尚品汇商城复习

版本：V 1.0

登录模块

一、登录业务介绍

咱们的单点登录 是 多服务间的单点.

一个公司有多个项目,这些项目使用同一个账户就能登录,登录一个其他的不需要再登录了.

社交登录:使用微信可以登录多个平台.

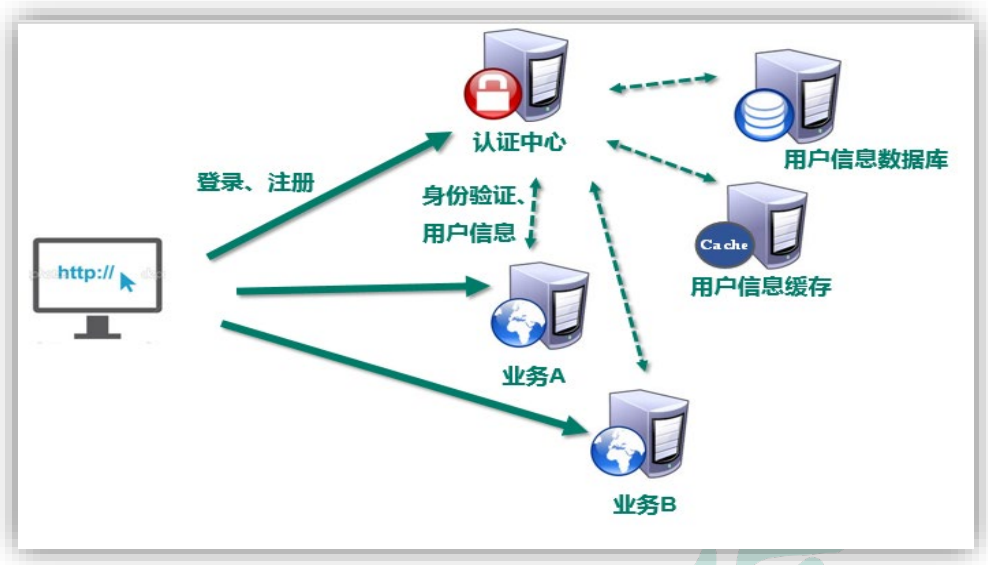
早期单一服务器，用户认证

缺点：单点性能压力，无法扩展

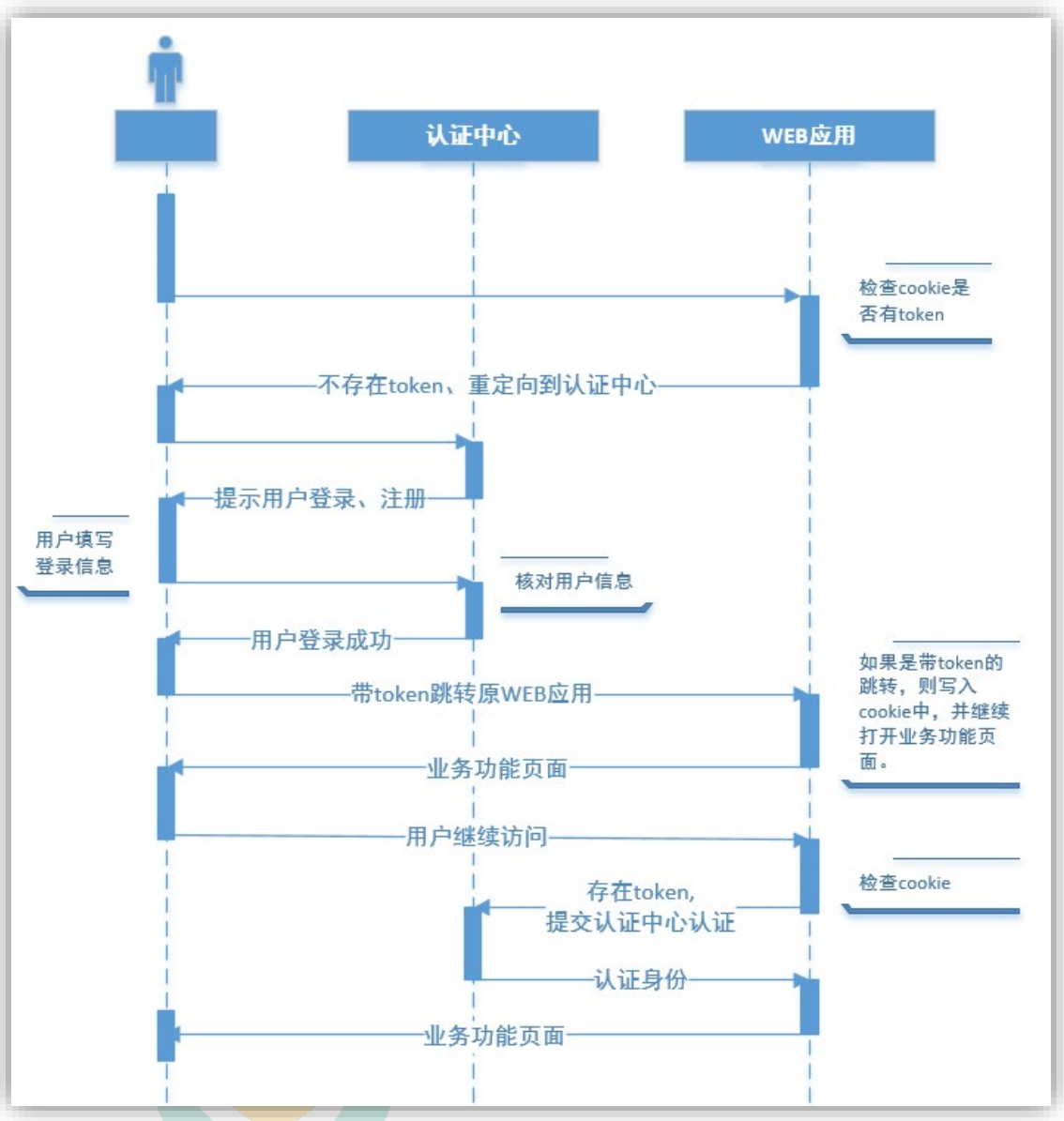
WEB 应用集群，session 共享模式

Tomcat 广播 session

分布式，SSO(single sign on)模式



业务流程图

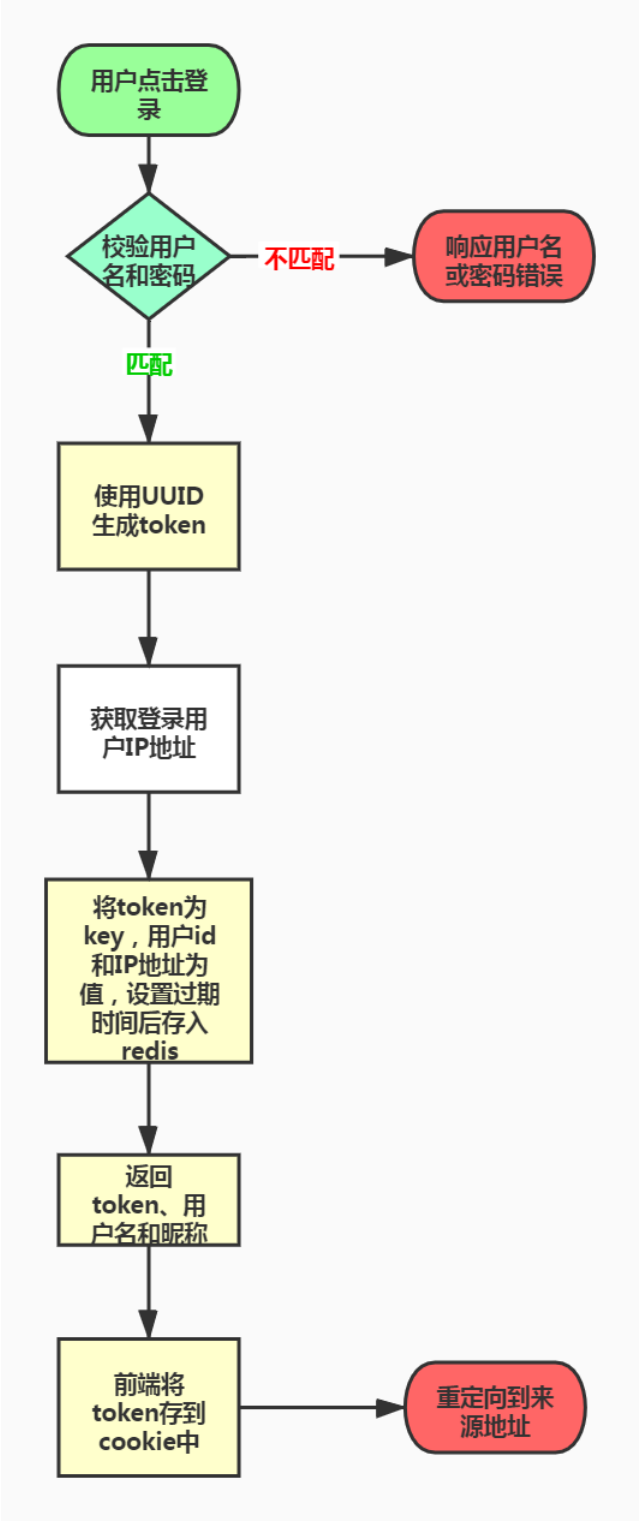


二、认证中心模块

数据库表：user_info！**密码应该是加密的！**

在设计密码加密方式时 一般是使用 MD5+盐的方式进行加密和解密。

1、 登录业务流程


用户点击登录
校验用户名和密码
不匹配
响应用户名或密码错误
匹配
使用UUID生成token
获取登录用户IP地址
将token为key，用户id和IP地址为值，设置过期时间后存入redis
返回token、用户名和昵称
前端将token存到cookie中
重定向到来源地址

2、业务实现

Web-all 工程

```
    package com.atguigu.gmall.all.controller;
/**
 * <p>
 * 用户认证接口
 * </p>
 *
 */
@Controller
@RequestMapping
public class PassportController {
    @GetMapping("login.html")
    public String login(HttpServletRequest request) {
        String originUrl = request.getParameter("originUrl");
        request.setAttribute("originUrl",originUrl);
        return "login";
    }
}
```

认证中心模块 service-user

```
package com.atguigu.gmall.user.controller;
/**
 * <p>
 * 用户认证接口
 * </p>
 */
@RestController
@RequestMapping("/api/user/passport")
public class PassportController {
    @Autowired
    private UserService userService;
    @Autowired
    private RedisTemplate redisTemplate;
    @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("name", info.getName());
            map.put("nickName", info.getNickName());
            map.put("token", token);
```

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

```
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;
    }
}
```

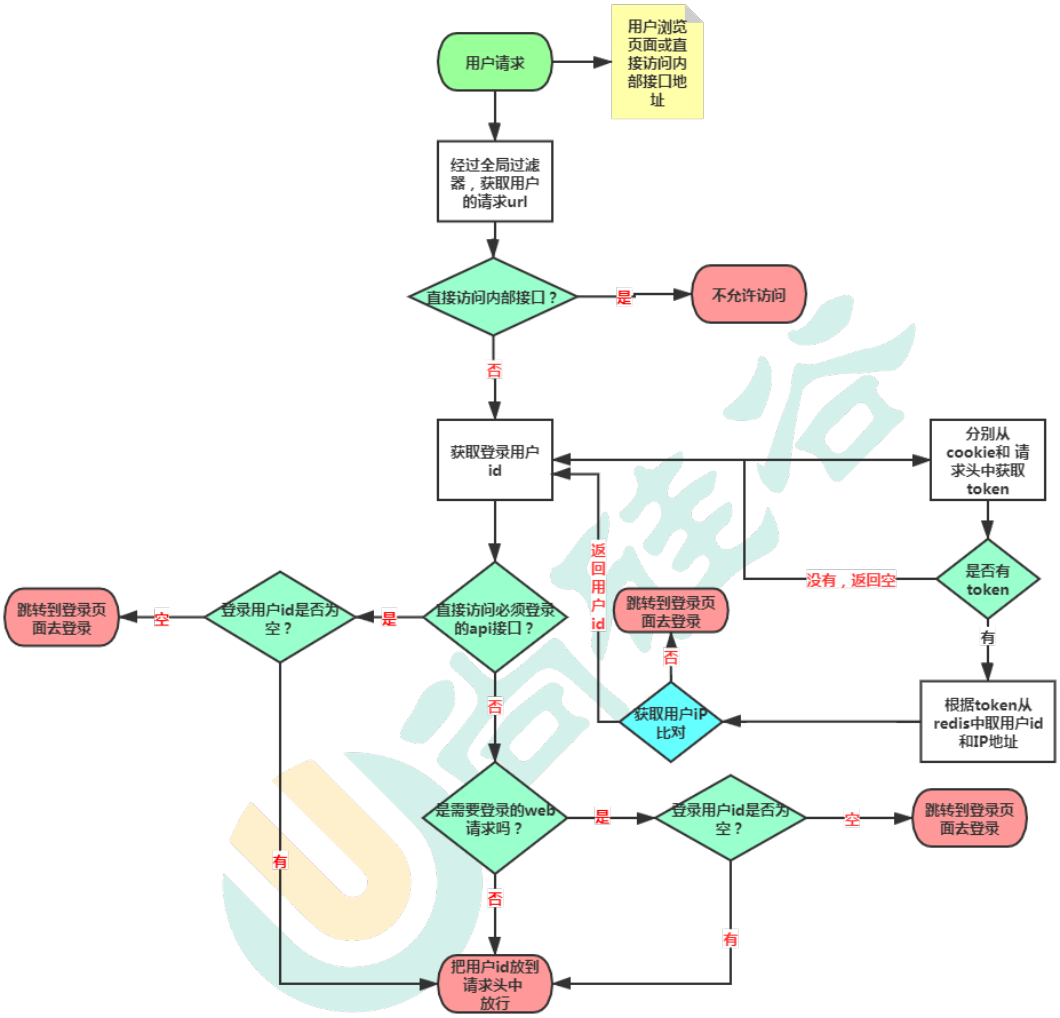
三、用户认证与服务网关整合

1、业务流程

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



2、业务实现

```
/**
 * <p>
 * 全局 Filter，统一处理会员登录与外部不允许访问的服务
 * </p>
 *
 */
@Component
public class AuthGlobalFilter implements GlobalFilter, Ordered {
```

```
    @Autowired
    private RedisTemplate redisTemplate;
    private AntPathMatcher antPathMatcher = new AntPathMatcher();
    //web 服务配置必须登录的 url
    private static String[] authUrls =
{"trade.html","myOrder.html","list.html"};
    @Override
    public Mono<Void> filter(ServerWebExchange exchange,
GatewayFilterChain chain) {
        ServerHttpRequest request = exchange.getRequest();
        String path = request.getURI().getPath();
        //内部服务接口，不允许外部访问
        if(antPathMatcher.match("/**/inner/**", path)) {
            ServerHttpResponse response = exchange.getResponse();
            return out(response);
        }
        String userId = this.getUserId(request);
        //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);
            }
        }
        // web 服务，同步请求，校验用户必须登录
        for(String url : authUrls) {
            if(path.indexOf(url) != -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 GET 方法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());
                return response.setComplete();
            }
        }
        //设置网关请求头
        String userTempId = this.getUserTempId(request);
        if(!StringUtils.isEmpty(userId)
|| !StringUtils.isEmpty(userTempId)) {
            if(!StringUtils.isEmpty(userId)) {
```

```
                request.mutate().header("userId", userId).build();
            }
            if(!StringUtils.isEmpty(userTempId)) {
                request.mutate().header("userTempId",
userTempId).build();
            }
            //将现在的 request 变成 exchange 对象
            return
chain.filter(exchange.mutate().request(request).build());
        }
        return chain.filter(exchange);
    }
    @Override
    public int getOrder() {
        return 0;
    }
    /**
     * api 接口鉴权失败返回数据
     * @param response
     * @return
     */
    private Mono<Void> out(ServerHttpResponse response) {
        Result result = Result.build(null,
ResultCodeEnum.LOGIN_AUTH);
        byte[] bits =
JSONObject.toJSONString(result).getBytes(StandardCharsets.UTF_8);
        DataBuffer buffer = response.bufferFactory().wrap(bits);
        //指定编码，否则在浏览器中会中文乱码
        response.getHeaders().add("Content-Type",
"application/json;charset=UTF-8");
        return response.writeWith(Mono.just(buffer));
    }
    /**
     * 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 userId =
(String)redisTemplate.opsForValue().get("user:login:" + token);
            return userId;
        }
        return "";
    }
    /**
     * 获取当前用户临时用户 id
     * @param request
     * @return
     */
    private String getUserTempId(ServerHttpRequest request) {
        String userTempId = "";
        List<String> tokenList =
request.getHeaders().get("userTempId");
        if(null  != tokenList) {
            userTempId = tokenList.get(0);
        } else {
            MultiValueMap<String, HttpCookie> cookieMultiValueMap
=  request.getCookies();
            HttpCookie cookie =
cookieMultiValueMap.getFirst("userTempId");
            if(cookie != null){
                userTempId = URLDecoder.decode(cookie.getValue());
            }
        }
        return userTempId;
    }
}
```

3、常见问题：

1、cookie 被禁用了能登录吗？-----不能.token

怎么解决的？ -------给用户提示：请您启用浏览器 Cookie 功能或更换浏览器

2、一顿绕

用户在 A 浏览器登录了 B 浏览器还用登录吗?

PC 客户端还用登录吗?

PC:电脑浏览器

客户端: 手机 APP

移动端: 手机浏览器

3、用户登录信息多久过期?

只要合理就行 2 小时 24 小时 7 天

咱们做的:设置有效时间 cookie+redis

看设置,

京东 PC 端设置 浏览器关闭 就退出登录.

APP:登录一次 只要你不清除 APP 是数据.

4、怎么防止 cookie 盗用（盗链）?

用户做什么行为 控制不了.

使用 IP 比对.

Token 存在 cookie 的时候不要见名之意