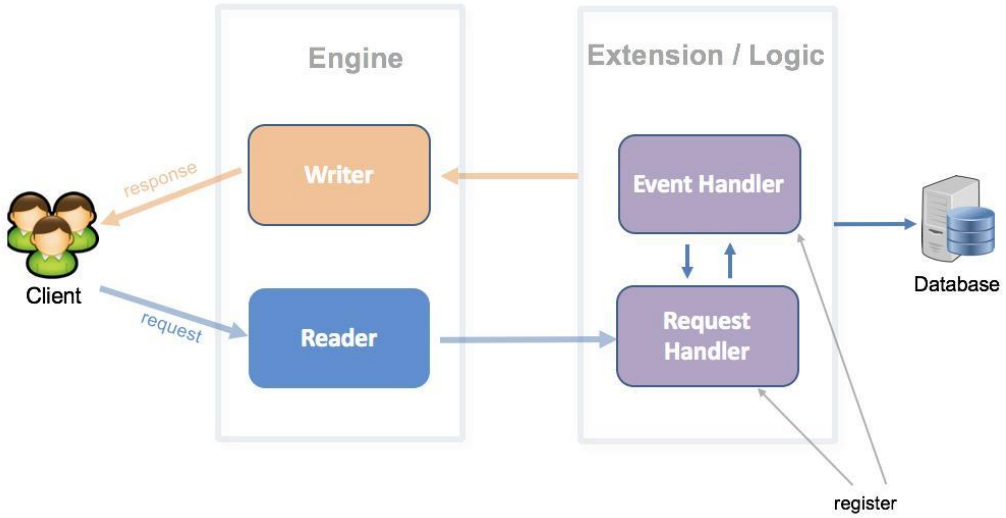
BitZero Engine Document

1. Giới thiệu BitZero

1.1 Flow tổng quan



Flow tổng quan của 1 gói tin từ client đến server và server gửi lại client

Khi một gói tin được gửi đến server, mỗi gói tin sẽ có một mã cmdId riêng biệt. Dựa trên mã cmdId này, engine sẽ đẩy đến các Handler tương ứng. Xem phần Xử lý gói tin đến để rõ hơn

Trong quá trình làm việc, các bạn chỉ cần quan tâm đến tầng Logic (Extension)

Ở Engine BitZero có 3 thành phần chính

- Reader: xử lý gói tin đến

- Writer: gửi gói về cho client

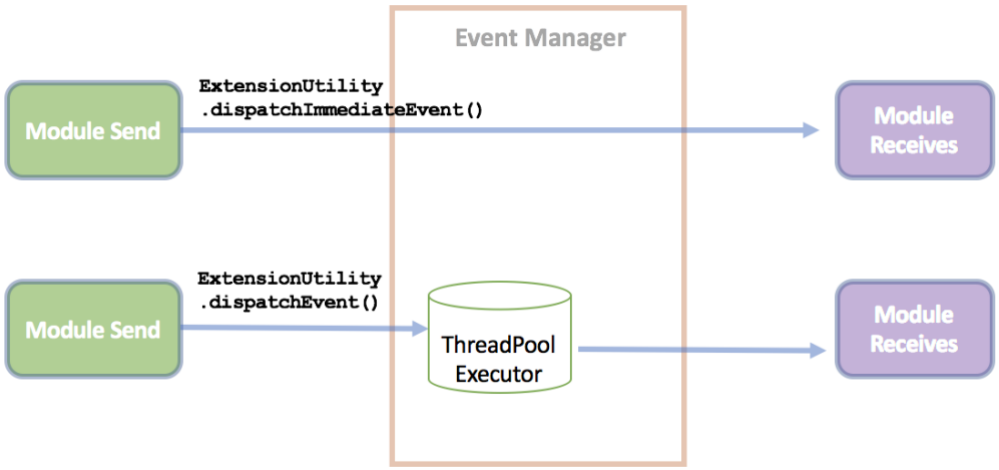
- Event Manager: hệ thống xử lý Sự kiện

1.2 Hệ thống xử lý Sự kiện

Sau khi xử lý 1 action, bạn muốn module khác nhận được thông tin và xử lý, bạn có thể bắn ra một sự kiện, và nhận ở module kia. Các bước cần làm

- Định nghĩa tên sự kiện
- Đăng ký nhận sự kiện cho Module Receive
- Bắn ra sự kiện từ Module Send
- Đăng ký xử lý trong Module Receive

Có 2 phương thức bắn ra sự kiện



- ExtensionUtility.dispatchEvent(): Sự kiện sẽ được đẩy vào một ThreadpoolExecutor và gửi đến các listener là các module đã đăng ký nhận event này
- ExtensionUtility.dispatchImmediateEvent() : cũng giống như dispatchEvent() nhưng sự kiện được thực thi trực tiếp trong cùng thead gửi

Ví dụ

Khi muốn xử lý một action ngay sau khi user login thành công (ví dụ cập nhật bảng xếp hạng …) ở trong module DemoHandler

- Bước 1: Định nghĩa tên sự kiện trong DemoEventType, class này extend từ IBZEventType

```
public enum DemoEventType implements IBZEventType {
    CHANGE_NAME,
    LOGIN_SUCCESS;

    private DemoEventType() {
    }
}
```

- Bước 2: đăng ký nhận sự kiện trong module DemoHandler

```
/**
 *  this method automatically loaded when run the program
 *  register new event, so the core will dispatch event type to this class
 */
public void init() {

    getParentExtension().addEventListener(DemoEventType.LOGIN_SUCCESS,  listener: this);
}
```

- Bước 3: bắn ra event khi user login thành công LoginSuccessHandler.onLoginSuccess()

```
/**
 * send login success to client
 * after receive this message, client begin to send game logic packet to server
 */
ExtensionUtility.instance().sendLoginOK(user);

/**
 * dispatch event here
 */
Map evtParams = new HashMap();
evtParams.put(DemoEventParam.USER, user);
evtParams.put(DemoEventParam.NAME, user.getName());
ExtensionUtility.dispatchEvent(new BZEvent(DemoEventType.LOGIN_SUCCESS, evtParams));
```

có thể truyền params vào trong event này

dispatch ra sự kiện, để các handler khác bắt

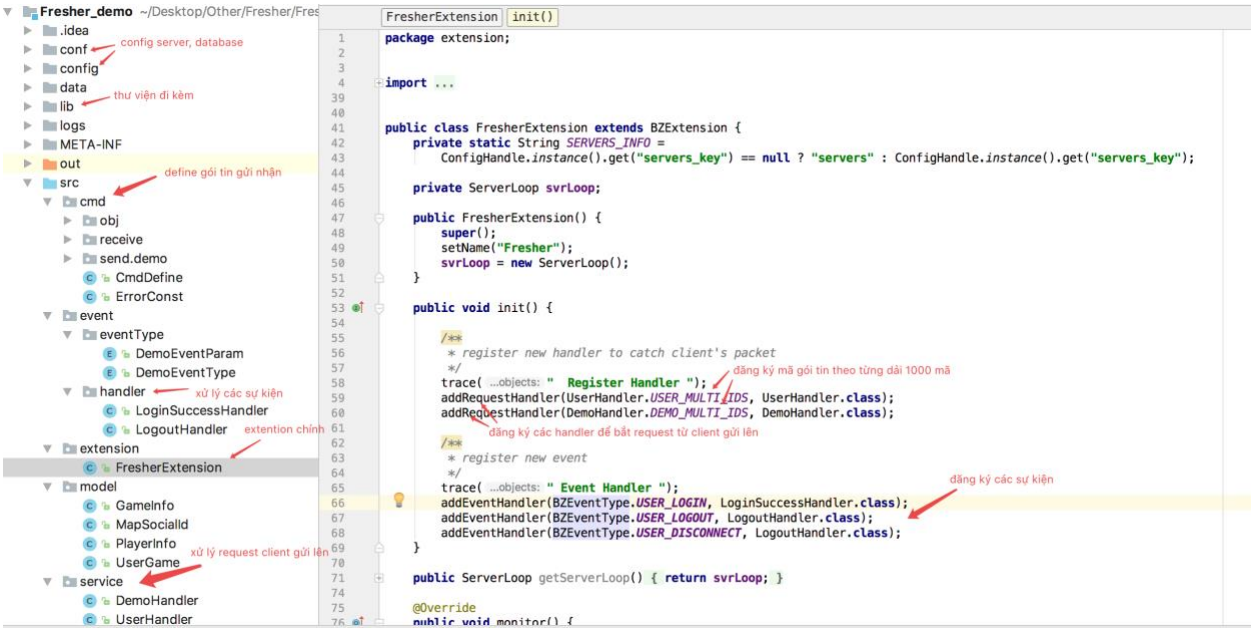
- Bước 4: đăng ký xử lý trong hàm handleServerEvent trong DemoHandler

```
/**
 * events will be dispatch here
 */
public void handleServerEvent(IBZEvent ibzevent) {
    if (ibzevent.getType() == DemoEventType.LOGIN_SUCCESS) {
        this.processUserLoginSuccess((User)ibzevent.getParameter(DemoEventParam.USI
    }
}
```

2. Sử dụng BitZero và cấu trúc project server

Engine được đặt trong lib bz-allinone.jar, khi chạy server mọi người add thêm lib này vào

2.1 Cấu trúc project



2.2 User login

Mặc định gói tin đầu tiên từ client gửi lên sẽ gọi vào FresherExtention.doLogin(), gói tin này sẽ xử lý đăng nhập cho user

```
public void doLogin(short cmdId, ISession session, DataCmd objData) {
    RequestLogin reqGet = new RequestLogin(objData);
    reqGet.unpackData();

    try {

        UserInfo uInfo = getUserInfo(reqGet.sessionKey, reqGet.userId, session.getAddress());
        User u = ExtensionUtility.instance().canLogin(uInfo,  password: "", session);
        if (u!=null)
            u.setProperty("userId", uInfo.getUserId());
    } catch (Exception e) {
        Debug.warn( ...objects: "DO LOGIN EXCEPTION " + e.getMessage());
        Debug.warn(ExceptionUtils.getStackTrace(e));
    }

}
```

hàm này để login user vào hệ thống, sau khi thành công sẽ dispatch ra sự kiện BZEventType.USER_LOGIN và sẽ được gọi đến LoginSuccessHandler, do đã đăng ký

trong conf/cluster.properties, chỉnh custom_login bằng một trong các tham số

```
# 1: social, 2: set direct, 3: autoincrement
custom_login=2
```

```
private UserInfo getUserInfo(String username, int userId, String ipAddress) throws Exception {
    int customLogin = ServerConstant.CUSTOM_LOGIN;
    switch(customLogin){
        case 1: // login zingme
            return ExtensionUtility.getUserInfoFormPortal(username);
        case 2: // set direct userid
            return GuestLogin.setInfo(userId, username: "Fresher_" + userId);
        default: // auto increment
            return GuestLogin.newGuest();
    }
}
```

Trong đó:

- custom_login = 1: login theo các mạng xã hội, ở build này sẽ làm với mạng xã hội zingme
- custom_login = 2: server sẽ set trực tiếp tài khoản login là tài khoản trong trường userId client gửi lên
- custom_login = 3: mỗi lần login, server sẽ tự tạo tài khoản mới

Thông tin gói RequestLogin

```
public class RequestLogin extends BaseCmd {
    public String sessionKey = "";
    public int userId = 0;
    public RequestLogin(DataCmd dataCmd) {
        super(dataCmd);
    }

    @Override
    public void unpackData() {
        ByteBuffer bf = makeBuffer();
        try {
            sessionKey = readString(bf);
            userId = readInt(bf);
        } catch (Exception e) {

        }
    }

}
```

Tuỳ vào cách login của user mà set giá trị custom_login và client gửi lên các tham số tương ứng

2.3 User disconnect

Khi client mất kết nối với server, server sẽ bắn ra sự kiện USER_DISCONNECT. Trong FresherExtension có đăng ký bắt sự kiện này ở class LogoutHandler

```
/**
 * register new event
 */
trace( ...args: " Event Handler ");
addEventHandler(BZEventType.USER_LOGIN, LoginSuccessHandler.class);
addEventHandler(BZEventType.USER_LOGOUT, LogoutHandler.class);
addEventHandler(BZEventType.USER_DISCONNECT, LogoutHandler.class);
```

2.4 Kết nối với Client

2.4.1 Xử lý gói tin đến

Mỗi gói tin connect từ client lên server sẽ có một mã cmdId khác nhau. Hệ thống sẽ chia các mã gói tin cmdId theo từng dải 1000 gói để xử lý.

Khi đăng ký handler, sẽ truyền vào một requestId là bội của 1000. Các cmdId bắt đầu từ requestId đến requestId+999 sẽ được đẩy vào handler này để xử lý

```
/**
 * register new handler to catch client's packet
 */
trace( ...args: "  Register Handler ");
addRequestHandler(UserHandler.USER_MULTI_IDS, UserHandler.class);
addRequestHandler(DemoHandler.DEMO_MULTI_IDS, DemoHandler.class);
```

Ví dụ khi bạn đăng ký DemoHandler, DemoHander.DEMO_MULTI_IDS có giá trị 2000 thì các gói tin từ 2000-2999 sẽ được được đẩy vào DemoHandler để xử lý

Trong DemoHandler, override lại hàm handleClientRequest, trong đây case mã gói tin để xử lý

```java
@Override
/**
 * this method handle all client requests with cmdId in range [1000:2999]
 *
 */
public void handleClientRequest(User user, DataCmd dataCmd) {
    try {
        switch (dataCmd.getId()) {
            // get username
            case CmdDefine.GET_NAME:
                processGetName(user);
                break;
            // set username
            case CmdDefine.SET_NAME:
                RequestSetName set = new RequestSetName(dataCmd);
                processSetName(set, user);
                break;
            case CmdDefine.MOVE:
                RequestMove move = new RequestMove(dataCmd);
                processMove(user, move);
                break;
        }

    } catch (Exception e) {
        logger.warn("DEMO HANDLER EXCEPTION " + e.getMessage());
        logger.warn(ExceptionUtils.getStackTrace(e));
    }

}
```

Đối với các gói tin từ client gửi lên, gói tin sẽ extend từ BaseCmd, và override lại hàm unpackData() để đọc theo thứ tự các kiểu dữ liệu, ví dụ ở đây đọc 1 biến short

```java
public class RequestMove extends BaseCmd{
    private short direction;
    public RequestMove(DataCmd dataCmd) {
        super(dataCmd);
        unpackData();
    }

    @Override
    public void unpackData() {
        ByteBuffer bf = makeBuffer();
        try {
            direction = readShort(bf);
        } catch (Exception e) {
            direction = DemoDirection.UP.getValue();
            CommonHandle.writeErrLog(e);
        }
    }

    public short getDirection() { return direction; }
}
```

2.4.2 Xử lý gói tin trả về

Sau khi xử lý, server gửi gói tin về client. Gói tin trả về extends BaseMsg và override hàm createData()

```
public class ResponseMove extends BaseMsg {
    private Point pos;
    public ResponseMove(short error, Point pos) {
        super(CmdDefine.MOVE, error);
        this.pos = pos;
    }

    @Override
    public byte[] createData() {
        ByteBuffer bf = makeBuffer();
        bf.putInt(pos.x);
        bf.putInt(pos.y);
        return packBuffer(bf);
    }

    public Point getPos() { return pos; }
}
```

2.5 Làm việc với Database

Dữ liệu của user sẽ được lưu trong Couchbase – một noSQL dạng document

Chú ý: khi chạy server trên máy local, chú ý bật memcached.exe trước để giả lập Couchbase

Lưu thông tin xuống db

Trong class DataHandler có cung cấp một số hàm để ghi và lấy dữ liệu từ database

```
public static void set(String key, Object obj) throws Exception {
    DataController.getController().set(PREFIX + key, obj);
}

public static Object get(String key) throws Exception {
    return DataController.getController().get(PREFIX + key);
}
```

Trong đó có hai hàm chính

- `DataController.getController().set(String key, Object obj)`
  Hàm save xuống database
  - Key: tên của key trong database
  - Obj: Object cần lưu

- `DataController.getController().get(String key)`
  Hàm lấy dữ liệu từ databases, trả về Object đã được lưu

Lưu ý Object obj nên ở dạng json, có thể sử dụng lib gson để encode/decode object thành dạng json. Xem demo trong class DataModel

Demo get/set from database.

Load userinfo trong src/event/handler/LoginSuccessHandler -> onLoginSuccess()

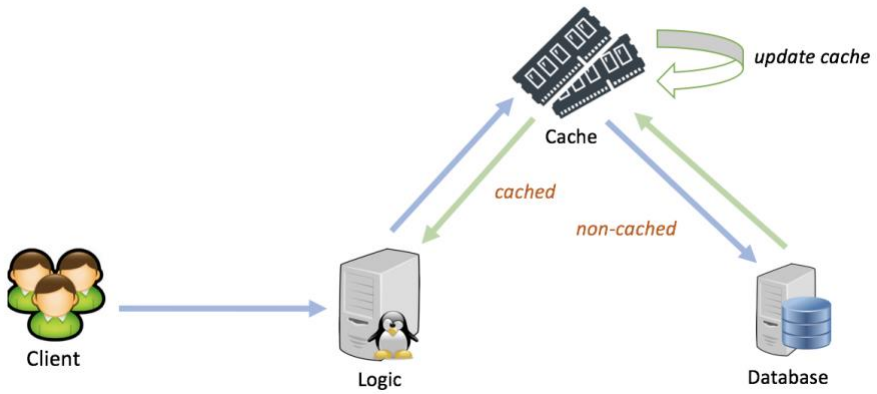
```
private void onLoginSuccess(User user) {
    trace(ExtensionLogLevel.DEBUG,  ...objects: "On Login Success ", user.getName());
    PlayerInfo pInfo = null;
    try {
        pInfo = (PlayerInfo) PlayerInfo.getModel(user.getId(), PlayerInfo.class);
    } catch (Exception e) {
        e.printStackTrace();
    }

    if (pInfo==null){
        pInfo = new PlayerInfo(user.getId(),  _name: "username_" + user.getId());
        try {
            pInfo.saveModel(user.getId());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

get from database

set to database

Cache Database



Cache database trên RAM

Khi user login thành công, server bắn ra sự kiện USER_LOGIN

Trong LoginSuccessHandler sẽ thực hiện một số tác vụ, trong đó có cache lại thông tin của user từ database lên trên RAM

```
/**
 * cache playerinfo in RAM
 */
user.setProperty(ServerConstant.PLAYER_INFO, pInfo);
```

từ các action sau, khi thao tác với thông tin user thì lấy trực tiếp trên RAM để thao tác

Ví dụ

```
private void processMove(User user, RequestMove move){
    try {
        PlayerInfo userInfo = (PlayerInfo) user.getProperty(ServerConstant.PLAYER_INFO);
        if (userInfo==null){
            send(new ResponseMove(DemoError.PLAYERINFO_NULL.getValue(), new Point()), user);
            return;
        }

        userInfo.move(move.getDirection());
        userInfo.saveModel(user.getId());

        send(new ResponseMove(DemoError.SUCCESS.getValue(), userInfo.position), user);

    } catch (Exception e) {
        send(new ResponseMove(DemoError.EXCEPTION.getValue(), new Point( x: 0, y: 0)), user);
    }
}
```

2.6 Config hệ thống

Các config chính cần quan tâm

- Conf/cluster.properties

```
# Database
db_prefix = fresher_group_x
dservers=127.0.0.1:11211
cservers=127.0.0.1:11211

# Config Extension
games=fresher
extensions=main
extension.main.path=extension.FresherExtension

# 1: social, 2: set direct, 3: autoincrement
custom_login=3
```

prefix cho key lưu xuống database

config port cho database, 11211 là port mặc định khi memcached.exe chạy

đường dẫn đến extension chính của game

- Config/server.xml

```
<serverSettings>
  <socketAddresses>
    <socket address="0.0.0.0" port="1101" type="TCP"/>
```

port mà server sẽ lắng nghe, client connect thông qua port này

3. FAQ – một số câu hỏi thường gặp

Hàm main nằm ở đâu?

Hàm main được đặt trong

bz-allinone.jar!\bitzero\server\Main.class

file này nằm trong lib nên các bạn không chỉnh sửa được. Trong hàm main() này sẽ khởi tạo framework, lắng nghe server trên port config trong config/server.xml và khởi tạo các handler tương ứng

Làm thế nào để gửi tin nhắn giữa 2 class?

Có hai cách để một module nhận gói tin:

- Từ client gửi đến. Module đó phải đăng ký nhận gói tin từ client gửi đến bằng cách extend class BaseServerEventHandler. Xem thêm phần Xử lý gói tin đến
- Sử dụng hệ thống Sự kiện để gửi tin. Một module sẽ bắn ra sự kiện và module kia sẽ đăng ký sự kiện nhận. Xem thêm phần Hệ thống xử lý Sự kiện

Làm thế nào để getUserByID từ uId?

Mỗi user khi đăng nhập hệ thống thành công sẽ được định nghĩa bởi hai tham số unique: id và username. Trong UserManager của BitZero có một số hàm:

```
User user = BitZeroServer.getInstance().getUserManager().get
```

```
getUserById(int i)                          User
getUserByName(String s)                     User
getUserBySession(ISession iSession)         User
getUserCount()                              int
getAllSessions()                            List<ISession>
getAllUsers()                               List<User>
getHighestCCU()                             int
```

Trong đó:

- getUserById(): lấy thông tin của user theo id
- getUserByName() : lấy thông tin user theo username

- getUserCount() : tổng số user đang online
- getAllUsers() : lấy danh sách toàn bộ User
- getHighestCCU() : tổng số user cùng online cao nhất kể từ khi server được khởi động

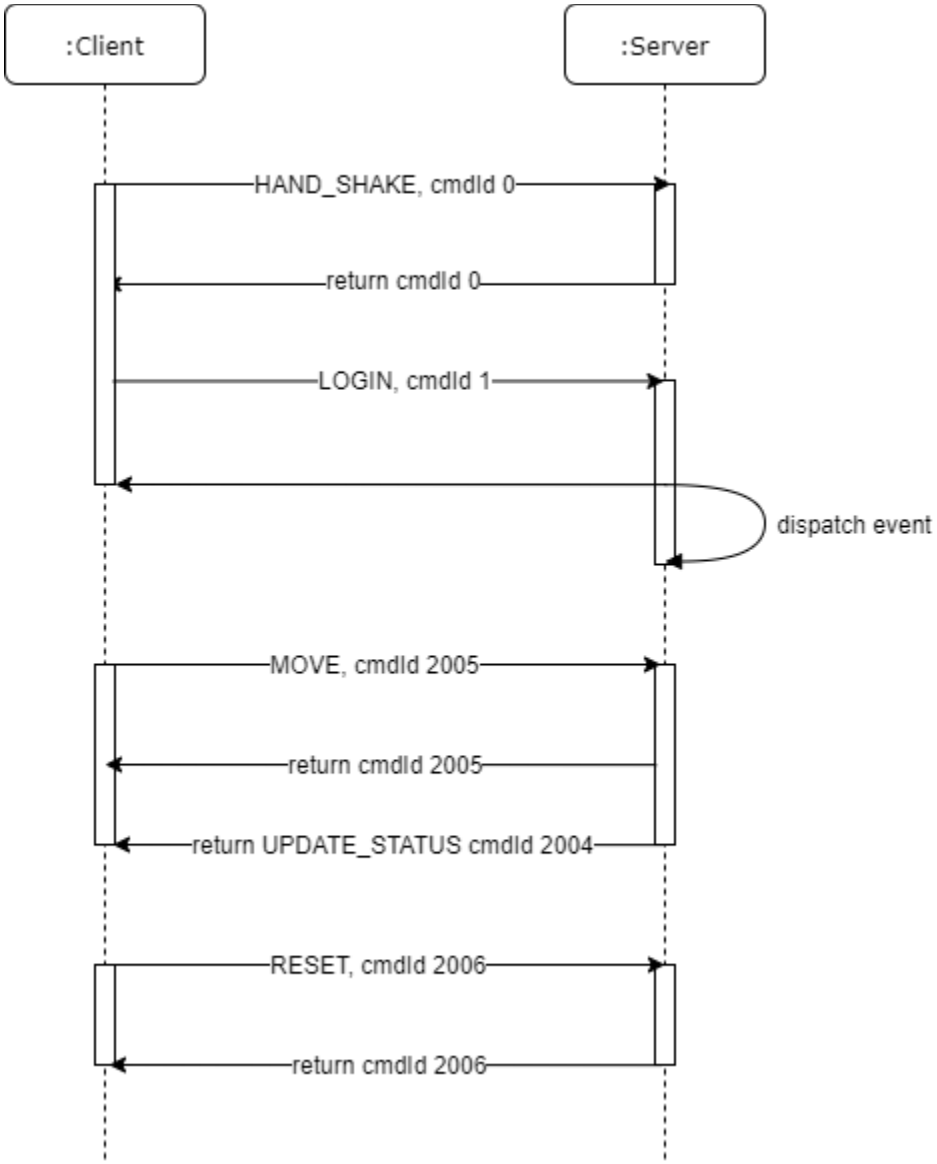
Hai client cùng login một tài khoản sẽ thế nào?

User login sau sẽ đá user login trước ra khỏi hệ thống, và engine sẽ bắn ra 2 sự kiện

- USER_DISCONNECT : đối với session đang login
- USER_LOGIN : đối với session login sau

4. Workshop

4.1 Flow demo



4.2 API

Tất cả các gói tin trả về sẽ có sẵn param Error. Bảng giá trị Error

Error	Description
0	Thành công
1	Lỗi chưa xác định
2	Thông tin user rỗng
3	Exception

4	Param không hợp lệ
5	Ô nãy đã được đi qua

a. LOGIN, cmdId = 1

Request		
Param	Type	Description
sessionKey	String	sessionKey để đăng nhập, với Fresher thì chưa cần quan tâm tham số này
userId	Int	UserId để đăng nhập

Response		
Param	Type	Description
Error	byte	

b. MOVE, cmdId = 2005

Request		
Param	Type	Description
x	int	x, y: vị trí trong map cần di chuyển đến
y	int	

Response		
Param	Type	Description
Error	byte	
x	int	
y	int	

c. UPDATE_STATUS

Response		
Param	Type	Description
Error	byte	
x	int	Vị trí hiện thời của nhân vật
y	int	
x_len	int	Kích thước mảng visited
y_len	int	
visited	boolean [][]	x_len * y_len giá trị boolean thể hiện vị trí đã qua hay chưa

d. RESET

Request		
Param	Type	Description

Response		
Param	Type	Description
Error	byte	

4.3 Practice

Các công việc cần làm

- Tìm hàm login tại client, thay đổi userId truyền lên server
- Thay đổi custom_login trong cluster.properties để server đăng nhập bằng userId
- Client put param vào gói MOVE để gửi vị trí lên server, server xử lý gói tin đến
- Server lưu thay đổi vào database
- Gửi data về client
- Bắn và xử lý sự kiện