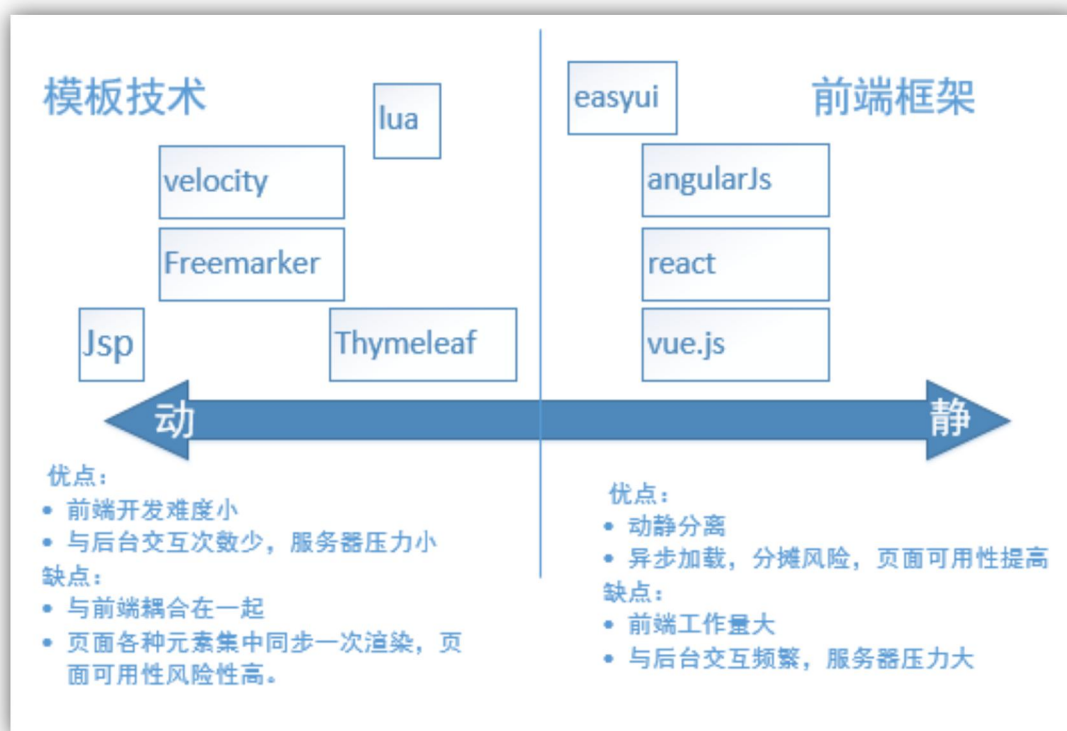
四 Thymeleaf

1 模板技术

把页面中的静态数据替换成从后台数据库中的数据。这种操作用 jsp 就可以实现。但是 Springboot 的架构不推荐使用 Jsp，而且支持也不好，所以如果你是用 springboot 的话，一般使用 Freemarker 或者 Thymeleaf。

而官方也是推荐使用 Thymeleaf。

关于与前端有关的技术的比较



2 Thymeleaf 简介

Thymeleaf 的主要目标是提供一个优雅和高度可维护的创建模板的方式。为了实现这一点, 它建立在自然模板的概念上, 将其逻辑注入到模板文件中, 不会影响模板被用作设计原型。这改善了设计的沟通, 弥合了设计和开发团队之间的差距。

比 Jsp 和 Freemarker 的优势, 一般的模板技术都会在页面加各种表达式、标签甚至是 java 代码, 而这些都是必须要经过后台服务器的渲染才能打开。

但如果前端开发人员做页面调整, 双击打开某个 jsp 或者 ftl 来查看效果, 基本上是打不开的。

那么 Thymeleaf 的优势就出来了, 因为 Thymeleaf 没有使用自定义的标签或语法, 所有的模板语言都是扩展了标准 H5 标签的属性

比如

```
<div th:text="${item.skuName}" ></div>
```

它的效果和 Jsp 中的

```
<div>${item.skuName}</div>
```

渲染后效果一样，但是如果你直接用浏览器打开页面文件，H5 会把 th:text 这种不认识的属性忽略掉。效果就和<div></div> 没有区别，所以对于前端调页面影响更新。以上只是举了一个例子，如果是循环、分支的判断效果更明显。

3 快速入门:

3.1 所有头文件

就行 Jsp 的<%@Page %>一样，Thymeleaf 的也要引入标签规范。不加这个虽然不影响程序运行，但是你的 idea 会认不出标签，不方便开发。

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="http://www.thymeleaf.org">
```

3.2 取出请求域中的值，即取得 request.attribute 中的值

```
<p th:text="${hello}">打底值</p>
```

3.3 循环

```
<table border="1">
  <tr th:each="skulImage:${skuInfo.skulImageList}">
    <td th:text="${skulImage.id}">
    </td>
    <td th:text="${skulImage.imgName}">
    </td>
  </tr>
```

```
</table>
```

3.4 判断

```
<td th:text="{skulImage.id}=='10'?'是 10':'不是 10'">
</td>
<td th:if="{skulImage.id}=='10'">123456</td>
```

3.5 取 session 中的属性

```
<div th:text="{session.userName}"></div>
```

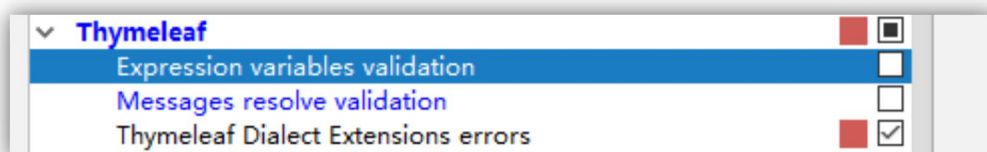
3.6 引用内嵌页

```
<div th:include="itemInner"/>
```

如果 idea 编辑器质疑你页面中的元素是否存在

```
<body>
<div th:text="{skuInfo.skuName}"></div>
<div th:text="{session.skuName}"></div>
```

通常不准确，可以去掉验证，在 settings->editor->Inspections 中关掉验证。



五 开发详情页功能

名称:

```
<div class="box-name" th:text="${skuInfo.skuName}">
    华为 HUAWEI Mate 10 6GB+128GB 亮黑色 移动联通电信 4G 手机 双卡双待
</div>
```

价格:

```
<span th:text="${#numbers.formatDecimal(skuInfo.price,1,2)}">4499.00</span>
```

重量:

```
<li th:text="${#numbers.formatDecimal(skuInfo.weight,1,2)}+' kg'"></li>
```

图片

```
<div class="box-lh-one">
    <ul>
        <li th:each="skuImage:${skuInfo.skuImageList}">
            
        </li>
    </ul>
</div>
```

`numbers.formatDecimal(<值>,<小数点左边的占位>,<小数点右边的保留位>)`

六 销售属性的处理

1 思路:

- 1、查出该商品的 spu 的所有销售属性和属性值
- 2、标识出本商品对应的销售属性
- 3、点击其他销售属性值的组合，跳转到另外的 sku 页面

2 查询出 sku 对应 spu 的销售属性

第 1、2 条通过此 sql 实现

```
SELECT sa.id ,sa.spu_id, sa.sale_attr_name,sa.sale_attr_id,
sv.id sale_attr_value_id,
sv.sale_attr_value_name,
skv.sku_id,
IF(skv.sku_id IS NOT NULL,1,0) is_check
FROM spu_sale_attr sa
INNER JOIN spu_sale_attr_value sv ON sa.spu_id=sv.spu_id AND sa.sale_attr_id=sv.sale_attr_id
LEFT JOIN sku_sale_attr_value skv ON skv.sale_attr_id= sa.sale_attr_id AND skv.sale_attr_value_id=sv.id
AND skv.sku_id=10
WHERE sa.spu_id=24
ORDER BY sv.sale_attr_id,sv.id
```

此 sql 列出所有该 spu 的销售属性和属性值，并关联某 skuid 如果能关联上 is_check 设为 1，否则设为 0。

页面开发部分

```
<div class="box-attr-2 clear" th:each="spuSaleAttr:${spuSaleAttrListCheckBySku}">
  <dl>
    <dt th:text="${spuSaleAttr.saleAttrName}">选择属性</dt>
    <dd th:class="({saleAttrValue.isCheck}=='1')?'redborder':"
th:each="saleAttrValue:${spuSaleAttr.spuSaleAttrValueList}">
      <div th:value="${saleAttrValue.id}" th:text="${saleAttrValue.saleAttrValueName}">
        属性值
      </div>
    </dd>
  </dl>
</div>
```

其中 th:class 的设置，redborder 是一个自定义的样式类，主要是边框设红。如果 isCheck=1 标识当前这个 sku 的所拥有的属性值，所以锁定为红边框。



需要增加修改的代码文件清单

13

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载，可访问百度：[尚硅谷官网](#)

增 加 SpuSaleAttrMapper.xml 的方法	selectSpuSaleAttrListCheckBySku 注意双参数的处理
增加 SpuSaleAttrMapper 的方法	
增加 ManageServiceImpl 的方法	getSpuSaleAttrListCheckBySku
增加 ManageService 的方法	getSpuSaleAttrListCheckBySku
修改 SpuSaleAttrValue	增加 isCheck 属性

代码见代码清单

增加 ManageServiceImpl 的方法	getSpuSaleAttrListCheckBySku
<pre> @Override public List<SpuSaleAttr> getSpuSaleAttrListCheckBySku(String skuld,String spuld){ List<SpuSaleAttr> spuSaleAttrList = spuSaleAttrMapper.selectSpuSaleAttrListCheckBySku(Long.parseLong(skuld),Long.parseLong(spuld)); return spuSaleAttrList; } </pre>	

增加 SpuSaleAttrMapper 的方法
public List<SpuSaleAttr> selectSpuSaleAttrListCheckBySku(long skuld,long spuld);

增 加 SpuSaleAttrMapper.xml 的方法	selectSpuSaleAttrListCheckBySku 注意双参数的处理
<pre> <select id ="selectSpuSaleAttrListCheckBySku" resultMap="spuSaleAttrMap"> SELECT sa.id ,sa.spu_id, sa.sale_attr_name,sa.sale_attr_id, sv.id sale_attr_value_id, sv.sale_attr_value_name, skv.sku_id, IF(skv.sku_id IS NOT NULL,1,0) is_check FROM spu_sale_attr sa INNER JOIN spu_sale_attr_value sv ON sa.spu_id=sv.spu_id AND sa.sale_attr_id=sv.sale_attr_id LEFT JOIN sku_sale_attr_value skv ON skv.sale_attr_id= sa.sale_attr_id AND skv.sale_attr_value_id=sv.id AND skv.sku_id=#{arg0} WHERE sa.spu_id=#{arg1} ORDER BY sv.sale_attr_id,sv.id </select> </pre>	

修改 SpuSaleAttrValue	增加 isCheck 属性
---------------------	---------------

修改 itemController	调用 getSpuSaleAttrListCheckBySku
<pre>List<SpuSaleAttr> spuSaleAttrListCheckBySku = manageService.getSpuSaleAttrListCheckBySku(skuInfo.getId(), skuInfo.getSpuId()); model.addAttribute("spuSaleAttrListCheckBySku", spuSaleAttrListCheckBySku);</pre>	

修改 item.html	增加渲染的代码
<pre><div class="box-attr-2 clear" th:each="spuSaleAttr:\${spuSaleAttrListCheckBySku}"> <dl> <dt th:text="\${spuSaleAttr.saleAttrName}">选择颜色</dt> <dd th:class="\${saleAttrValue.isCheck=='1'}?'redborder':'"' th:each="saleAttrValue:\${spuSaleAttr.spuSaleAttrValueList}"> <div th:value="\${saleAttrValue.id}" th:text="\${saleAttrValue.saleAttrValueName}"> 摩卡金 </div> </dd> </dl> </div></pre>	

3 点击其他销售属性值的组合，跳转到另外的 sku 页面

实现思路：

1、从页面中获得得所有选中的销售属性进行组合比如：

“属性值 1|属性值 2|属性值 3” 用这个字符串匹配一个对照表，来获得 skuld。并进行跳转，或者告知无货。

2、后台要生成一个“属性值 1|属性值 2|属性值 3: skuld”的一个 json 串以提供页面进行匹配。如 `valuesSku:{"46|50":"10","47|50":"13","48|49":"12","47|49":"11"}`

3、需要从后台数据库查询出该 spu 下的所有 skuld 和属性值关联关系。然后加工成如上的

Json 串。

实现：

skuSaleAttrValueMapper.xml 中 sql

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper SYSTEM "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.atguigu.gmall.manage.mapper.SkuSaleAttrValueMapper">
    <select id="selectSkuSaleAttrValueListBySpu" parameterType="long" resultMap="skuSaleAttrValueMap">
        SELECT  skv.*
        FROM sku_sale_attr_value skv
        INNER JOIN sku_info sk ON skv.sku_id=sk.id
        where  sk.spu_id=#{spuld}
        ORDER BY skv.sku_id , skv.sale_attr_id
    </select>
    <resultMap id="skuSaleAttrValueMap" type="com.atguigu.gmall.bean.SkuSaleAttrValue"
    autoMapping="true">
        <result property="id" column="id" ></result>
    </resultMap>
</mapper>
```

要注意排序，方便后面整理。

实现类很简单：

```
public List<SkuSaleAttrValue> getSkuSaleAttrValueListBySpu(String spuld){
    List<SkuSaleAttrValue> skuSaleAttrValueList =
    skuSaleAttrValueMapper.selectSkuSaleAttrValueListBySpu(Long.parseLong(spuld));
    return skuSaleAttrValueList;
}
```

难点是整理成咱们要求的 json 串，这个写在 controller 类。

```
List<SkuSaleAttrValue> skuSaleAttrValueListBySpu =
manageService.getSkuSaleAttrValueListBySpu(skulInfo.getSpuld());

//把列表转换成 valueid1|valueid2|valueid3 : skuld 的 哈希表 用于在页面中定位查询
String valueIdsKey="";

Map<String,String> valuesSkuMap=new HashMap<>();

for (int i = 0; i < skuSaleAttrValueListBySpu.size(); i++) {
    SkuSaleAttrValue skuSaleAttrValue = skuSaleAttrValueListBySpu.get(i);
    if(valueIdsKey.length() != 0){
        valueIdsKey = valueIdsKey + "|";
    }
    valueIdsKey = valueIdsKey + skuSaleAttrValue.getSaleAttrValueId();

    if((i+1) ==
```



```
skuSaleAttrValueListBySpu.size() || !skuSaleAttrValue.getSkuld().equals(skuSaleAttrValueListBySpu.get(i+1).getSkuld()) ) {  
  
    valuesSkuMap.put(valueIdsKey,skuSaleAttrValue.getSkuld());  
    valueIdsKey="";  
}  
  
}  
  
//把 map 变成 json 串  
String valuesSkuJson = JSON.toJSONString(valuesSkuMap);  
  
model.addAttribute("valuesSkuJson",valuesSkuJson);
```

在 item 中增加隐藏域

```
<input id="valuesSku" type="hidden" th:value= " ${valuesSkuJson}" />  
<input id="skuId" type="hidden" th:value= " ${skuInfo.id}" />
```

valuesSku 存储属性值与 skuld 的对照 json

skuld 存放当前商品的 skuld.

修改页面的 js

```
//红边框  
$($(".box-attr-2 dd").click(function() {  
    $(this).addClass("redborder").siblings("dd").removeClass("redborder");  
    switchSkuld();  
}))
```

增加点击销售属性时触发切换 sku 方法

```
function switchSkuld() {  
    var redborderDivs = $(".redborder div");  
    var valueIdkeys="";  
    for(i=0;i<redborderDivs.length;i++){  
        var redborderDiv= redborderDivs.eq(i);  
        var attrValueId = redborderDiv.attr("value");  
        if(i>0){  
            valueIdkeys+="|";  
        }  
        valueIdkeys+=attrValueId;  
    }  
    console.log("valueIdkeys:"+valueIdkeys);  
    var valueIdSkuJson= $("#valueIdSkuJson").val();  
    console.log("valueIdSkuJson:"+valueIdSkuJson);  
    var skuSelfId= $("#skuld").val();
```

```
var valueIdSku = JSON.parse(valueIdSkuJson);
var skuldTarget = valueIdSku[valueIdkeys];
if(!skuldTarget){
    $("#cartBtn").attr("class", "box-btns-two-off");
    $("#cartBtn").attr("canClick", '0');
    $("#cartBtn").css("cursor", 'not-allowed' );
}else{
    if(skuSelfId!=skuldTarget){
        window.location.href="/"+skuldTarget+".html";
    }else{
        $("#cartBtn").attr("class", "box-btns-two");
        $("#cartBtn").attr("canClick", '1');
        $("#cartBtn").css("cursor", 'pointer' );
    }
}
}
}
}
```

最后测试：



七 性能优化

1 思路：

虽然咱们实现了页面需要的功能，但是考虑到该页面是被用户高频访问的，所以性能必须进行尽可能的优化。

一般一个系统最大的性能瓶颈，就是数据库的 io 操作。从数据库入手也是调优性价比最高的切入点。

一般分为两个层面，一是提高数据库 sql 本身的性能，二是尽量避免直接查询数据库。

提高数据库本身的性能首先是优化 sql，包括：使用索引，减少不必要的大表关联次数，控制查询字段的行数 and 列数。另外当数据量巨大是可以考虑分库分表，以减轻单点压力。

这部分知识在 mysql 高级已有讲解，这里大家可以以详情页中的 sql 作为练习，尝试进行优化，这里不做赘述。

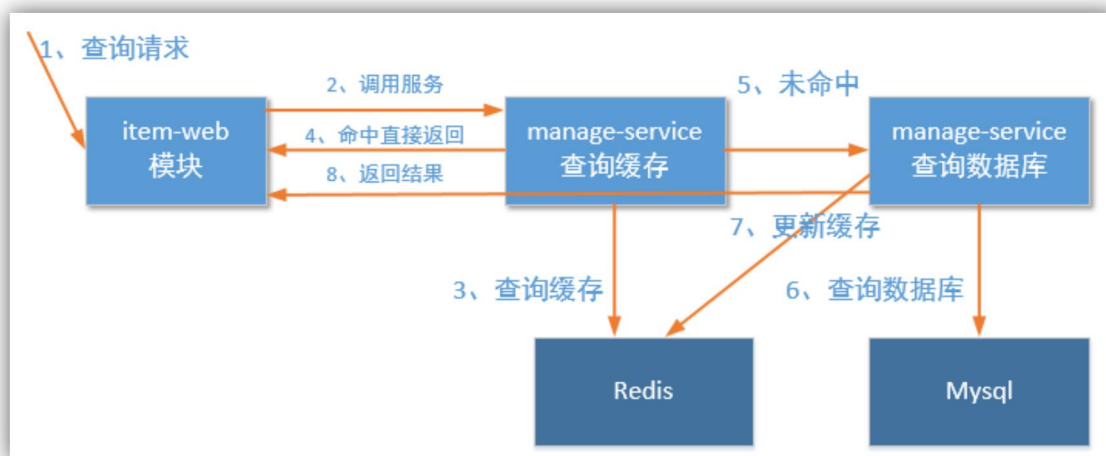
重点要讲的是另外一个层面：尽量避免直接查询数据库。

解决办法就是：**缓存**

缓存可以理解是数据库的一道保护伞，任何请求只要能在缓存中命中，都不会直接访问数据库。而缓存的处理性能是数据库 10-100 倍。

咱们就用 Redis 作为缓存系统进行优化。

结构图：



安装 Redis : 略

2 整合 redis 到大工程中。

由于 redis 作为缓存数据库，要被多个项目使用，所以要制作一个通用的工具类，方便工程中的各个模块使用。

而主要使用 redis 的模块，都是后台服务的模块，xxx-service 工程。所以咱们把 redis 的工具类放到 service-util 模块中，这样所有的后台服务模块都可以使用 redis。

首先引入依赖包

```

<!-- https://mvnrepository.com/artifact/redis.clients/jedis -->
<dependency>
  <groupId>redis.clients</groupId>
  <artifactId>jedis</artifactId>
  <version>2.9.0</version>
</dependency>

```

分别按照之前的方式放到 parent 模块和 service-util 的 pom 文件中。

然后在 service-util 中创建两个类 RedisConfig 和 RedisUtil

RedisConfig 负责在 spring 容器启动时自动注入，而 RedisUtil 就是被注入的工具类以供其他模块调用。

RedisUtil

```
public class RedisUtil {  
  
    private JedisPool jedisPool;  
  
    public void initPool(String host,int port ,int database){  
        JedisPoolConfig poolConfig = new JedisPoolConfig();  
        poolConfig.setMaxTotal(200);  
        poolConfig.setMaxIdle(30);  
        poolConfig.setBlockWhenExhausted(true);  
        poolConfig.setMaxWaitMillis(10*1000);  
        poolConfig.setTestOnBorrow(true);  
        jedisPool=new JedisPool(poolConfig,host,port,20*1000);  
    }  
  
    public Jedis getJedis(){  
        Jedis jedis = jedisPool.getResource();  
        return jedis;  
    }  
}
```

RedisConfig

```
@Configuration  
public class RedisConfig {  
  
    //读取配置文件中的redis的ip地址  
    @Value("${spring.redis.host:disabled}")  
    private String host;  
  
    @Value("${spring.redis.port:0}")  
    private int port;  
  
    @Value("${spring.redis.database:0}")  
    private int database;  
  
    @Bean  
    public RedisUtil getRedisUtil(){  
        if(host.equals("disabled")){  
            return null;  
        }  
        RedisUtil redisUtil=new RedisUtil();  
        redisUtil.initPool(host,port,database);  
    }  
}
```

```
        return redisUtil;
    }
}
```

同时，任何模块想要调用 redis 都必须在 application.properties 配置，否则不会进行注入。

```
spring.redis.host=redis.server.com
spring.redis.port=6379
spring.redis.database=0
```

现在可以在 manage-service 中的 getSkuInfo()方法测试一下

```
try {
    Jedis jedis = redisUtil.getJedis();
    jedis.get("test","text_value" );
}catch (JedisConnectionException e){
    e.printStackTrace();
}
```

3 使用 redis 进行业务开发

开始开发先说明 redis key 的命名规范，由于 Redis 不像数据库表那样有结构，其所有的数据全靠 key 进行索引，所以 redis 数据的可读性，全依靠 key。

企业中最常用的方式就是：object:id:field

比如：sku:1314:info

user:1092:password

重构 getSkuInfo 方法

```
public SkuInfo getSkuInfo(String skuId){

    Jedis jedis = redisUtil.getJedis();
    String skuKey= RedisConst.sku_prefix+skuId+RedisConst.skuInfo_suffix;
    String skuInfoJson = jedis.get(skuKey);
    if(skuInfoJson!=null ){
```

```
        System.err.println( Thread.currentThread().getName()+"：命中缓存"
" );
        SkuInfo skuInfo = JSON.parseObject(skuInfoJson, SkuInfo.class);
        jedis.close();
        return skuInfo;
    }else{
        System.err.println( Thread.currentThread().getName()+"：未命中
缓存" );

        System.err.println( Thread.currentThread().getName()+"：查询
数据##### ##" );
        SkuInfo skuInfoDB = getSkuInfoDB(skuId);
        String skuInfoJsonStr = JSON.toJSONString(skuInfoDB);

        jedis.setex(skuKey,RedisConst.skuinfo_exp_sec,skuInfoJsonStr);
        System.err.println( Thread.currentThread().getName()+"：数据库
更新完毕##### ##" );
        jedis.close();
        return skuInfoDB;
    }
}
```

以上基本实现使用缓存的方案。

4 高并发时可能会出现的问题：

但在高并发环境下还有如下三个问题。

- 1、如果 redis 宕机了，或者链接不上，怎么办？
- 2、如果 redis 缓存在高峰期到期失效，在这个时刻请求会向雪崩一样，直接访问数据库如何处理？

- 3、如果用户不停地查询一条不存在的数据，缓存没有，数据库也没有，那么会出现什么

情况，如何处理？

```
public SkuInfo getSkuInfo(String skuld){
    SkuInfo skuInfo = null;
    try {
        Jedis jedis = redisUtil.getJedis();
        String skuInfoKey = ManageConst.SKUKEY_PREFIX + skuld + ManageConst.SKUKEY_SUFFIX;
        String skuInfoJson = jedis.get(skuInfoKey);

        if (skuInfoJson == null || skuInfoJson.length() == 0) {
            System.err.println(Thread.currentThread().getName()+"缓存未命中！");
            String skuLockKey = ManageConst.SKUKEY_PREFIX + skuld + ManageConst.SKULOCK_SUFFIX;
            String lock = jedis.set(skuLockKey, "OK", "NX", "PX", ManageConst.SKULOCK_EXPIRE_PX);

            if ("OK".equals(lock)) {
                System.err.println(Thread.currentThread().getName()+"获得分布式锁！");
                skuInfo = getSkuInfoFromDB(skuld);
                if(skuInfo==null){
                    jedis.setex(skuInfoKey, ManageConst.SKUKEY_TIMEOUT, "empty");
                    return null;
                }

                String skuInfoJsonNew = JSON.toJSONString(skuInfo);
                jedis.setex(skuInfoKey, ManageConst.SKUKEY_TIMEOUT, skuInfoJsonNew);
                jedis.close();
                return skuInfo;
            }else{
                System.err.println(Thread.currentThread().getName()+"未获得分布式锁，开始自旋！");
                Thread.sleep(1000);
                jedis.close();
                return getSkuInfo( skuld);
            }
        } else if(skuInfoJson.equals("empty")){
            return null;
        } else {
            System.err.println(Thread.currentThread().getName()+"缓存已命中!!!!!!!!!!!!!!!!!!!!!!");
            skuInfo = JSON.parseObject(skuInfoJson, SkuInfo.class);
            jedis.close();
            return skuInfo;
        }
    }

    } catch (JedisConnectionException e){
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    return getSkuInfoFromDB(skuld);
}
```


作业： 依照 `getSkuInfo` 的方法补全，`getSpuSaleAttrListCheckBySku` 方法和 `getSkuSaleAttrValueListBySpu`，实现详情页面全部信息都从缓存中查询。

`skuInfo` 保存后，清除原缓存中数据。