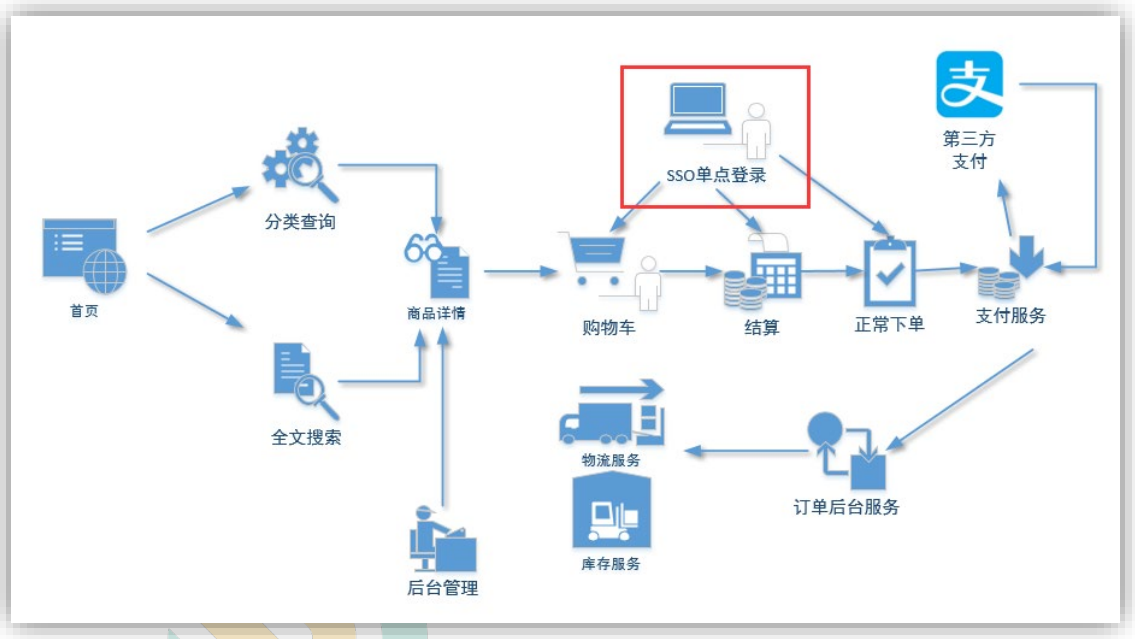
尚品汇商城

单点登录

所在位置：

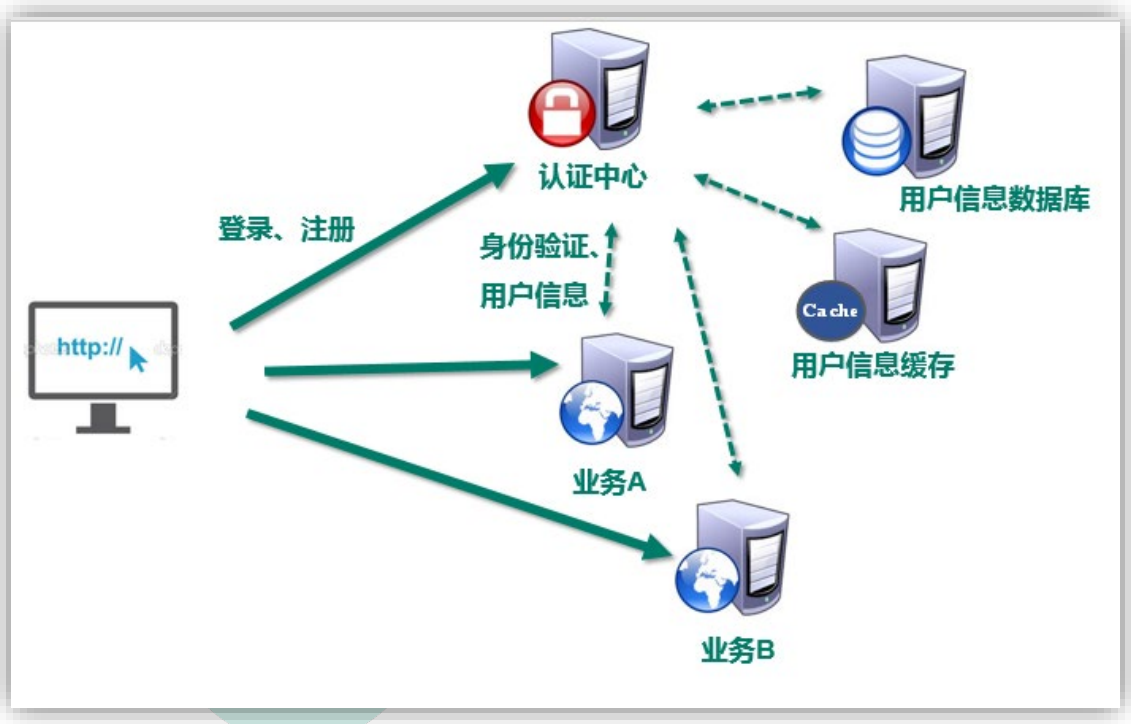


一、单点登录业务介绍

早期单一服务器，用户认证。

缺点：单点性能压力，无法扩展

分布式，SSO(single sign on)模式



解决：

用户身份信息独立管理，更好的分布式管理。

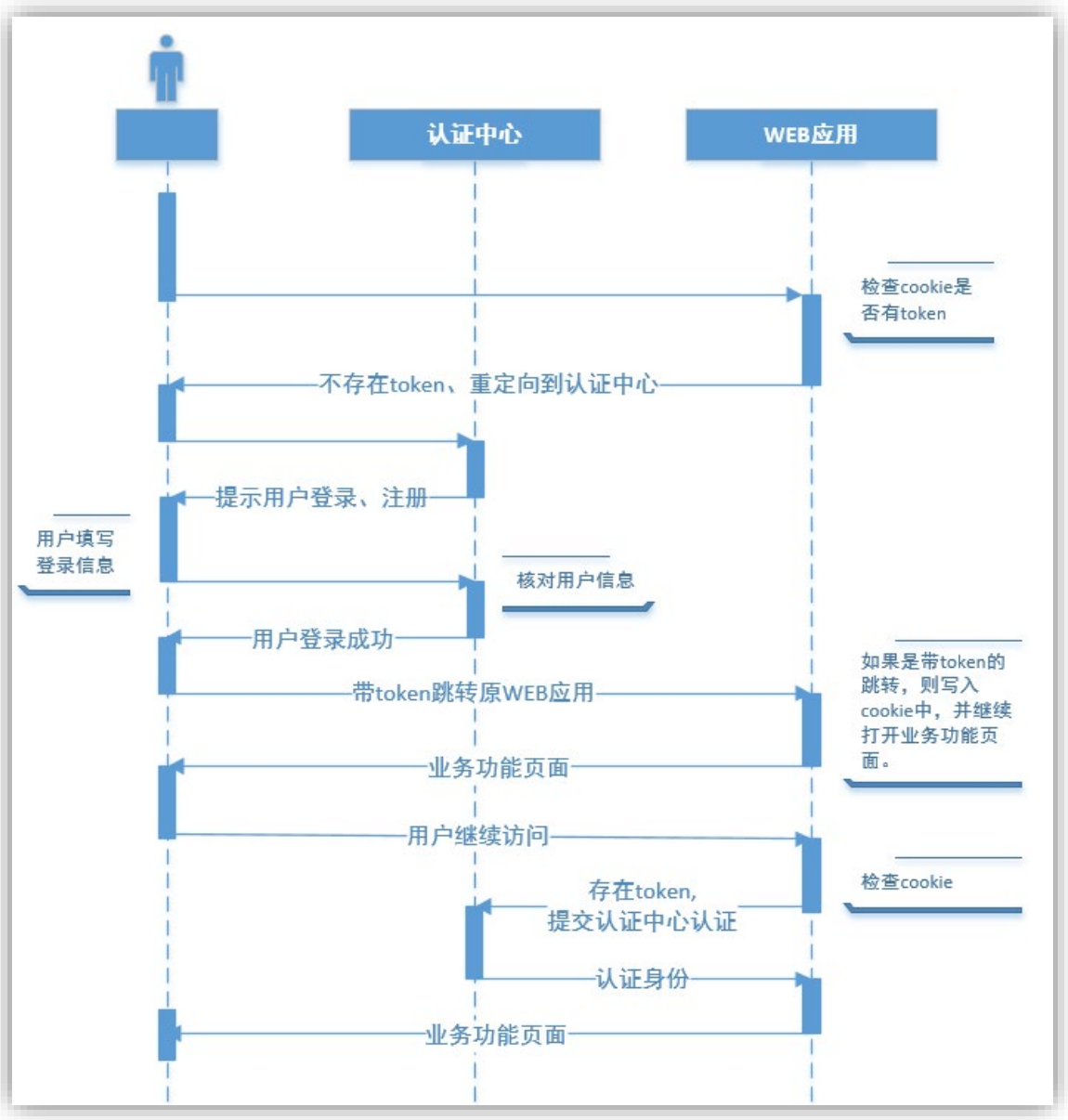
可以自己扩展安全策略

跨域不是问题

缺点：

认证服务器访问压力较大。

业务流程图{用户访问业务时，必须登录的流程}{单点登录的过程}



二、认证中心模块

2.1 实现思路

1、用接收的用户名密码核对后台数据库

2、核对通过，用 uuid 生成 token

3、将用户 id 加载到写入 redis，redis 的 key 为 token，value 为用户 id。

4、登录成功返回 token 与用户信息，将 token 与用户信息记录到 cookie 里面

5、重定向用户到之前的来源地址。

数据库表：user_info，并添加一条数据！**密码应该是加密的！**

2.2 搭建认证中心模块 service-user

2.2.1 搭建 service-user 服务

搭建方式如 service-item

2.2.2 修改配置 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
https://maven.apache.org/xsd/maven-4.0.0.xsd">

   <modelVersion>4.0.0</modelVersion>
```

```
    <parent>
        <groupId>com.atguigu.gmall</groupId>
        <artifactId>service</artifactId>
        <version>1.0</version>
    </parent>

    <version>1.0</version>
    <artifactId>service-user</artifactId>
    <packaging>jar</packaging>
    <name>service-user</name>
    <description>service-user</description>

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

</project>
```

2.2.3 添加配置文件

bootstrap.properties

```
spring.application.name=service-user

spring.profiles.active=dev
```

```
spring.cloud.nacos.discovery.server-addr=192.168.254.129:8848
spring.cloud.nacos.config.server-addr=192.168.254.129:8848
spring.cloud.nacos.config.prefix=${spring.application.name}
spring.cloud.nacos.config.file-extension=yaml
spring.cloud.nacos.config.shared-configs[0].data-id=common.yaml
```

启动类

```
package com.atguigu.gmall.user;



@SpringBootApplication
@ComponentScan({"com.atguigu.gmall"})
@EnableDiscoveryClient
public class ServiceUserApplication {

    public static void main(String[] args) {
        SpringApplication.run(ServiceUserApplication.class, args);
    }

}
```

2.2.4 封装登录接口

2.2.4.1 编写接口

```
package com.atguigu.gmall.user.service;
```

```
public interface UserService {


    /**
     * 登录方法
   * @param userInfo
     * @return
     */
    UserInfo login(UserInfo userInfo);


}
```

2.2.4.2 Mapper

UserInfoMapper

```
package com.atguigu.gmall.user.mapper;


import com.atguigu.gmall.model.user.UserInfo;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import org.apache.ibatis.annotations.Mapper;


@Mapper
public interface UserInfoMapper extends BaseMapper<UserInfo> {
}
```

2.2.4.3 实现类

```
package com.atguigu.gmall.user.service.impl;



@Service
public class UserServiceImpl implements UserService {
```

```
    // 调用mapper 层
  @Autowired
    private UserInfoMapper userInfoMapper;



    @Override
    public UserInfo login(UserInfo userInfo) {
        // select * from userInfo where userName = ? and passwd = ?
        // 注意密码是加密:
    String passwd = userInfo.getPasswd(); //123
        // 将passwd 进行加密
    String                          newPasswd                          =
DigestUtils.md5DigestAsHex(passwd.getBytes());


        QueryWrapper<UserInfo> queryWrapper = new QueryWrapper<>();
        queryWrapper.eq("login_name", userInfo.getLoginName());
        queryWrapper.eq("passwd", newPasswd);
        UserInfo info = userInfoMapper.selectOne(queryWrapper);
        if (info != null) {


            return info;
        }
        return null;
    }
}
```

2.2.4.4 控制器

```
package com.atguigu.gmall.user.controller;
/**
```

```
 * <p>
 * 用户认证接口
 * </p>
 */
@RestController
@RequestMapping("/api/user/passport")
public class PassportApiController {


    @Autowired
    private UserService userService;


    @Autowired
    private RedisTemplate redisTemplate;


    /**
     * 登录
     * @param userInfo
     * @param request
     * @param response
     * @return
     */
    @PostMapping("login")
    public Result login(@RequestBody UserInfo userInfo,
HttpServletRequest request, HttpServletResponse response) {
        System.out.println("进入控制器！");
        UserInfo info = userService.login(userInfo);


        if (info != null) {
            String token =
UUID.randomUUID().toString().replaceAll("-", "");
            HashMap<String, Object> map = new HashMap<>();
```

```
            map.put("nickName", info.getNickName());
            map.put("token", token);

            JSONObject userJson = new JSONObject();
            userJson.put("userId", info.getId().toString());
            userJson.put("ip", IpUtil.getIpAddress(request));

redisTemplate.opsForValue().set(RedisConst.USER_LOGIN_KEY_PREFIX + token, userJson.toJSONString(), RedisConst.USERKEY_TIMEOUT, TimeUnit.SECONDS);
            return Result.ok(map);
        } else {
            return Result.fail().message("用户名或密码错误");
        }
    }

    /**
     * 退出登录
     * @param request
     * @return
     */
    @GetMapping("logout")
    public Result logout(HttpServletRequest request){
        redisTemplate.delete(RedisConst.USER_LOGIN_KEY_PREFIX + request.getHeader("token"));
        return Result.ok();
    }
}
```

2.3 在 web-all 模块添加实现

2.3.1 配置网关 server-gateway

```
- id: service-user
  uri: lb://service-user
  predicates:
  - Path=/*/user/**


- id: web-passport

  uri: lb://web-all

  predicates:

  - Host=passport.gmall.com
```

2.3.2 在 web-all 项目中跳转页面

```
package com.atguigu.gmall.all.controller;


/**
 * <p>
 * 用户认证接口
 * </p>
 *
 */
```

```
@Controller
public class PassportController {


    /**
     *
     * @return
     */
    @GetMapping("login.html")
    public String login(HttpServletRequest request) {
        String originUrl = request.getParameter("originUrl");
        request.setAttribute("originUrl",originUrl);
        return "login";
    }

}
```

2.3.3 登录页面

页面资源： \templates\login1.html

login1.html 没有公共头部信息

login.html 有公共头部信息

Html 关键代码

```
<form class="sui-form">
   <div class="input-prepend"><span class="add-on loginname"></span>
      <input id="inputName" type="text" v-model="user.loginName"
placeholder="邮箱/用户名/手机号" class="span2 input-xfat">
   </div>
   <div class="input-prepend"><span class="add-on loginpwd"></span>
      <input id="inputPassword" type="password" v-
model="user.passwd" placeholder="请输入密码" class="span2 input-
xfat">
   </div>
   <div class="setting">
      <label class="checkbox inline">
         <input name="m1" type="checkbox" value="2" checked="">
         自动登录
   </label>
      <span class="forget">忘记密码？</span>
   </div>
   <div class="logined">
```

```
      <a class="sui-btn btn-block btn-xlarge btn-danger"
href="javascript:" @click="submitLogin()">登  录</a>
   </div>
</form>

<script src="/js/api/login.js"></script>
<script th:inline="javascript">
   var item = new Vue({
      el: '#profile',

      data: {
            originUrl: [[${originUrl}]],
            user: {
                loginName: '',
                passwd: ''
            }
      },

      created() {
      },

      methods: {
            submitLogin() {
                login.login(this.user).then(response => {

                    if (response.data.code == 200) {
                        //把token 存在 cookie 中、也可以放在
localStorage 中
            auth.setToken(response.data.data.token)

auth.setUserInfo(JSON.stringify(response.data.data))
```

```
                        console.log("originUrl:"+this.originUrl);
                        if(this.originUrl == ''){
window.location.href="http://www.gmall.com/index.html"
                            return ;
                        } else {
                            window.location.href =
decodeURIComponent(this.originUrl)
                    }
                      } else {
                    alert(response.data.data.message)
                }

                 })
            }
        }
    })
</script>
```

2.4 头部信息处理

web-all 项目：common/header.html，common/head.html

功能：头部信息为公共信息，所有页面都具有相关的头部，所以我们可以单独提取出来，头部页面显示登录状态与关键字搜索等信息

2.4.1 提取头部信息

提取头部信息我们会用到 thymeleaf 两个标签：

`th:fragment：定义代码块`

`th:include：将代码块片段包含的内容插入到使用了 th:include 的 HTML 标签中`

1，定义头部代码块（/common/header.html），关键代码

```
<div id="nav-bottom" th:fragment="header">
    <!--顶部-->
        <div class="nav-top" id="header">
                ...
        </div>
</div>
```

2，在其他页面引用头部代码块

```
<div th:include="common/header :: header"></div>
```

2.4.2 头部登录状态处理

思路：登录成功后我们将用户信息写入了 cookie，所以我们判断 cookie 中是否有用户信息，如果有则显示登录用户信息和退出按钮，我们采取 vue 的渲染方式

关键代码

Header.html 中

```
<ul class="fl">
    <li class="f-item">尚品汇欢迎您！</li>
    <li  v-if="userInfo.nickName == ''" class="f-item">请<span><a href="javascript:" @click="login()">登录</a></span>  <span><a href="#">免费注册</a></span></li>
    <li  v-if="userInfo.nickName != ''" class="f-item"><span>{{userInfo.nickName}}</span>  <span><a href="javascript:" @click="logout()">退出</a></span></li>
</ul>

<script type="text/javascript" src="/js/plugins/jquery/jquery.min.js"></script>
<script type="text/javascript" src="/js/plugins/jquery.cookie.js"></script>
<script src="/js/plugins/vue.js"></script>
```

```
<script src="/js/plugins/axios.js"></script>
<script src="/js/auth.js"></script>
<script src="/js/request.js"></script>
<script src="/js/api/login.js"></script>
<script th:inline="javascript">
    var item = new Vue({
        el: '#header',

        data: {
            userInfo: {
                nickName: '',
                name: ''
            }
        },

        created() {
            this.showInfo()
        },

        methods: {
            showInfo() {
                // debugger
                if(auth.getUserInfo()) {
                    this.userInfo = auth.getUserInfo()
                    console.log("--------"+this.userInfo.nickName)
                }
            },


            logout() {
```

```
                //debugger
                login.logout().then(response => {
                    console.log("已退出")
                    auth.removeToken()
                    auth.removeUserInfo()

                    //跳转页面
          window.location.href = "/"
                })
            }
        }
    })
</script>
```

2.4.3 头部关键字搜索

```
<div class="input-append">
    <input id="keyword" type="text" v-model="keyword" class="input-error input-xxlarge" />
    <button class="sui-btn btn-xlarge btn-danger" @click="search()" type="button">搜索</button>
</div>
<script th:inline="javascript">
    var item = new Vue({
        el: '#header',

        data: {
            keyword: [[${searchParam?.keyword}]],
            userInfo: {
                nickName: '',
                name: ''
```

```
        }
    },

    created() {
        this.showInfo()
    },

    methods: {
        showInfo() {
            // debugger
            if(auth.getUserInfo()) {
                this.userInfo = auth.getUserInfo()
                console.log("--------"+this.userInfo.nickName)
            }
        },

        search() {
            if(this.keyword == null) this.keyword = ''
            window.location.href =
'http://list.gmall.com/search.html?keyword=' + this.keyword
        },

        login() {
            window.location.href =
'http://passport.gmall.com/login.html?originUrl='+window.location.hr
ef
        },

        logout() {
            //debugger
            login.logout().then(response => {
                console.log("已退出")
```

```
                    auth.removeToken()
                    auth.removeUserInfo()


                    //跳转页面
            window.location.href = "/"
                })
            }
        }
    })
</script>
```

说明：[[${searchParam?.keyword}]]，searchParam 为搜索列表的搜索对象，如果存在 searchParam 对象，显示关键字的值

2.4.4 头部公共 js

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>


<div th:fragment="head">
    <script type="text/javascript" src="/js/plugins/jquery/jquery.min.js"></script>
    <script type="text/javascript" src="/js/plugins/jquery.cookie.js"></script>
    <script src="/js/plugins/vue.js"></script>
    <script src="/js/plugins/axios.js"></script>
    <script src="/js/auth.js"></script>
```

```
    <script src="/js/request.js"></script>
</div>
</body>
</html>
```

引用

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=9; IE=8; IE=7; IE=EDGE">
    <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />
    <title>尚品汇，欢迎登录</title>
    <link rel="icon" href="./img/favicon.ico">

    <link rel="stylesheet" type="text/css" href="/css/all.css" />
    <link rel="stylesheet" type="text/css" href="/css/pages-login.css" />
    <div th:include="common/head :: head"></div>
</head>

<body>
    <!-- 头部栏位 -->
    <!--页面顶部-->
    <div th:include="common/header :: header"></div>
```

三、用户认证与服务网关整合

3.1 实现思路

1. 所有请求都会经过服务网关，服务网关对外暴露服务，不管是 api 异步请求还是 web 同步请求都走网关，在网关进行统一用户认证
2. 既然要在网关进行用户认证，网关得知道对哪些 url 进行认证，所以我们得对 url 制定规则
3. Web 页面同请求（如：*.html），我采取配置白名单的形式，凡是配置在白名单里面的请求都是需要用户认证的（注：也可以采取域名的形式，方式多多）

4. Api 接口异步请求的，我们采取 url 规则匹配，如：/api/**/auth/**，如凡是满足该规则的都必须用户认证

在网关添加 redis 相关配置！

```
redis:
  host: 192.168.200.129
  port: 6379
  database: 0
  timeout: 1800000
  password:


authUrls:
  url: trade.html,myOrder.html,list.html
```

pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis-reactive</artifactId>
</dependency>
```

在网关中添加 redis 的配置类。

注意需要引入 RedisConfig 配置类

3.2 在服务网关添加 fillter

server-gateway 项目中添加一个过滤器

```
package com.atguigu.gmall.gateway.filter;


@Component
public class AuthGlobalFilter implements GlobalFilter{

    @Autowired
    private RedisTemplate redisTemplate;

    // 匹配路径的工具类
  private AntPathMatcher antPathMatcher = new AntPathMatcher();

    @Value("${authUrls.url}")
    private String authUrls;

    @Override
    public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain
chain) {
        // 获取到请求对象
   ServerHttpRequest request = exchange.getRequest();
        // 获取 Url
        String path = request.getURI().getPath();
        // 如果是内部接口，则网关拦截不允许外部访问！
   if (antPathMatcher.match("/**/inner/**",path)){
            ServerHttpResponse response = exchange.getResponse();
            return out(response,ResultCodeEnum.PERMISSION);
        }
        // 获取用户 Id
        String userId = getUserId(request);
```

```
//token 被盗用
        if("-1".equals(userId)) {
            ServerHttpResponse response = exchange.getResponse();
            return out(response,ResultCodeEnum.PERMISSION);
        }
        // 用户登录认证
    //api 接口，异步请求，校验用户必须登录
    if(antPathMatcher.match("/api/**/auth/**", path)) {
             if(StringUtils.isEmpty(userId)) {
                 ServerHttpResponse response = exchange.getResponse();
                 return out(response,ResultCodeEnum.LOGIN_AUTH);
             }
        }
        // 验证 url
        for (String authUrl : authUrls.split(",")) {
            // 当前的 url 包含登录的控制器域名，但是用户 Id 为空!
     if (path.indexOf(authUrl)!=-1 && StringUtils.isEmpty(userId)){
                 ServerHttpResponse response = exchange.getResponse();
                 //303 状态码表示由于请求对应的资源存在着另一个 URI，应使用重定向
获取请求的资源
      response.setStatusCode(HttpStatus.SEE_OTHER);

response.getHeaders().set(HttpHeaders.LOCATION,"http://www.gmall.com/login.
html?originUrl="+request.getURI());
                 // 重定向到登录
      return response.setComplete();
             }
        }

        // 将 userId 传递给后端
    if (!StringUtils.isEmpty(userId)){
             request.mutate().header("userId",userId).build();
             // 将现在的 request 变成 exchange 对象
     return chain.filter(exchange.mutate().request(request).build());
        }
        return chain.filter(exchange);
    }
```

工具类中 AuthContextHolder 有获取用户 Id 的方法!

```
/**
 * 获取当前登录用户 id
 * @param request
 * @return
 */
public static String getUserId(HttpServletRequest request) {
    String userId = request.getHeader("userId");
    return StringUtils.isEmpty(userId) ? "" : userId;
}
```

3.3 在服务网关中判断用户登录状态

在网关中如何获取用户信息：

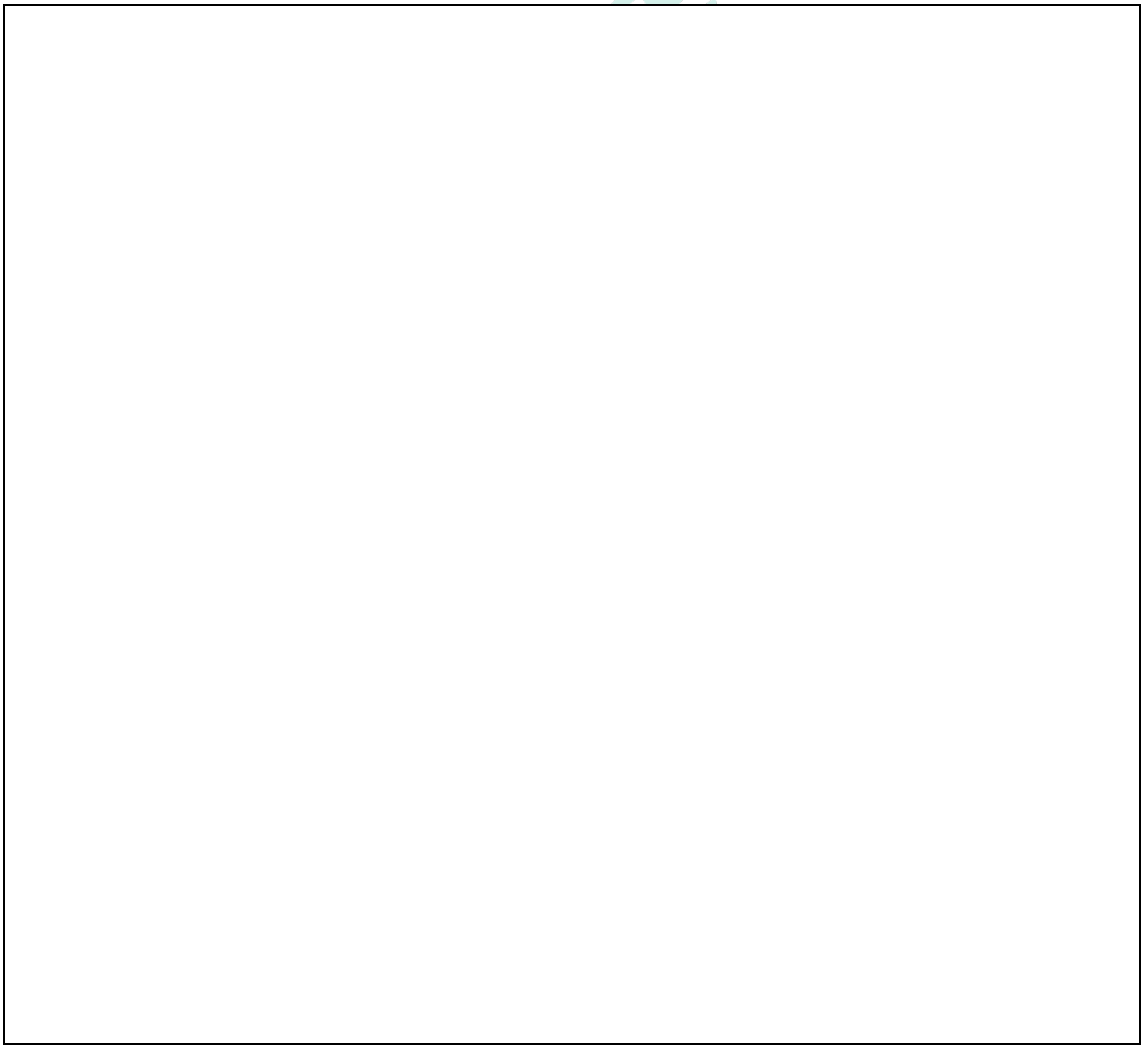
1，从 cookie 中获取（如：web 同步请求）

2，从 header 头信息中获取（如：异步请求）

如何判断用户信息合法：

登录时我们返回用户 token，在服务网关中获取到 token 后，我在到 redis 中去查看用户 id，如果用户 id 存在，则 token 合法，否则不合法，同时校验 ip，防止 token 被盗用。

3.3.1 取用户信息



```
/**
 * 获取当前登录用户 id
 * @param request
 * @return
 */
private String getUserId(ServerHttpRequest request) {
    String token = "";
    List<String> tokenList = request.getHeaders().get("token");
    if(null  != tokenList) {
        token = tokenList.get(0);
    } else {
        MultiValueMap<String,  HttpCookie>  cookieMultiValueMap  =
request.getCookies();
        HttpCookie cookie = cookieMultiValueMap.getFirst("token");
        if(cookie != null){
            token = URLDecoder.decode(cookie.getValue());
        }
    }
```

```
 if(!StringUtils.isEmpty(token)) {
        String                          userStr                          =
(String)redisTemplate.opsForValue().get("user:login:" + token);
        JSONObject userJson = JSONObject.parseObject(userStr);
        String ip = userJson.getString("ip");
        String curIp = IpUtil.getGatwayIpAddress(request);
        //校验token 是否被盗用
        if(ip.equals(curIp)) {
            return userJson.getString("userId");
        } else {
            //ip 不一致
            return "-1";
        }
    }
    return "";
}
```

3.3.2 输入信息 out 方法

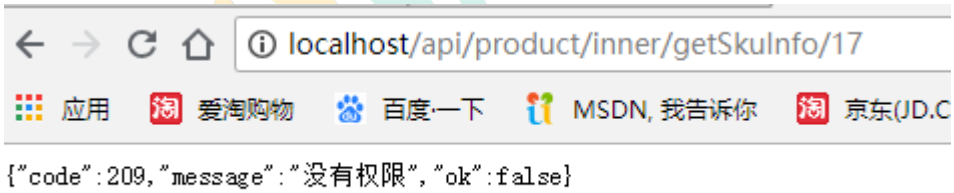
```
```

```
// 接口鉴权失败返回数据

private Mono<Void> out(ServerHttpResponse response,ResultCodeEnum
resultCodeEnum) {

    // 返回用户没有权限登录

  Result<Object> result = Result.build(null, resultCodeEnum);

    byte[] bits =
JSONObject.toJSONString(result).getBytes(StandardCharsets.UTF_8);

    DataBuffer wrap = response.bufferFactory().wrap(bits);

    response.getHeaders().add("Content-Type",
"application/json;charset=UTF-8");

    // 输入到页面

  return response.writeWith(Mono.just(wrap));

}
```

3.3.3 测试

1.通过网关访问内部接口，则不能访问！

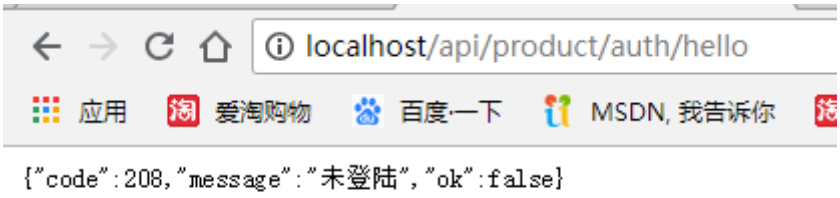
http://localhost/api/product/inner/getSkuInfo/17



2. 测试登录权限

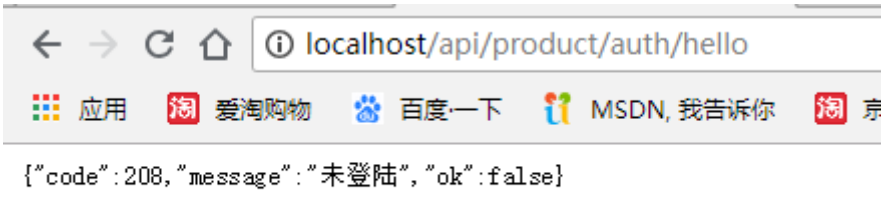
测试一:

未登录： http://localhost/api/product/auth/hello



登录完成之后继续测试！

登录：http://localhost/api/product/auth/hello

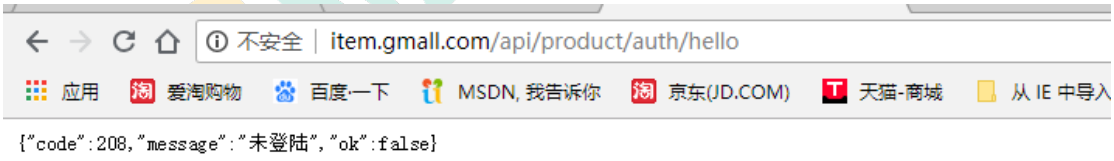


使用 localhost 访问，你登录或者不登录，都会提示未登录！

测试二：

用户在未登录情况下测试：

http://item.gmall.com/api/product/auth/hello



在上面的访问链接的时候，如果用户登录了，那么还会继续提示未登录！

404 表示资源没有！没有提示未登录！

原因：

测试一：访问资源的时候，没有获取到 userId

测试二：访问资源的时候，获取到了 userId

因为：我们登录成功的时候，将 token 放入了 cookie 中。在放入 cookie 的时候，我们给 cookie 设置了一个作用域。

return $.cookie('token', token, {domain: 'gmall.com', expires: 7, path: '/'})

测试一：使用的域名是 localhost，测试二：使用 item.gmall.com 包含 gmall.com

所以测试二是正确的！以后我们访问的时候，不会通过 localhost 访问，都是通过域名访问的！

3. 验证 Url 访问的是控制器

未登录直接访问：会弹出登录页面

http://list.gmall.com/list.html

4. 登录之后，然后在访问

会显示查询结果!

http://list.gmall.com/list.html

问题：https://blog.csdn.net/xjszsd/article/details/121746176