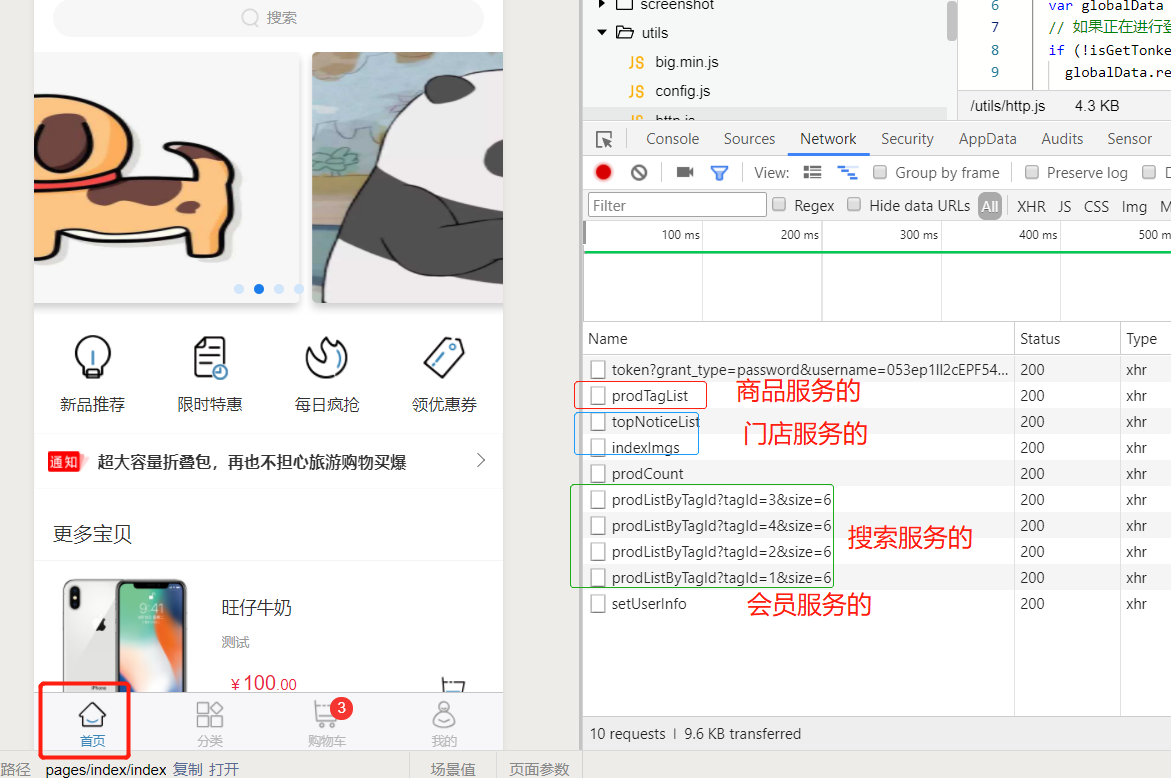
# EGO前台商城的接口处理

## 说在前面

我们现在已经完成了商品模块，搜索模块，用户模块，门店模块等，打开前台商城，会有很多加载的接口，我们完善一下

## 打开微信小程序控制台，查看首页接口



## 我们从上往下一个一个来

### **/prod/prodTag/prodTagList（商品服务的）**

#### 在product-api中添加vo对象（用于前台展示的）

|  |
| --- |
| @Data  public class ProdTagVo {  @ApiModelProperty("标签活动的id")  private Long id;  @ApiModelProperty("标签活动的标题")  private String title;  @ApiModelProperty("标签活动的样式")  private Integer style;  } |

#### 修改product-service的ProdTagController

|  |
| --- |
| /\*\*  \* 前台的数据交互  \*/  @GetMapping({"prodTagList"})  @ApiOperation("加载首页数据的标签分组")  public ResponseEntity<List<ProdTagVo>> loadIndexTag() {  List<ProdTagVo> prodTagList = prodTagService.loadIndexProdTag();  if ((prodTagList == null) || (prodTagList.isEmpty())) {  return ResponseEntity.ok(Collections.emptyList());  }  return ResponseEntity.ok(prodTagList);  } |

#### 修改product-service的ProdTagService

|  |
| --- |
| /\*\*  \* 加载前台的商品标签展示  \*  \* @return  \*/  List<ProdTagVo> loadIndexProdTag(); |

#### 修改product-service的ProdTagServiceImpl

|  |
| --- |
| /\*\*  \* 加载前台的商品标签展示  \*  \* @return  \*/  @Override  public List<ProdTagVo> loadIndexProdTag() {  ArrayList<ProdTagVo> prodTagVos = new ArrayList<>();  //查询prodTag表  List<ProdTag> prodTags = prodTagMapper.selectList(new LambdaQueryWrapper<ProdTag>()  .eq(ProdTag::getStatus, 1)  .orderByAsc(ProdTag::getSeq)  );  //组装数据  prodTags.forEach(prodTag -> {  ProdTagVo prodTagVo = new ProdTagVo();  prodTagVo.setId(prodTag.getId());  prodTagVo.setStyle(prodTag.getStyle());  prodTagVo.setTitle(prodTag.getTitle());  prodTagVos.add(prodTagVo);  });  return prodTagVos;  } |

### /shop/notice/topNoticeList（门店服务的）

#### 在store-api中添加vo对象（用于前台展示的）

|  |
| --- |
| @Data  public class NoticeVo {  @ApiModelProperty("公告的标题")  private String title;  } |

#### 修改store-service中的NoticeController

|  |
| --- |
| /\*\*  \* 加载前台的置顶通知  \*  \* @return  \*/  @GetMapping({"topNoticeList"})  @ApiOperation("加载置顶的通知")  public ResponseEntity<List<NoticeVo>> loadTopNotice() {  List<NoticeVo> notices = noticeService.loadIndexTopNotice();  if (notices == null) {  return ResponseEntity.ok(Collections.emptyList());  }  return ResponseEntity.ok(notices);  } |

#### 修改store-service中的NoticeService

|  |
| --- |
| /\*\*  \* 加载前台的置顶公告  \*  \* @return  \*/  List<NoticeVo> loadIndexTopNotice(); |

#### 修改store-service中的NoticeServiceImpl

|  |
| --- |
| /\*\*  \* 加载前台的置顶公告  \*  \* @return  \*/  @Override  public List<NoticeVo> loadIndexTopNotice() {  ArrayList<NoticeVo> noticeVos = new ArrayList<>();  //查询Notice表  List<Notice> notices = noticeMapper.selectList(new LambdaQueryWrapper<Notice>().  eq(Notice::getIsTop, 1).  eq(Notice::getStatus, 1).  orderByAsc(Notice::getPublishTime)  );  //组装数据  notices.forEach(notice -> {  NoticeVo noticeVo = new NoticeVo();  noticeVo.setTitle(notice.getTitle());  noticeVos.add(noticeVo);  });  return noticeVos;  } |

### /admin/indexImg/indexImgs（门店服务的）

#### 在store-api中添加vo对象（用于前台展示的）

|  |
| --- |
| @Data  @AllArgsConstructor  @NoArgsConstructor  public class IndexImgVo {  @ApiModelProperty("轮播图的地址")  private String imgUrl;  @ApiModelProperty("广告对应的商品id")  private Long relation;  } |

#### 修改store-service中的IndexImgController

|  |
| --- |
| /\*\*  \* 前台轮播图的加载  \*/  @GetMapping("/indexImgs")  @ApiOperation("加载前台商城的轮播图")  public ResponseEntity<List<IndexImgVo>> loadIndexImgs() {  List<IndexImgVo> indexImgList = indexImgService.loadIndexImgs();  if (indexImgList == null || indexImgList.isEmpty()) {  return ResponseEntity.ok(Collections.EMPTY\_LIST);  }  return ResponseEntity.ok(indexImgList);  } |

#### 修改store-service中的IndexImgService

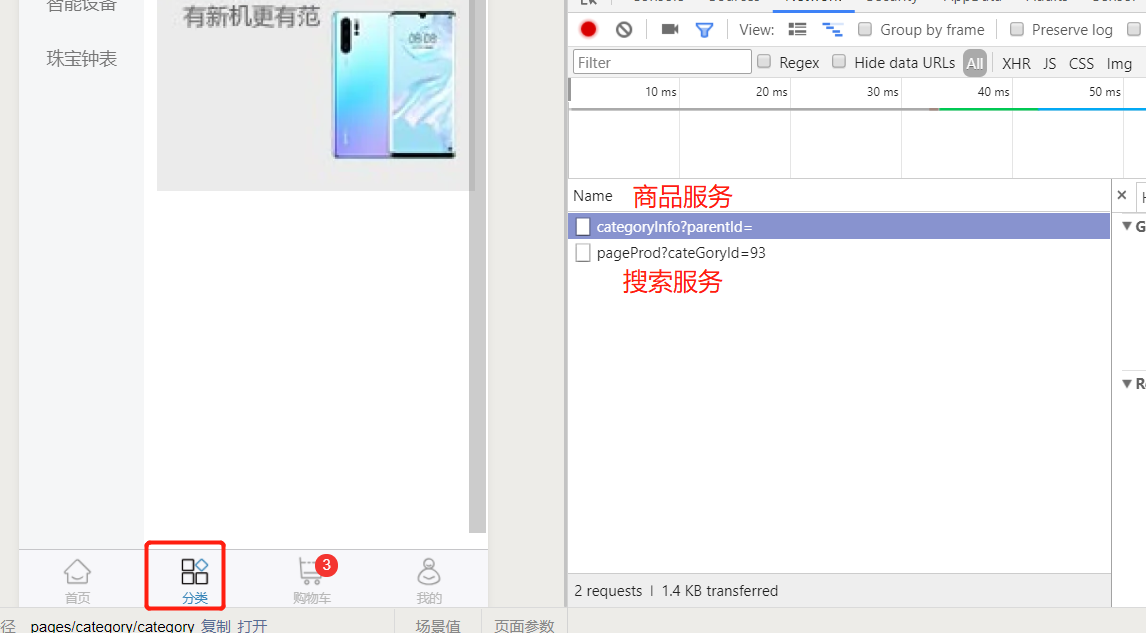
|  |
| --- |
| /\*\*  \* 轮播图的加载  \*  \* @return  \*/  List<IndexImgVo> loadIndexImgs(); |

#### 修改store-service中的IndexImgServiceImpl

|  |
| --- |
| /\*\*  \* 轮播图的加载  \*  \* @return  \*/  @Override  @Cacheable(key = IndexImgConstant.INDEX\_ALL\_IMGS)  public List<IndexImgVo> loadIndexImgs() {  //查询IndexImg表  List<IndexImg> indexImgs = indexImgMapper.selectList(new LambdaQueryWrapper<IndexImg>()  .eq(IndexImg::getStatus, 1)  .orderByAsc(IndexImg::getSeq) //前端根据这个排序  );  //转换对象  List<IndexImgVo> indexImgVos = new ArrayList<>();  if (indexImgs != null && indexImgs.size() > 0) {  indexImgs.forEach(indexImg -> {  IndexImgVo indexImgVo = new IndexImgVo();  indexImgVo.setImgUrl(indexImg.getImgUrl());  indexImgVo.setRelation(indexImg.getRelation());  indexImgVos.add(indexImgVo);  });  }  return indexImgVos;  } |

### /prod/prodListByTagId（搜索服务的，之前写完了）

## 查看分类接口



### /prod/category/category/categoryInfo（商品服务）

#### 在product-api中添加vo对象（用于前台展示的）

|  |
| --- |
| @Data  public class CategoryVo {  @ApiModelProperty("分类的id")  private Long categoryId;  @ApiModelProperty("分类的名称")  private String categoryName;  @ApiModelProperty("分类的名称")  private String pic;  } |

#### 修改product-service中CategoryController

|  |
| --- |
| @GetMapping({"/category/categoryInfo"})  @ApiOperation("加载所有的二级菜单")  public ResponseEntity<List<CategoryVo>> loadAllSubCategorys() {  List<CategoryVo> categories = categoryService.loadAllSubCategorys();  if ((categories == null) || (categories.isEmpty())) {  return ResponseEntity.ok(Collections.emptyList());  }  return ResponseEntity.ok(categories);  } |

#### 修改product-service中CategoryService

|  |
| --- |
| /\*\*  \* 列出所有分类的子节点  \*  \* @return  \*/  List<CategoryVo> loadAllSubCategorys(); |

#### 修改product-service中CategoryServiceImpl

|  |
| --- |
| /\*\*  \* 列出所有分类的子节点  \*  \* @return  \*/  @Override  public List<CategoryVo> loadAllSubCategorys() {  //查询category表  List<Category> categories = categoryMapper.selectList(new LambdaQueryWrapper<Category>()  .eq(Category::getGrade, 2)  );  //组装数据  ArrayList<CategoryVo> categoryVos = new ArrayList<>(categories.size() \* 2);  categories.forEach(category -> {  CategoryVo categoryVo = new CategoryVo();  categoryVo.setCategoryId(category.getCategoryId());  categoryVo.setCategoryName(category.getCategoryName());  categoryVo.setPic(category.getPic());  categoryVos.add(categoryVo);  });  return categoryVos;  } |

### /prod/pageProd（搜索服务的，之前已经写完了哈哈）