# Message design by Actor

AKKA- .

:

.

.

UML TOOL(STAR UML) .

AKKA , .

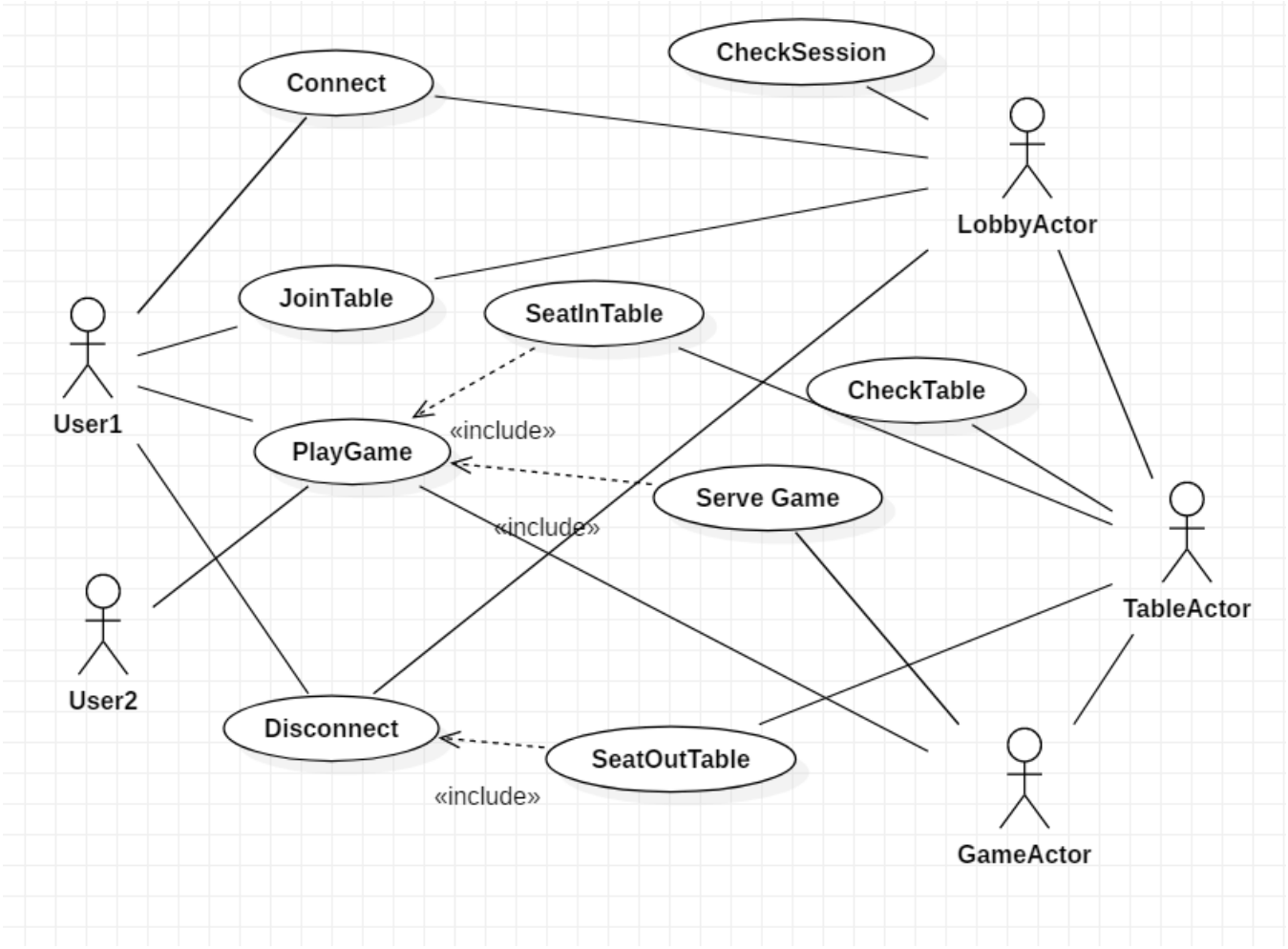
, .

.

## vs

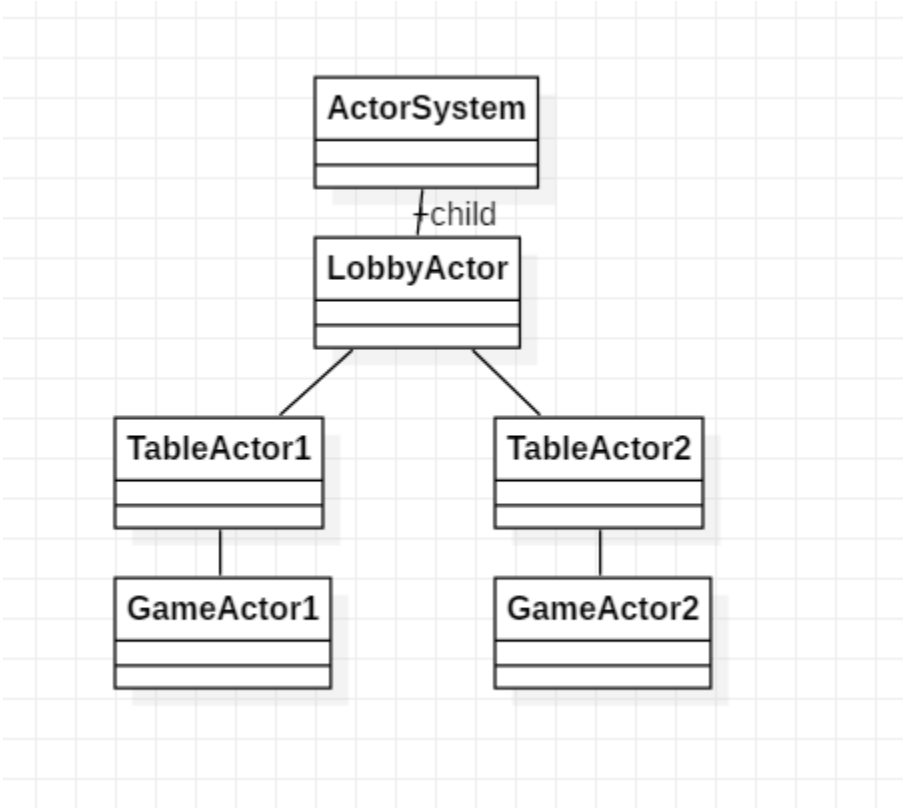
- : https://github.com/psmon/gameweb/tree/master/src/main/java/com/vgw/demo/gameweb/thread
- : https://github.com/psmon/gameweb/tree/master/src/main/java/com/vgw/demo/gameweb/actor
- Code : https://github.com/psmon/gameweb

## UseCase




Actor . UseCase .

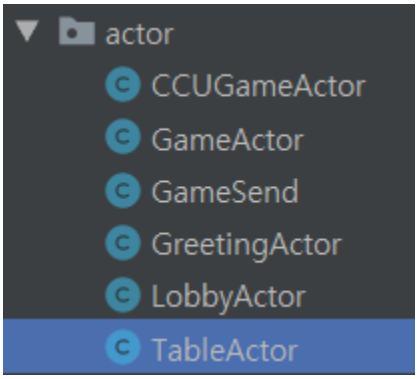
, AKKA .




UserCase ,  .

AKKA System , ,  . - https://doc.akka.io/docs/akka/2.5/cluster-usage.html#a-simple-cluster-example

, . - AKKA

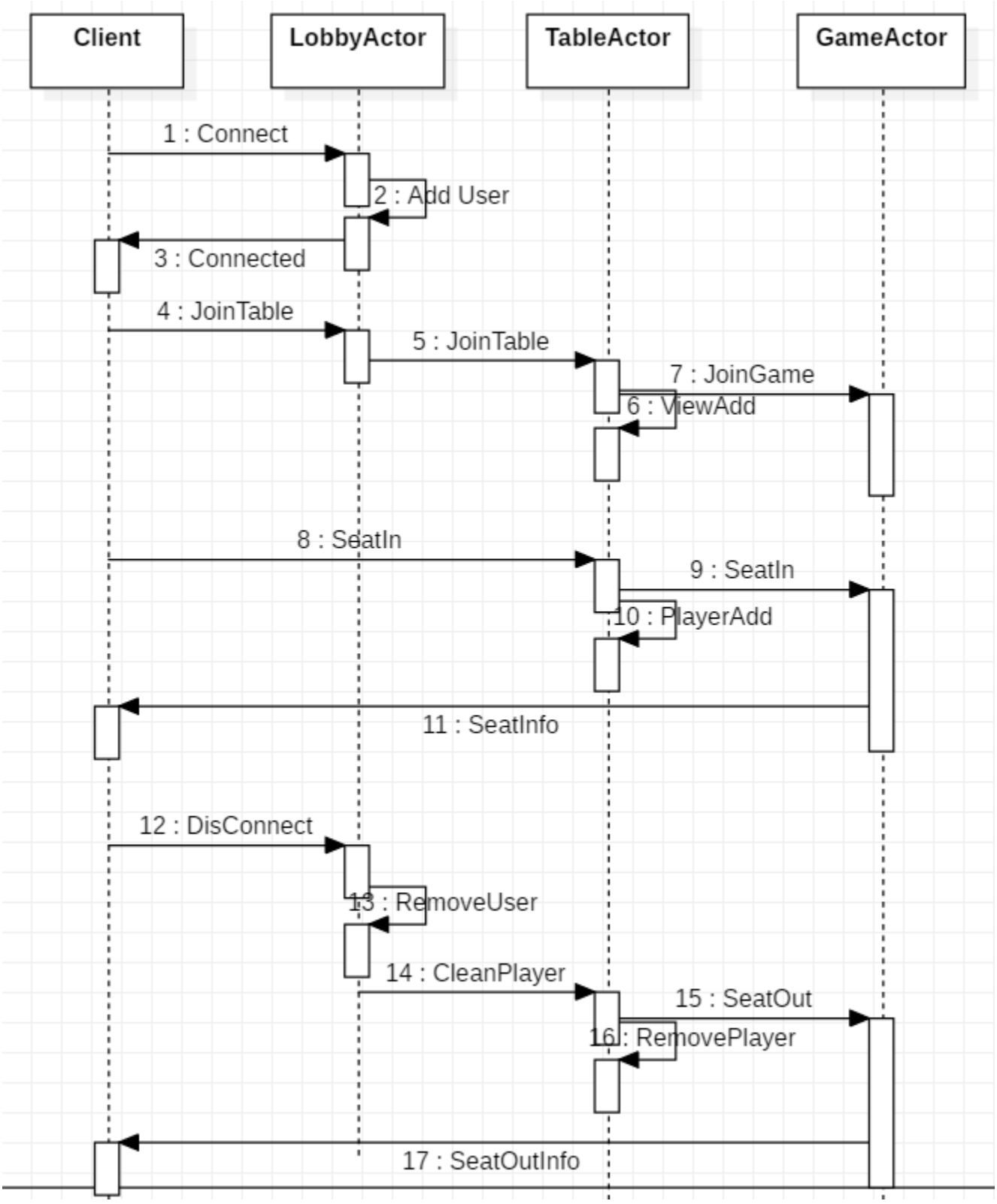



.

. ( = )

.

.


Client
LobbyActor
TableActor
GameActor
1 : Connect
2 : Add User
3 : Connected
4 : JoinTable
5 : JoinTable
7 : JoinGame
6 : ViewAdd
8 : SeatIn
9 : SeatIn
10 : PlayerAdd
11 : SeatInfo
12 : DisConnect
13 : RemoveUser
14 : CleanPlayer
15 : SeatOut
16 : RemovePlayer
17 : SeatOutInfo

,

AKKA ()

(WS) .

.

switch .

.

```
@Override
public AbstractActor.Receive createReceive() {
    return receiveBuilder()
            .match(ConnectInfo.class, c -> {
                if(c.getCmd()== ConnectInfo.Cmd.CONNECT){
                    sessionMgr.put(c.getSessionId(),c.getWsSender());
                    log.info("user connected:"+c.getSessionId());
                }else if(c.getCmd()== ConnectInfo.Cmd.DISCONET){
                    sessionMgr.remove(c.getSessionId());
                    Player removeUser = new Player();
                    removeUser.setSession(c.getSessionId());

                    if(c.getTableNo()>0){
                        findTableByID(c.getTableNo()).tell(new SeatOut(removeUser),ActorRef.noSender());
                    }else{
                        findTableALL().tell(new SeatOut(removeUser),ActorRef.noSender());
                    }
                    log.info("user disconnected:"+c.getSessionId());
                }
                sessionMgr.put(c.getSessionId(),c.getWsSender());
            })
            .match(TableCreate.class, t->{
                // Create a table under the lobby, if you have an Actor named TableManagement, you can move
easily.
                String tableUID = "table-" + t.getTableId();
                if(t.getCmd() == TableCreate.Cmd.CREATE){
                    ActorRef tableActor = getContext().actorOf( TableActor.props(t,this.getSelf() ), tableUID);
                    tableActor.tell(t,ActorRef.noSender());
                }
            })
            .match(JoinGame.class, j->{
                joinGameTable(j.getTableId(),j.getName(),j.getSession());
            })
            .match(MessageWS.class, m->{
                send(m.getSession(),m.getGameMessage());
            })
            .build();
}
```

## ActorPath

, . ( .)

.( .)

, .

### OOP VS ACTOR

| OOP | ACTOR |
|---|---|
| Lobby a;<br>a.getTable(1).getTableName(); | LobbyActor a;<br>TableActor b;<br>b.tell("some ask",a) |

OOP . - ? .

, OOP OOP

.( .)

### Ask

```
private ActorRef findTableByID(int tableID) throws Exception {
    String tableActorPath = "/user/lobby/table-"+tableID;
    ActorSelection tableSelect = this.getContext().actorSelection(tableActorPath);
    FiniteDuration duration = FiniteDuration.create(1, TimeUnit.SECONDS);
    Future<ActorRef> fut = tableSelect.resolveOne(duration);
    ActorRef tableActor = Await.result(fut, duration);
    return tableActor;
}
```

, .

```
ActorRef someActor = system.ActorOf(........);
someActor.tell("some message",null);
```

, .

tell .

, .

.

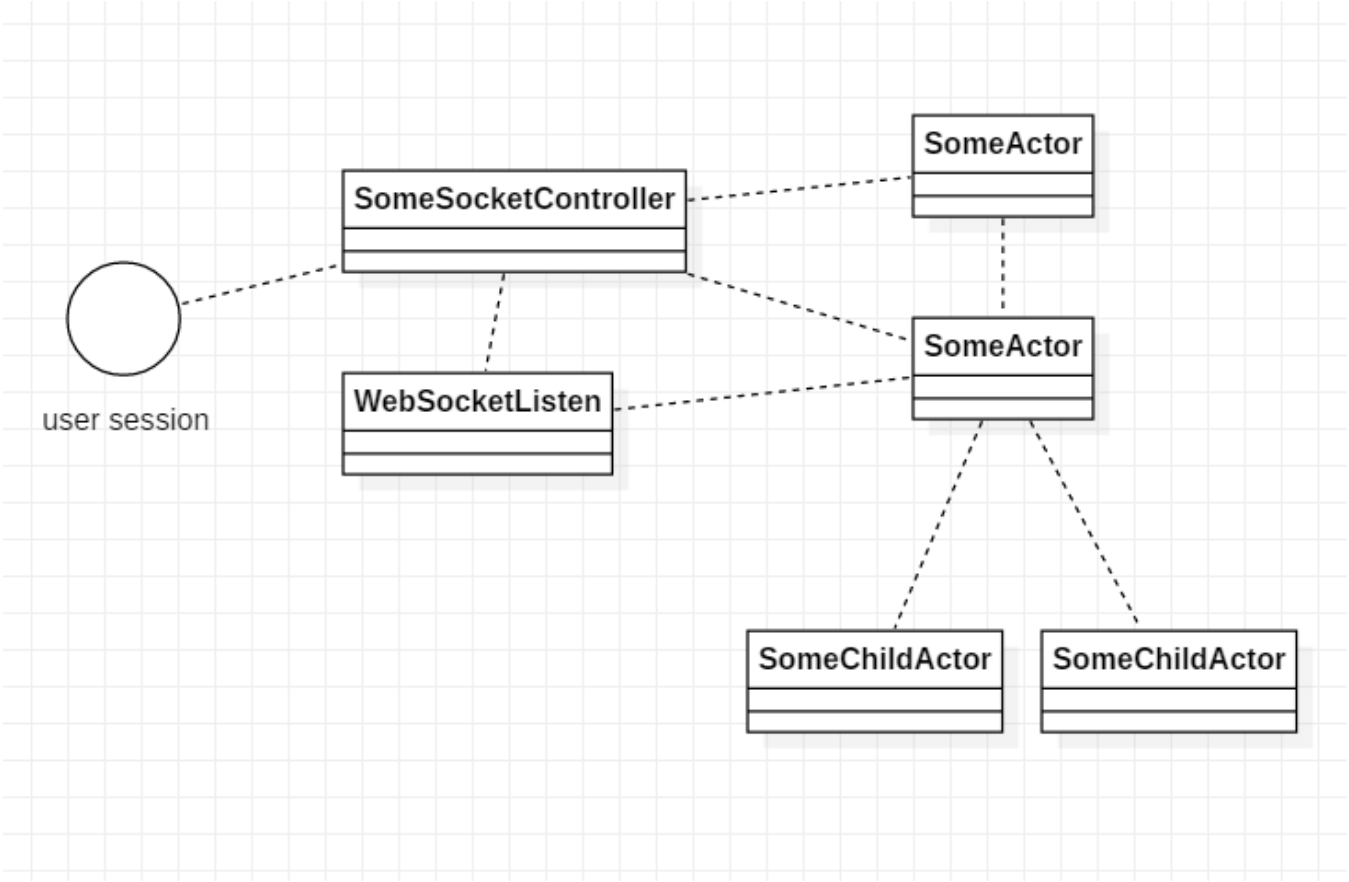
Lock/Unlock .

```
ActorSelection lobbyActor = system.ActorSelection("user/lobby");
lobbyActor.tell("some message",null);


//     .
ActorSelection tableAllActor = system.ActorSelection("user/lobby/table/*");
tableAllActor.tell("some message",null);
```

,

.




.

, .

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-websocket</artifactId>
</dependency>


<dependency>
   <groupId>org.webjars</groupId>
   <artifactId>webjars-locator-core</artifactId>
</dependency>
<dependency>
   <groupId>org.webjars</groupId>
   <artifactId>sockjs-client</artifactId>
   <version>1.0.2</version>
</dependency>

<dependency>
   <groupId>org.webjars</groupId>
   <artifactId>stomp-websocket</artifactId>
   <version>2.3.3</version>
</dependency>
```

Stomp Spring .

: https://spring.io/guides/gs/messaging-stomp-websocket/

```
public class WebSocketEventListener {
    @Autowired
    Lobby lobby;

    @Autowired
    private ActorSystem system;

    private static final Logger logger = LoggerFactory.getLogger(WebSocketEventListener.class);

    @Autowired
    private SimpMessageSendingOperations messagingTemplate;

    @EventListener
    public void handleWebSocketConnectListener(SessionConnectedEvent event) {
        logger.info("Received a new web socket connection");
        StompHeaderAccessor headerAccessor = StompHeaderAccessor.wrap(event.getMessage());
        String sessionId = headerAccessor.getUser().getName();
    }

    @EventListener
    void handleSessionConnectedEvent(SessionConnectedEvent event) {
        // Get Accessor
        StompHeaderAccessor sha = StompHeaderAccessor.wrap(event.getMessage());
        String sessionId = sha.getUser().getName();

        // OOP VS MESSAGE

        // #OOP - It's simple to develop locally, but many things need to change in order to be extended
remotely.
        lobby.addSender(sessionId,messagingTemplate);

        // #ACTOR - This can be extended to remote without implementation changes.
        ActorSelection lobby = system.actorSelection("/user/lobby");
        lobby.tell(new ConnectInfo(sessionId,messagingTemplate, ConnectInfo.Cmd.CONNECT), ActorRef.noSender());

        // Suppose you create several child actors under table.
        // You can send a message to all children with the following commands.
        // This is very useful.
        // Sameple Cmd : ActorSelection lobby = system.actorSelection("/user/table/*");
    }

    @EventListener
    public void handleWebSocketDisconnectListener(SessionDisconnectEvent event) {
        StompHeaderAccessor headerAccessor = StompHeaderAccessor.wrap(event.getMessage());
        String username = (String) headerAccessor.getSessionAttributes().get("username");
        String session = headerAccessor.getUser().getName();

        if(username != null) {
            logger.info("User Disconnected : " + username);
            GameMessage gameMessage = new GameMessage();
            gameMessage.setType(GameMessage.MessageType.LEAVE);
            gameMessage.setSender(username);
            //messagingTemplate.convertAndSend("/topic/public", gameMessage);

            // #OOP
            lobby.removeSender(session);

            // #ACTOR
            ActorSelection lobby = system.actorSelection("/user/lobby");
            lobby.tell(new ConnectInfo(session,messagingTemplate, ConnectInfo.Cmd.DISCONET), ActorRef.
noSender());
        }
    }

}
```

.

, .

:

- https://github.com/psmon/gameweb/blob/master/src/main/java/com/vgw/demo/gameweb/config/WebSocketConfig.java
- https://github.com/psmon/gameweb/blob/master/src/main/java/com/vgw/demo/gameweb/controler/ws/WebSocketEventListener.java

```
@Controller
@SuppressWarnings("Duplicates")
public class GameController {

    private static final Logger logger = LoggerFactory.getLogger(GameController.class);

    @MessageMapping("/game.req")
    @SendTo("/topic/public")
    public GameMessage gameReq(@Payload GameMessage gameMessage,
                               SimpMessageHeaderAccessor headerAccessor) {

        String  sessionId = headerAccessor.getUser().getName();
        logger.info("GameMsg:" + gameMessage );
        String gamePacket = gameMessage.getContent();
        String splitMessage[] = gamePacket.split("!!");

        String userName = headerAccessor.getSessionAttributes().get("username").toString();
        String userSession =  headerAccessor.getUser().getName();

        Object objTableNo = headerAccessor.getSessionAttributes().get("tableNo");
        Integer tableNo = objTableNo!=null? (Integer)objTableNo : -1;
....
}
```

.

, .

, .

:

- https://github.com/psmon/gameweb/tree/master/src/main/java/com/vgw/demo/gameweb/message
- https://github.com/psmon/gameweb/blob/master/src/main/java/com/vgw/demo/gameweb/controler/ws/GameController.java

## JS

```
var stompClient = null;

function setConnected(connected) {
    $("#connect").prop("disabled", connected);
    $("#disconnect").prop("disabled", !connected);
    if (connected) {
        $("#conversation").show();
    }
    else {
        $("#conversation").hide();
        sceneControler('intro');
    }
    $("#greetings").html("");
}
```

```
function connect() {
    var socket = new SockJS('/ws');
    stompClient = Stomp.over(socket);
    stompClient.connect({}, function (frame) {
        setConnected(true);
        console.log('Connected: ' + frame);

        // for broad cast
        stompClient.subscribe('/topic/public', onMessageReceived );

        // for send to some
        stompClient.subscribe('/user/topic/public', onMessageReceived );

        var username=$("#name").val();
        if(username.length<1){username="Unknown"};
        // Tell your username to the server
        stompClient.send("/app/lobby.addUser",
            {},
            JSON.stringify({sender: username, type: 'JOIN'})
        )
    });
}

function disconnect() {
    if (stompClient !== null) {
        stompClient.disconnect();
    }
    setConnected(false);
    console.log("Disconnected");
}

function joinTable(tableNo) {
    stompClient.send("/app/game.req",
        {},
        JSON.stringify({content: 'join',num1:tableNo, type: 'GAME'})
    )
}

function seatTable() {
    stompClient.send("/app/game.req",
        {},
        JSON.stringify({content: 'seat', type: 'GAME'})
    )
}


function sendChatMsg() {
    var content = $('#gamemsg').val();
    stompClient.send("/app/hello",
        {},
        JSON.stringify({content: content, type: 'CHAT'})
    )
}

function sendGameMsg() {
    var content = $('#gamemsg').val();
    stompClient.send("/app/game.req",
        {},
        JSON.stringify({content: content, type: 'GAME'})
    )
}

function sendGameAction(action) {
    var content = $('#gamemsg').val();
    stompClient.send("/app/game.req",
        {},
        JSON.stringify({content: action.content,num1:action.num1,num2:action.num2, type: 'ACTION'})
    )
}

function showGreeting(message) {
```

```
    $("#greetings").append("<tr><td>" + message + "</td></tr>");
}

function onMessageReceived(payload) {
    var message = JSON.parse(payload.body);

    var messageElement = document.createElement('li');

    if(message.type == 'JOIN') {
        showGreeting('Welcome ' + message.sender)
    } else if (message.type == 'LEAVE') {
        showGreeting(message.sender + 'left!')
    } else if(message.type == 'GAME'){
        messageControler(message);
        //showGreeting(message.content);
    } else{
        showGreeting(message.content);
    }
}

$(function () {
    $("form").on('submit', function (e) {
        e.preventDefault();
    });
    $( "#connect" ).click(function() { connect(); });
    $( "#disconnect" ).click(function() { disconnect(); });
    $( "#send" ).click(function() { sendGameMsg(); });


});
```

.js .

4 .

- connect :
- disconnect :
- onMessage :
- send :

:

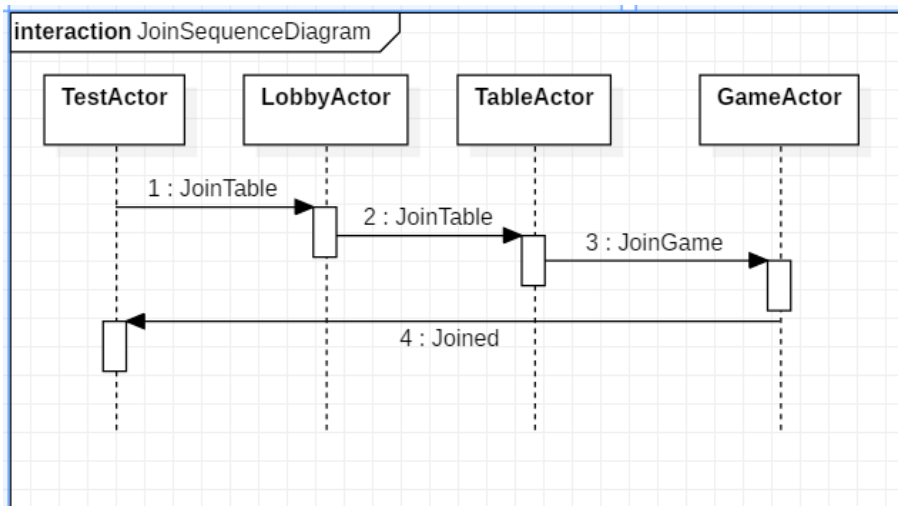
- https://github.com/psmon/gameweb/blob/master/src/main/resources/static/app.js

, .

, .

Akka TestKit , .

## TestCase


interaction JoinSequenceDiagram
TestActor
LobbyActor
TableActor
GameActor
1 : JoinTable
2 : JoinTable
3 : JoinGame
4 : Joined

TestCode

```
import akka.actor.ActorRef;
import akka.actor.ActorSystem;
import akka.testkit.javadsl.TestKit;
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringRunner;
import java.time.Duration;

@RunWith(SpringRunner.class)
@SpringBootTest
@ContextConfiguration(classes = AppConfiguration.class)
@SuppressWarnings("Duplicates")
public class SessionTest {

    static ActorSystem system;

    @BeforeClass
    public static void setup() {
        system = ActorSystem.create();
    }

    @AfterClass
    public static void teardown() {
        TestKit.shutdownActorSystem(system);
        system = null;
    }

    @Test
    public void testIt() {
        new TestKit(system) {{
            final ActorRef lobbyActor = system.actorOf(LobbyActor.props(),"lobby");
            final String testSessionID = "jaskfjkjaslfalsf";

            // Create TableActor
            for(int i=0;i<10;i++){
                lobbyActor.tell( new TableCreate(i,"table-"+i , TableCreate.Cmd.CREATE),getRef() );
                expectMsg(Duration.ofSeconds(1), "created");
            }

            // Try Connect
            lobbyActor.tell(new ConnectInfo(testSessionID, null,ConnectInfo.Cmd.CONNECT),getRef());
            expectMsg(Duration.ofSeconds(1), "done");

            // Find User
            lobbyActor.tell(new ConnectInfo(testSessionID, null,ConnectInfo.Cmd.FIND),getRef());
            expectMsg(Duration.ofSeconds(1), "User exists");

            // Join Table : Forward Check , lobby->table->game->getRef()
            lobbyActor.tell(new JoinGame(1,"test",testSessionID),getRef() );
            expectMsg(Duration.ofSeconds(1), "joined");

            // Try Disconnect
            lobbyActor.tell(new ConnectInfo(testSessionID, null,ConnectInfo.Cmd.DISCONET),getRef());
            expectMsg(Duration.ofSeconds(3), "done");

            // Find Again
            lobbyActor.tell(new ConnectInfo(testSessionID, null,ConnectInfo.Cmd.FIND),getRef());
            expectMsg(Duration.ofSeconds(1), "User does not exist");
        }};
    }
}
```


- https://getakka.net/articles/intro/what-are-actors.html
- https://doc.akka.io/docs/akka/2.5/general/addressing.html - Actor Path
- https://www.baeldung.com/akka-with-spring