乐字节教育高级架构课程

正所谓“授人以鱼不如授人以渔”，你们想要的 Java 学习资料来啦！不管你是学生，还是已经步入职场的同行，希望你们都要珍惜眼前的学习机会，奋斗没有终点，知识永不过时。

扫描下方二维码即可领取

乐字节晓啡

乐字节官方交流群

点击去结算跳转订单系统

前台系统添加订单系统url

shop-portal的application-dev.yml

```
# shop-order系统url
shop.order.url: http://localhost:9092/shop-order/
```

订单系统页面处理

将 `填写核对单.html` 导入订单系统修改名称为shop-order的order/preOrder.ftl

设置变量ctx，其值为项目的根路径

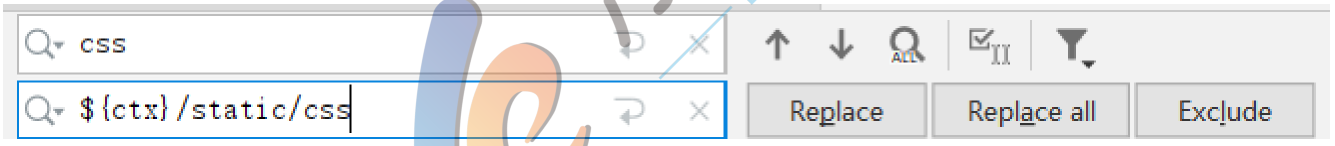
```
<!-- 设置项目根路径全局变量 -->
<#assign ctx=request.contextPath/>
```

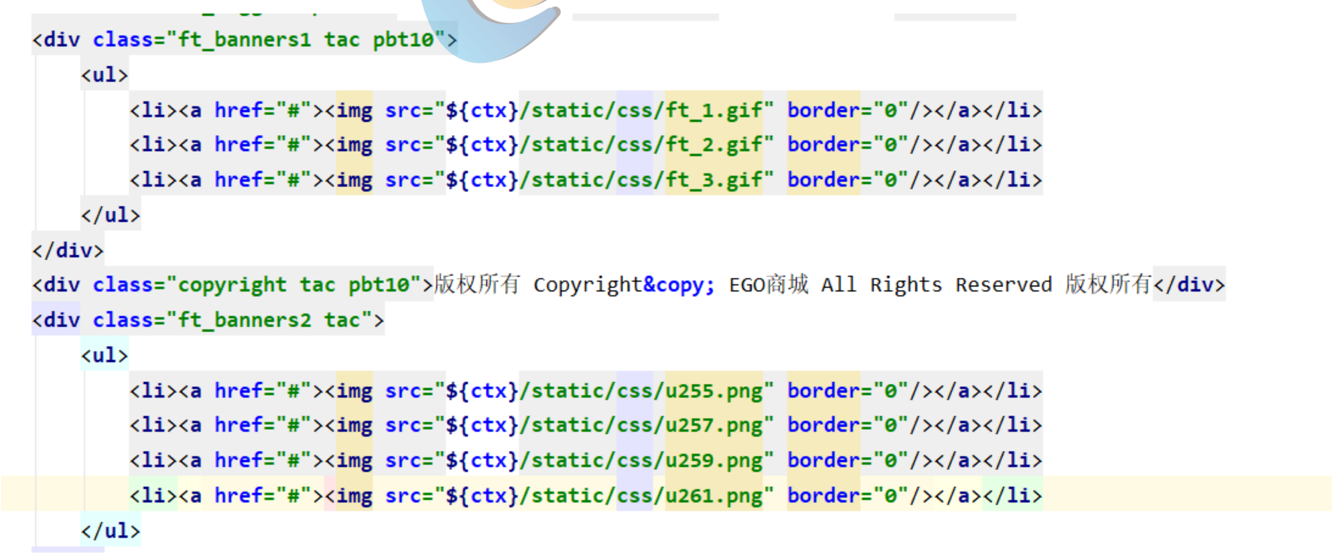
处理页面中的链接地址，编写页面跳转请求的代码

`css/` 替换为 `${ctx}/static/css/`

`js/` 替换为 `$(ctx)/static/js/`

`images/` 替换为 `${ctx}/static/images/`





```
<div class="ft_banners1 tac pbt10">
    <ul>
        <li><a href="#"><img src="${ctx}/static/css/ft_1.gif" border="0"/></a></li>
        <li><a href="#"><img src="${ctx}/static/css/ft_2.gif" border="0"/></a></li>
        <li><a href="#"><img src="${ctx}/static/css/ft_3.gif" border="0"/></a></li>
    </ul>
</div>
<div class="copyright tac pbt10">版权所有 Copyright&copy; EGO商城 All Rights Reserved 版权所有</div>
<div class="ft_banners2 tac">
    <ul>
        <li><a href="#"><img src="${ctx}/static/css/u255.png" border="0"/></a></li>
        <li><a href="#"><img src="${ctx}/static/css/u257.png" border="0"/></a></li>
        <li><a href="#"><img src="${ctx}/static/css/u259.png" border="0"/></a></li>
        <li><a href="#"><img src="${ctx}/static/css/u261.png" border="0"/></a></li>
    </ul>
```

前台系统编写全局拦截器

shop-portal的PortalCommonInterceptor.java

```java
package com.xxxx.portal.interceptor;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import org.springframework.util.StringUtils;
import org.springframework.web.servlet.HandlerInterceptor;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * 全局拦截器
 *
 * @author zhoubin
 * @since 1.0.0
 */
@Component
public class PortalCommonInterceptor implements HandlerInterceptor {

   @Value("${shop.order.url}")
   private String shopOrderUrl;

   //请求处理的方法之前执行
   //如果返回true，执行下一个拦截器或者目标程序，如果返回false，不执行下一个拦截器或者目标程序
   @Override
   public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
Object handler) throws Exception {
      //获取Application对象中的url地址
      ServletContext context = request.getSession().getServletContext();
      String orderUrl = (String) context.getAttribute("orderUrl");
      if (StringUtils.isEmpty(orderUrl)){
         context.setAttribute("orderUrl",shopOrderUrl);
      }
      return true;
   }

   //请求处理的方法之后执行
   @Override
   public void postHandle(HttpServletRequest request, HttpServletResponse response, Object
handler, ModelAndView modelAndView) throws Exception {

   }

   //处理后执行清理工作
   @Override
   public void afterCompletion(HttpServletRequest request, HttpServletResponse response,
Object handler,
                               Exception ex) throws Exception {

   }
}
```

shop-portal的MvcConfig.java

```java
package com.xxxx.portal.config;

import com.xxxx.portal.interceptor.PortalCommonInterceptor;
import com.xxxx.portal.interceptor.PortalLoginInterceptor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

/**
 * MVC配置类
 *
 * @author zhoubin
 * @since 1.0.0
 */
@Configuration
@EnableWebMvc
public class MvcConfig implements WebMvcConfigurer {

   @Autowired
   private PortalLoginInterceptor loginInterceptor;
   @Autowired
   private PortalCommonInterceptor commonInterceptor;

   /**
    * 放行静态资源
    *
    * @param registry
    */
   @Override
   public void addResourceHandlers(ResourceHandlerRegistry registry) {
      registry.addResourceHandler("/static/**").addResourceLocations("classpath:/static/");
   }

   /**
    * addInterceptor:添加自定义拦截器
    * addPathPatterns: 添加拦截请求 /**表示拦截所有
    * excludePathPatterns: 不拦截的请求
    *
    * @param registry
    */
   @Override
   public void addInterceptors(InterceptorRegistry registry) {
      registry.addInterceptor(commonInterceptor)
            .addPathPatterns("/**");
      registry.addInterceptor(loginInterceptor)
            .addPathPatterns("/cart/**")
            .excludePathPatterns("/static/**")
            .excludePathPatterns("/login/**")
```

```
                .excludePathPatterns("/user/login/**")
                .excludePathPatterns("/user/logout/**");
    }
}
```

订单系统编写Controller

shop-order的OrderController.java

```
package com.xxxx.order.controller;

import com.alibaba.dubbo.config.annotation.Reference;
import com.xxxx.common.pojo.Admin;
import com.xxxx.rpc.service.CartService;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

import javax.servlet.http.HttpServletRequest;

/**
 * 订单Controller
 *
 * @author zhoubin
 * @since 1.0.0
 */
@Controller
@RequestMapping("order")
public class OrderController {

   @Reference(interfaceClass = CartService.class)
   private CartService cartService;

   /**
    * 跳转到预订单页面
    * @return
    */
   @RequestMapping("preOrder")
   public String preOrder(Model model, HttpServletRequest request){
      Admin admin = (Admin) request.getSession().getAttribute("user");
      model.addAttribute("cartResult",cartService.getCartList(admin));
      return "order/preOrder";
   }
}
```

前台系统编写Controller

shop-portal的OrderController.java

```
package com.xxxx.portal.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

import javax.servlet.http.HttpServletRequest;

/**
 * 订单
 *
 * @author zhoubin
 * @since 1.0.0
 */
@Controller
@RequestMapping("order")
public class OrderController {

    /**
     * 跳转订单系统
     * @param request
     * @return
     */
    @RequestMapping("toPreOrder")
    public String toPreOrder(HttpServletRequest request){
        return
"redirect:"+request.getSession().getServletContext().getAttribute("orderUrl")+"order/preOrd
er";
    }
}
```