链滴

基于 Gitlab 进行开发团队管理的尝试——04.CAS 单点登录

作者：crick77

原文链接：https://ld246.com/article/1581130363431

来源网站：链滴

背景

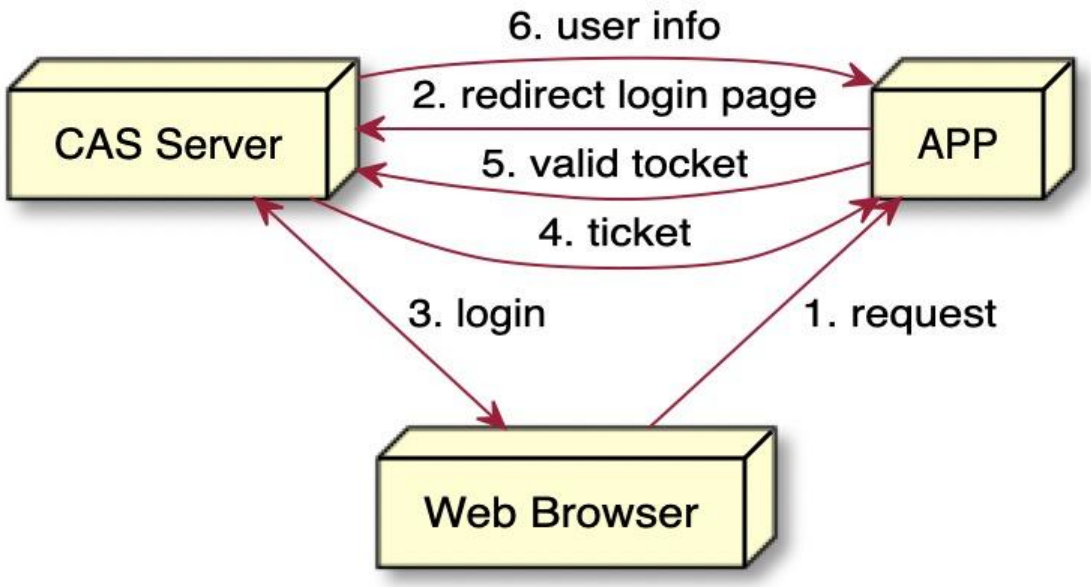
大部分公司内部有很多应用需要使用 CAS(Central Authentication Service,即:统一认证服务) 完成用登录验证。如果每个应用单独接入域账号验证，除了浪费工作量，安全性也得不到保障。

通用解决方案为部署一套 CAS 服务实现登录验证以及 SSO(Single Sign On) 单点登录。

相较于臃肿的开源项目解决方案，或者自己造一个轮子，其实还有一套轻量级的解决方案 -> **通过 Gitlab 的 applications 实现 CAS。**

CAS

网上相关资料很多，这里只描述基本的交互流程

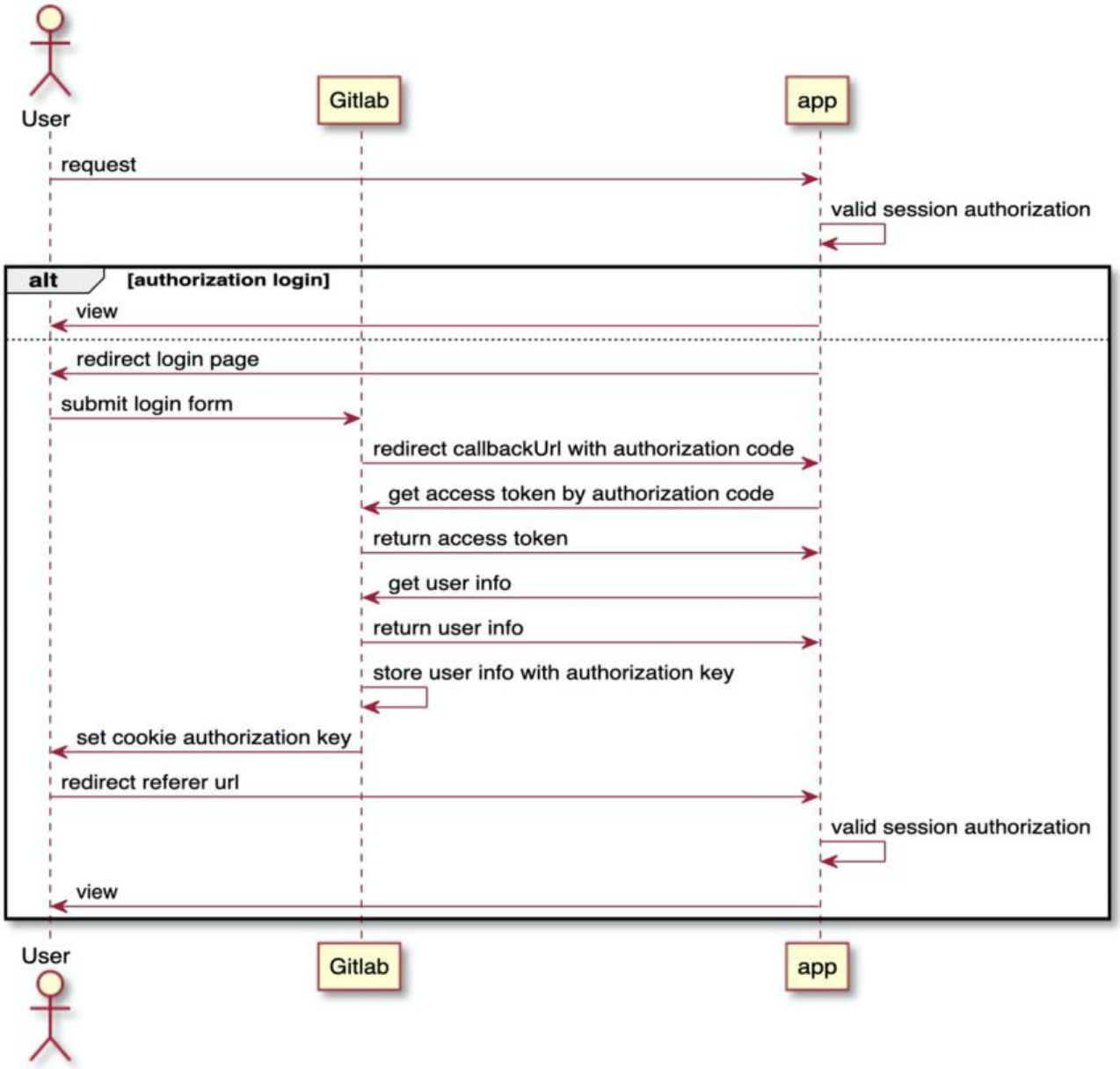


实现

将 GitLab 作为授权服务器，通过代码实现 GitLab Applications 交互，并调用 API 获取用户信息。

GitLab CAS

基本遵循 CAS 交互流程，部分字段命名不同。



代码实现

1. clientId & Secret 通过创建 Gitlab applications 获得
2. 通过浏览器转换，所以 callback 地址可以配置为 localhost
3. user 信息可以 store，也可以采用 **JWT** 方式
4. 代码为示例代码，实际使用过程中可以按需求封装 starter

```
@Controller
public class OauthController {

    private Logger logger = LoggerFactory.getLogger(this.getClass());

    @Value("${oauth2.server.url:https://gitlab.com}")
    private String gitlabServerUrl;
    @Value("${oauth2.client.id:xxx}")
    private String clientId;
    @Value("${oauth2.client.secret:xxxx}")
```

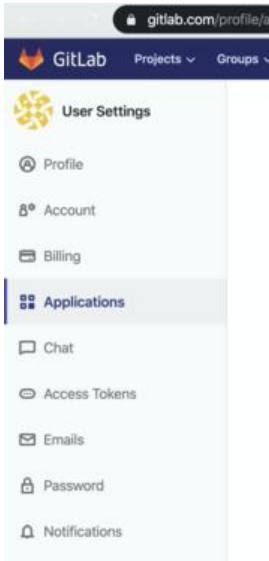
```
    private String clientSecret;
    @Value("${oauth2.client.callback.url:http://localhost:9000/callback}")
    private String callbackUrl;

    private static final String CURRENT_USER = "CurrentUser";
    private static final String AUTHORIZATION_KEY = "Authorization";
    private Map<String, User> userStore = new HashMap<>();
    private RestTemplate restTemplate = new RestTemplate();

    @GetMapping({"/main","/"})
    @ResponseBody
    public String main() {
        User user = (User) RequestContextHolder.getRequestAttributes().getAttribute(CURRENT
USER, RequestAttributes.SCOPE_SESSION);
        return "<html><body>hi:" + user.username + " This is Main</body></html>";
    }

    /**
     * 授权后redirect url
     * @param code 用于获取accessToken，只能使用一次
     */
    @GetMapping("/callback")
    public String callback(@RequestParam(value = "code", required = false) String code,
                           RedirectAttributes redirectAttributes,
                           HttpServletRequest request, HttpServletResponse response) {
        String referer = request.getParameter("referer");
        String accessToken = getAccessToken(code, buildCallbackUrl(referer));
        User user = getUser(accessToken);

        String uuid = UUID.randomUUID().toString();
        userStore.put(uuid, user);
        //set cookie
        response.addCookie(new Cookie(AUTHORIZATION_KEY, uuid));
        return "redirect:" + referer;
    }

    private String buildCallbackUrl(String referer) {
        return callbackUrl + "?referer=" + referer;
    }

    private User getUser(String accessToken) {
        return restTemplate.getForObject(gitlabServerUrl + "/api/v4/user?access_token=" + acce
sToken, User.class);
    }

    /**
     * 通过code去gitlab获取accessToken
     * @param code
     * @param redirectUri 回调地址，必须与授权时参数一致
     */
    private String getAccessToken(String code, String redirectUri) {
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_FORM_URLENCODED);
```

```
        MultiValueMap<String, String> params = new LinkedMultiValueMap<>();
        params.add("grant_type", "authorization_code");
        params.add("client_id", clientId);
        params.add("client_secret", clientSecret);
        params.add("code", code);
        params.add("redirect_uri", redirectUri);

        HttpEntity<MultiValueMap<String, String>> entity = new HttpEntity<>(params, headers
;

        ResponseEntity<JSONAccessTokenResponse> response =
                restTemplate.exchange(gitlabServerUrl + "/oauth/token",
                        HttpMethod.POST,
                        entity,
                        JSONAccessTokenResponse.class);
        return Objects.requireNonNull(response.getBody()).access_token;
    }

    @Configuration
    class WebConfig implements WebMvcConfigurer {
        @Override
        public void addInterceptors(InterceptorRegistry registry) {
            registry.addInterceptor(
                    new HandlerInterceptorAdapter() {
                        @Override
                        public boolean preHandle(HttpServletRequest request, HttpServletResponse re
ponse, Object handler) throws Exception {
                            Optional<String> authorizationKeyOp = Arrays.stream(request.getCookies())
                                    .filter(it->it.getName().equals(AUTHORIZATION_KEY))
                                    .map(Cookie::getValue)
                                    .findAny();
                            if (authorizationKeyOp.isPresent()) {
                                // 授权信息存在，获取user信息放入session
                                RequestContextHolder.getRequestAttributes().setAttribute(CURRENT_USE
, userStore.get(authorizationKeyOp.get()), RequestAttributes.SCOPE_SESSION);
                                return super.preHandle(request, response, handler);
                            } else {
                                // 授权信息不存在，去gitlab进行验证
                                String referer = request.getRequestURL().toString();
                                String redirectUri = URLEncoder.encode(buildCallbackUrl(referer), "utf-8");
                                String gitlabAuthUrl = gitlabServerUrl + "/oauth/authorize?response_typ
=code&redirect_uri=" + redirectUri + "&client_id=" + clientId;
                                logger.info("gitlabAuthUrl:{}", gitlabAuthUrl);
                                response.sendRedirect(gitlabAuthUrl);
                                return false;
                            }
                        }
                    })
                    .addPathPatterns("/main", "/test");
        }
    }

    static class JSONAccessTokenResponse implements Serializable {
        public String access_token;
```

```
    }

    static class User implements Serializable {
        public String name;
        public String username;
    }
}
```

新建 GitLab applications

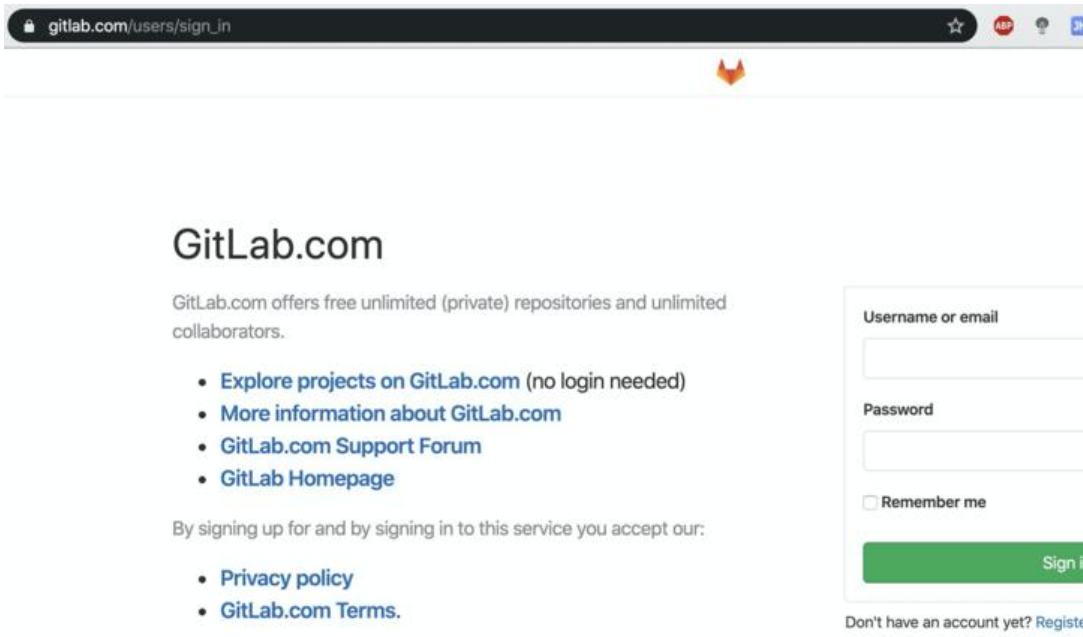
1. User Settings -> Applications 中填写名称及 Redirect URI, 只需要勾选 Confidential



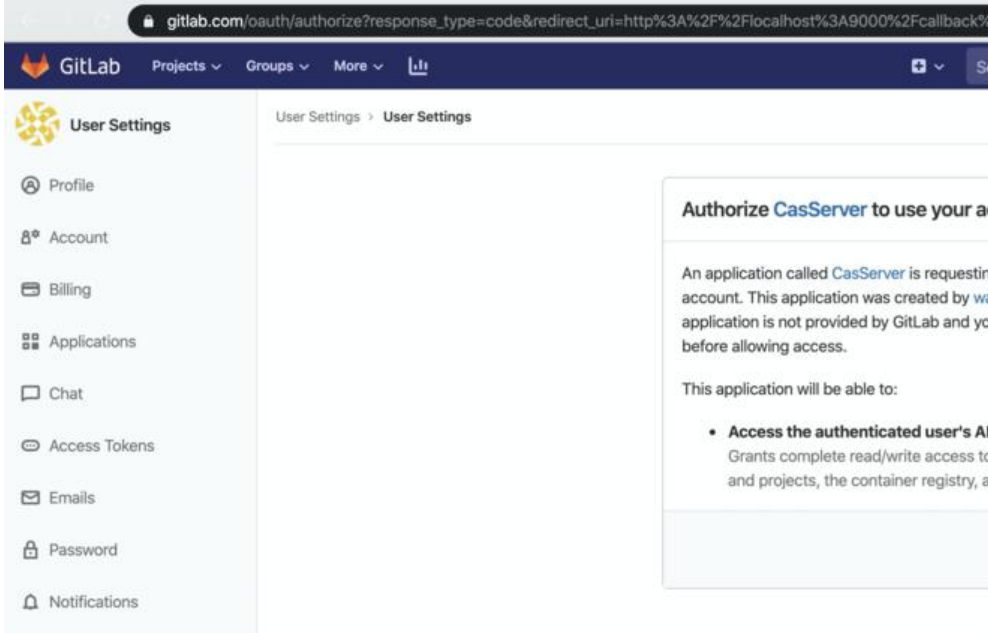
2. 点击保存并记录 ClientId 及 Secret

效果

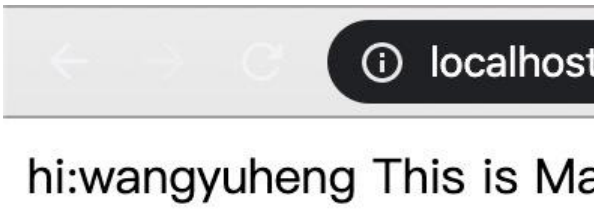
1. 浏览器访问 http://localhost:9000/main
2. 跳转至 Gitlab 登录页面



3. 同意授权(此操作只需进行一次)

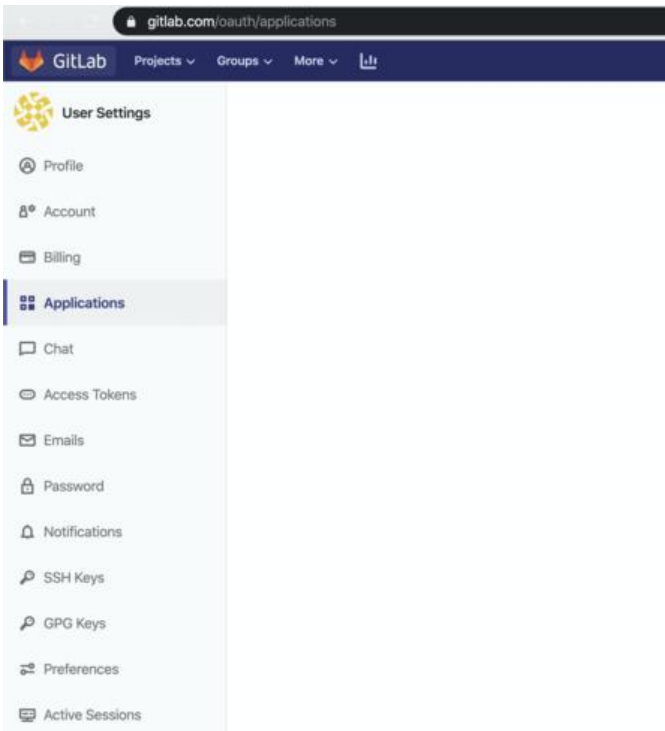


4. 自动跳转至 http://localhost:9000/main 并获取登录账号信息



5. 刷新页面为登陆状态，无需重新登陆。登录信息已在 Cookie 中，相同域名下无需重新登陆。

6. 在 Gitlab Applications 中可以查看授权状态并取消授权



FAQs

原文链接：基于 Gitlab 进行开发团队管理的尝试——04.CAS 单点登录

1. GitLab 报错 The redirect url included is not valid -> 确认传递的 callback url 的 path 及参与 applications 中填写的一致