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

添加单点登录环境

订单系统配置拦截器

shop-order的OrderLoginInterceptor.java

```
package com.xxxx.order.interceptor;

import com.alibaba.dubbo.config.annotation.Reference;
import com.xxxx.common.pojo.Admin;
import com.xxxx.common.util.CookieUtil;
import com.xxxx.common.util.JsonUtil;
import com.xxxx.sso.service.SSOService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.redis.core.RedisTemplate;
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.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.concurrent.TimeUnit;

/**
 * 订单登录拦截器
 *
 * @author zhoubin
 * @since 1.0.0
 */
@Component
public class OrderLoginInterceptor implements HandlerInterceptor {

    @Reference(interfaceClass = SSOService.class)
    private SSOService ssoService;

    @Value("${user.ticket}")
    private String userTicket;

    @Value("${ego.portal.url}")
    private String portalUrl;

    private static final String COOKIE_NAME = "userTicket";

    @Autowired
    private RedisTemplate<String, String> redisTemplate;

    //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 {
        //获取用户名票据
        String ticket = CookieUtil.getCookieValue(request, COOKIE_NAME);
        if (!StringUtils.isEmpty(ticket)) {
            //如果票据存在，进行验证
            Admin admin = ssoService.validate(ticket);
            //将用户信息添加至session中，用于页面返显
            request.getSession().setAttribute("user", admin);
            //重新设置新的失效时间
            redisTemplate.opsForValue().set(userTicket + ":" + ticket,
JsonUtil.object2JsonStr(admin), 30,
                    TimeUnit.MINUTES);
            return true;
        }
        //票据不存在或者用户验证失败，重定向前台门户登录页面
        response.sendRedirect(portalUrl + "login?redirectUrl=" + request.getRequestURL());
        return false;
    }

    //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-order的MvcConfig.java

```
package com.xxxx.order.config;

import com.xxxx.order.interceptor.OrderLoginInterceptor;
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 OrderLoginInterceptor loginInterceptor;

    /**
     * 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(loginInterceptor)
                .addPathPatterns("/**")
                .excludePathPatterns("/static/**")
                .excludePathPatterns("/login/**")
                .excludePathPatterns("/image/**")
                .excludePathPatterns("/user/login/**")
                .excludePathPatterns("/user/logout/**");
    }
}
```

前台系统修改Controller

精确匹配订单系统发出的login请求，获取重定向url参数

shop-portal的PageController.java

```
/**
 * 精确匹配登录
 * @param redirectUrl
 * @param model
 * @return
 */
@RequestMapping("login")
public String login(String redirectUrl, Model model){
   model.addAttribute("redirectUrl",redirectUrl);
   return "login";
}
```

前台系统页面处理

shop-portal的login.ftl

```
<input type="hidden" id="redirectUrl" value="${redirectUrl!''}"/>
<script type="text/javascript">
    // 用户登录
    function userLogin() {
        $.ajax({
            url: "${ctx}/user/login",
            type: "POST",
            data: $("#formlogin").serialize(),
            dataType: "JSON",
            success: function (result) {
                if (200 == result.code) {
                    // 如果存在重定向url则重定向至该url
                    if ($("#redirectUrl").val()) {
                        location.href = $("#redirectUrl").val();
                    }else {
                        location.href = "${ctx}/index";
                    }

                } else {
                    layer.msg("用户名或密码错误，请重新输入！");
                }
            },
            error: function () {
                layer.alert("亲，系统正在升级中，请稍后再试！");
            }
        });
    }
</script>
```

测试

- 商城前台系统未登录然后访问订单系统，是否会跳转至商城前台系统登录页面
- 商城前台系统登录以后访问订单系统，是否可以进入订单系统index.ftl
- 商城前台系统未登录然后访问订单系统，是否会跳转至商城前台系统登录页面，如果这时候用户登录，是否可以跳回至订单系统index.ftl
- 商城前台系统安全退出以后访问订单系统，是否会跳转至商城前台系统登录页面