用户管理

当前登录用户信息可通过 `BJLRoom` 的 `loginUser` 属性获取，获取时机为进入教室成功（监听到 `enterRoomSuccess` ）之后。

在线用户信息管理
加载在线用户信息
监听用户进入、退出教室
主讲人切换
踢出教室用户管理

1. 在线用户列表

在线用户信息列表采用分页加载，参考 `BJLOnlineUsersVM` 。在线的活跃用户信息会在教室进入成功后自动请求，当 `BJLOnlineUsersVM` 的 `activeUsersSynced` 状态变成 YES 之后，活跃的用户数据已经加载完成，可以使用了。通过 `BJLOnlineUsersVM` 的 `onlineTeacher` 和 `currentPresenter` 可以快捷地获取到当前的老师和主讲人。

- 获取当前登录用户信息。

```
// 当前登录用户信息
BJLUser *user = self.room.loginUser;
```

- 监听在线用户变化。

`onlineUsersTotalCount` 和 `onlineUsers` 都能作为当前教室人数的参考。当教室内人数少于 100 人时，并且已经加载完教室内的全部用户，即 `hasMoreOnlineUsers` 为 NO 时，任何用户的退出和进入都会更新 `onlineUsers` 的内容，此时可以直接使用 `onlineUsers` 的数量作为当前用户数。否则，在 `onlineUsers` 的内容不是完整的数据，或者人数过多时，教室内的人数可以通过监听 `onlineUsersTotalCount` 获取，该属性是教室内规律更新的用户数通知，不是实时的，有滞后性。

```
[self bjl_kvo:BJLMakeProperty(self.room.onlineUsersVM, onlineUsers)
    observer:^BOOL(id _Nullable value, id _Nullable oldValue, BJLPropertyChange * _Nullable change) {
    bjl_strongify(self);
    [self updateTitleWithOnlineUsersTotalCount];
    [self.tableView reloadData];
    return YES;
}];
```

- 加载更多用户。

```
/**
 加载更多在线用户
 #discussion 加载成功更新 `onlineUsers`
 #discussion 参考 `hasMoreOnlineUsersofGroup:` 来获取某一个group是否可以加载更多
 #param count 传 0 默认 20、最多 30
 #return BJLError:
 BJLErrorCode_invalidCalling    错误调用，如 `hasMoreOnlineUsersofGroup:` 为 NO 时调用此方法
```

```
*/
 if (self.room.onlineUsersVM.hasMoreOnlineUsers
    && [self atTheBottomOfTableView]) {
    [self.room.onlineUsersVM
loadMoreOnlineUsersWithCount:20

groupID:self.room.loginUser.groupID];
 }
```

2. 用户进入、退出

- 监听用户进入教室。

```
[self
bjl_observe:BJLMakeMethod(self.room.onlineUsersV
 onlineUserDidEnter:)
        observer:^BOOL(BJLUser *user) {
   bjl_strongify(self);
   NSlog(@"onlineUser in: %@", user.name);
   return YES;
}];
```

- 监听用户退出教室。

```
[self
bjl_observe:BJLMakeMethod(self.room.onlineUsersV
 onlineUserDidExit:)
        observer:^BOOL(BJLUser *user) {
   bjl_strongify(self);
   NSLog(@"onlineUser out: %@", user.name);
   return YES;
}];
```

- 单独某个用户音视频变化的通知，可用于独立窗口绑定用户音视频

```
/**
 用户更新音视频状态
 */
-(BJLObservable)didRecieveUserStateUpdateWithUse
(NSString *)userNumber
                                audioState:
(BJLUserMediaState)audioState
                                videoState:
(BJLUserMediaState)videoState;
```

3. 分组教室信息

分组教室可以获取当前用户所在的组的用户信息，老师可以看到所有小组的成员，后台支持配置是否显示其他分组的用户信息。对于有分组的教室，每个小组会有部分功能是小组之间分离的，比如每个小组都可以有不同的公告，分组标识颜色，分组的发言权限控制，分组的点赞数据等。

- 教室内分组信息。

```
@property (nonatomic, readonly, copy, nullable)
NSArray<BJLUserGroup *> *groupList;
```

- 获取各个分组的颜色。

```
/**
 教室内分组颜色
 #param groupID 对应分组的 ID
 #return 十六进制 RGB 色值字符串，如 `#FFFFFF`
 */
```

```
NSString *color = [self.room.onlineUsersVM getGroupColorWithID:group.groupID]
```

- 分组人数变化通知。

```
/**
 学生分组人数变化
 #param groupCountDic <groupID, count> 提供分组及其人数变化
 */
-(BJLObservable)onlineUserGroupCountDidChange:(NSDictionary *)groupCountDic;

example:
 [self bjl_observe:BJLMakeMethod(self.room.onlineUsersVM onlineUserGroupCountDidChange:)
        observer:^BOOL(NSDictionary *groupCountDic) {
    // bjl_strongify(self);
    [groupCountDic enumerateKeysAndObjectsUsingBlock:^(NSString *groupID, NSNumber *count, BOOL * _Nonnull stop) {
        NSLog(@"Group %@ has %@ users", groupID, count);
    }];
    return YES;
}];
```

```
/**
 批量更新用户分组信息变化
```

```
 #param groupInfo 学生所在分组的信息, nil表示分组被移除
 */
-(BJLObservable)onlineUserGroupInfoDidChangeWith:(NSArray<NSString *> *)userNumbers groupInfo:(nullable BJLUserGroup *)groupInfo;
```

4. 音视频分组

- 小班课场景提供了音视频分组的逻辑，老师可以将学生区分为不同组，只看组内音视频，而其他业务不受影响。
- 音视频分组和常规分组互斥。不可同时存在。

```
@property (nonatomic, readonly) BOOL isMediaGroup;
```

5. 切换主讲人

切换主讲人，主讲人只能由教室内最大权限的老师来设置，之后老师本人或者助教才能被设置为主讲人, 参考 `BJLOnlineUsersVM` 。教室内默认老师身份的用户作为主讲人，可以监听 `currentPresenter` 获取主讲人的变化。

```
/** 切换主讲
 #discussion 1. 主讲人只能由老师设置，2. 之后老师本人或者助教才能被设置为主讲人
 #param userID  老师或助教的 userID
 #return BJLError:
 BJLErrorCode_invalidCalling    不支持切换主讲，参考 `room.featureConfig.canChangePresenter`
 BJLErrorCode_invalidArguments  错误参数
 BJLErrorCode_invalidUserRole   错误权限，要求老师权限
```

```
 */
BJLError *error = [self.room.onlineUsersVM requestChangePresenterWithUserID:userID];
```

6. 踢出的用户列表

- 踢出用户。

```
/**
 踢出用户
 #param userID 学生ID
 #return BJLError:
 BJLErrorCode_invalidArguments  错误参数；
 BJLErrorCode_invalidCalling   错误调用，如要踢出的用户是老师或助教；
 BJLErrorCode_invalidUserRole  错误权限，要求老师或助教权限。
 */
 BJLError *error = [self.room.onlineUsersVM blockUserWithID:user.ID];

 /**
 学生被加入黑名单列表
 #discussion 被踢出的用户非当前用户
 #param blockedUser 被踢出的学生
 */
- (BJLObservable)didBlockUser:(BJLUser *)blockedUser;
```

- 解除用户被踢出的状态。

```
/**
 解除用户被踢出的状态
 #param userNumber userNumber
```

```
#return BJLError:
BJLErrorCode_invalidArguments  错误参数；
BJLErrorCode_invalidCalling   错误调用，如要踢出的用户是老师或助教；
BJLErrorCode_invalidUserRole   错误权限，要求老师或助教权限。
*/
BJLError *error = [self.room.onlineUsersVM freeBlockedUserWithNumber:user.number];

/**
用户黑名单被解除
#param userNumber userNumber
*/
- (BJLObservable)didFreeBlockedUserWithNumber:(NSString *)userNumber;
```

- 加载被踢出的用户列表

```
/** 加载黑名单列表 */
- (void)loadBlockedUserList;
/**
返回黑名单列表
#param userList 用户列表
*/
- (BJLObservable)didReceiveBlockedUserList:(NSArray<BJLUser *> *)userList;

example:

[self bjl_observe:BJLMakeMethod(self.room.onlineUsersV didReceiveBlockedUserList:)
```

```
        observer:^BOOL(NSArray<BJLUser *> *userList) {
    bjl_strongify(self);
    self.blockedUserList = [userList mutableCopy];
    return YES;
}];
```

- 解禁全部黑名单

```
/**
 解除全部用户黑名单
 #return BJLError:
 BJLErrorCode_invalidArguments  错误参数；
 BJLErrorCode_invalidCalling   错误调用，如要踢出的用户是老师或助教；
 BJLErrorCode_invalidUserRole   错误权限，要求老师或助教权限。
 */
- (nullable BJLError *)freeAllBlockedUsers;

/** 所有黑名单被解除 */
- (BJLObservable)didFreeAllBlockedUsers;
```