链滴

Solo 登录验证社区账号

作者：88250

原文链接：https://ld246.com/article/1554262218912

来源网站：链滴

本文是《Solo 从设计到实现》的一个章节，该系列文章将介绍 Solo 这款 Java 博客系统是如何从无有的，希望大家能通过它对 Solo 从设计到实现有个直观地了解、能为想参与贡献的人介绍清楚项目也希望能为给~~重复发明~~重新定义博客系统的人做个参考
eart

社区登录重定向

Solo 是通过黑客派社区账号实现登录验证的，用户登录时重定向到社区登录页面，登录完成以后重向回 Solo。

方法 OAuthProcessor#redirectAuth ：

```
/**
 * Redirects to HacPai auth page.
 *
 * @param context the specified context
 */
@RequestProcessing(value = "/login/redirect", method = HttpMethod.GET)
public void redirectAuth(final RequestContext context) {
    String referer = context.param("referer");
    if (StringUtils.isBlank(referer)) {
        referer = Latkes.getServePath();
    }

    final String state = RandomStringUtils.randomAlphanumeric(16);
    STATES.put(state, referer);

    final String loginAuthURL = "https://hacpai.com/login?goto=" + Latkes.getServePath() + "/
ogin/callback";
    final String path = loginAuthURL + "&state=" + state + "&v=" + Server.VERSION;
    context.sendRedirect(path);
}
```

登录后重定向回调

1. 验证 state 防止重放攻击
2. 处理 Referer 获取参数以及后续跳转
3. 根据 Open Id（HacPai User Id）查库，并根据系统状态处理（初始化、添加访客用户等）

方法 OAuthProcessor#authCallback ：

```
/**
 * OAuth callback.
 *
 * @param context the specified context
 */
public synchronized void authCallback(final RequestContext context) {
    String state = context.param("state");
    final String referer = STATES.get(state);
    if (null == referer) {
        context.sendError(400);
        return;
```

原文链接：Solo 登录验证社区账号

```
}
STATES.remove(state);

final Response response = context.getResponse();
final Request request = context.getRequest();
final String accessToken = context.param("access_token");
final JSONObject userInfo = Solos.getUserInfo(accessToken);
if (null == userInfo) {
    LOGGER.log(Level.WARN, "Can't get user info with token [" + accessToken + "]");
    context.sendError(401);
    return;
}

final String userId = userInfo.optString("userId");
final String userName = userInfo.optString(User.USER_NAME);
final String userAvatar = userInfo.optString("avatar");

JSONObject user = userQueryService.getUserByGitHubId(userId);
if (null == user) {
    if (!initService.isInited()) {
        final JSONObject initReq = new JSONObject();
        initReq.put(User.USER_NAME, userName);
        initReq.put(UserExt.USER_AVATAR, userAvatar);
        initReq.put(UserExt.USER_B3_KEY, userName);
        initReq.put(UserExt.USER_GITHUB_ID, userId);
        initService.init(initReq);
    } else {
        final JSONObject addUserReq = new JSONObject();
        addUserReq.put(User.USER_NAME, userName);
        addUserReq.put(UserExt.USER_AVATAR, userAvatar);
        addUserReq.put(User.USER_ROLE, Role.VISITOR_ROLE);
        addUserReq.put(UserExt.USER_GITHUB_ID, userId);
        addUserReq.put(UserExt.USER_B3_KEY, userName);
        try {
            userMgmtService.addUser(addUserReq);
        } catch (final Exception e) {
            LOGGER.log(Level.ERROR, "Registers via oauth failed", e);
            context.sendError(500);
            return;
        }
    }
} else {
    user.put(User.USER_NAME, userName);
    user.put(UserExt.USER_AVATAR, userAvatar);
    try {
        userMgmtService.updateUser(user);
    } catch (final Exception e) {
        LOGGER.log(Level.ERROR, "Updates user name failed", e);
        context.sendError(500);
        return;
    }
}

user = userQueryService.getUserByName(userName);
```

```
    if (null == user) {
        LOGGER.log(Level.WARN, "Can't get user by name [" + userName + "]");
        context.sendError(404);
        return;
    }

    Solos.login(user, response);
    Statics.clear();
    context.sendRedirect(referer);
    LOGGER.log(Level.INFO, "Logged in [name={}, remoteAddr={}] with oauth", userName, Req
ests.getRemoteAddr(request));
}
```

关于 GitHub Id

以前的版本是使用 GitHub OAuth 登录的，后来切换为社区登录了。但是数据库字段还有实现方面有做迁移，依然保留 GitHub Id 命名，现在这个字段存储的是黑客派用户 id。