ChatApp API Specification Doc

1. Design Decisions
2. Register

* User must register before initiating the chatroom application
* If the connection is lost or the page is refreshed, user needs to register again
* The profile information will be used to filter whether a user is qualified to join a chatroom

1. Create chatroom

* User will create a chatroom of his own when registered
* A new user will cause the update of all the active user on the chat App

1. Join in new chatroom

* User can join in any chatroom showed in the “join” tab
* The chatrooms showed are already qualified for the user’s profile.

1. Leave chatroom

* User chooses to exit a chatroom by clicking the “Leave” or “Leave all” button
* User is forced to leave all the chatrooms if user uses the word “hate” in the message. (The user must register again)

1. Receive notification

* User will be notified when a new user join in or leave the chatroom and the reason.

1. Create new chatarea

* The user clicks on the “chat” button. This will cause a request sent from front-end to back-end, which aims to check whether a chat history exists between the users.

1. Send message

* Sending messages to one of the members in the chatroom, there will be a “received”
* Only the owner of the chatroom can send messages to all the other users in the chatroom, the message will appear on the private chat screen.

1. View-Model Communication
2. backend -> frontend
3. AckResponse

{

"message":{

"id":1,

"roomId":-1,

"senderId":3,

"receiverId":4,

"message":"hello",

"isReceived":true

},

"type":"AckResponse"

}

1. NewMessageResponse

{

"message":{

"id":1,

"roomId":-1,

"senderId":3,

"receiverId":4,

"message":"hello",

"isReceived":false

},

"type":"NewMessageResponse"

}

1. RoomNotificationResponse

//send to all the users in the chatroom

{

"roomId":1,

"notifications":["User2 joins this room"],

"type":"RoomNotificationResponse"

}

1. RoomUsersResponse

// send to all the users in the chatroom

// Observable Chatroom setChanged and notify all the users to update the uselist

{

"roomId":1,

"users":[1,2],

"type":"RoomUsersResponse"

}

1. UserChatHistoryResponse

{

"chatHistory":[

{

"id":1,

"roomId":-1,

"senderId":3,

"receiverId":4,

"message":"hello",

"isReceived":true

}

],

"type":"UserChatHistoryResponse"

}

1. UserRoomResponse

// user’s joinedRoom list

{

"userId":2,

"joinedRoomIds":[2],

"availableRoomIds":[1,3,4], // already filtered

"type":"UserRoomResponse"

}

1. frontend -> backend
2. signup

{

"commandType":"signup",

"age":18,

"location":"North America",

"school":"Rice", // user profile

"ageResL":15, //minimum age

"ageResU":20, //maximum age

"locRes":["North America"], // location restriction

"schoolRes":["Rice"] // school restriction

}

1. joinroom

{

"commandType":"joinRoom",

"chatroom":1 // the id of new joined room

}

1. message

{

"commandType":"message",

"name": [-1 | userId], //-1 stands for send all

"content":"hello"

}

1. ack

{

"commandType":"ack",

"msgId":1

}

1. chatHistory

{

"commandType":"chatHistory",

"name":3

}

1. leaveroom

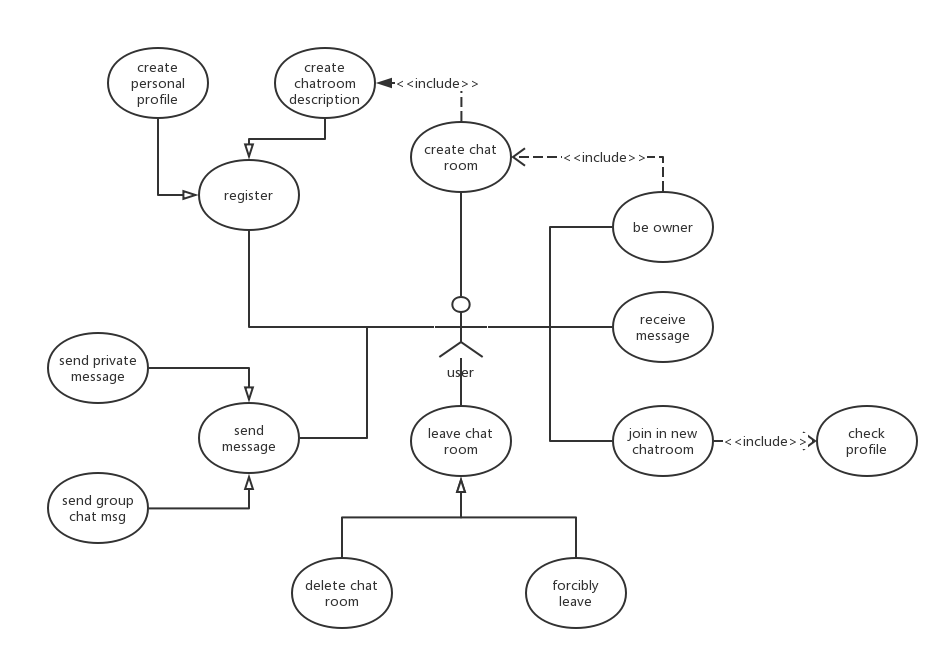
{

"commandType":"leaveRoom",

"chatroom": [-1 | roomId] // -1 means leave all the chatrooms

}

1. Use Case Diagram



1. APIs
2. Package controller

* chatAppController

the chat app controller communicates with all the clients on the web socket

* WebSocketController

webSocketController creates a web socket for the server

1. Package cmd

**Interface IUserCmd**

The interface IUserCmd provides an interface whenever Observable like Dispatcher of ChatRoom want to notify observers (users). It has the following concrete class:

* AddRoomCmd

used to inform all users that a new chatroom is created

* CollectNameCmd

command to generate the map of userId to username

* JoinRoomCmd

command used to join the chatroom

* LeaveRoomCmd

command used to leave the chatroom

* NotifyCmd

command containing response that should be delivered to the client

* RemoveRoomCmd

command executed when the owner leaves its room, notify all the users in the room

* UpdateRoomListCmd

command used to update the roomlist

|  |  |  |
| --- | --- | --- |
| Methods | | |
| Modifier and Type | Method | Description |
| public void | execute(User context) | The command excuted on the receiver |

Method Detail

* execute

public void execute(User context)

execute is the function that all command will execute once the command is passed to observer’s update

Parameters:

user - the object that will execute the receiving comand

1. Package obj

* ChatRoom

the ChatRoom class extends Observable, defines a chat room object and private fields

* Message

the Message class defines a message object and its private fields

* User

the User class defines a user object and its private fields

1. Package res

**Abstract Class AResponse**

Abstract class AResponse wraps the message sending from server to client. It has the following subclasses:

* AckResponse

wrapping the confirmation message indicates whether the message has been received

* CommandResponse

formats the command string achieved from the websocket

* NewMessageResponse

defines the format of new chat message

* RoomNotificationResponse

defines the format of the chatroom notification, i.e. change of the members in the chatroom and corresponding reason.

* RoomUsersResponse

wraps the message of the chatroom userlist

* UserChatHistoryResponse

wraps the information of all history messages in the chatroom (between two users)

* UserRoomResponse

wraps the information of all the chatrooms (joined and available) of one user.

|  |  |  |
| --- | --- | --- |
| Fields | | |
| Modifier and Type | Field | Description |
| private String | type | The type of the response, using class name to denote tyoe |

|  |  |
| --- | --- |
| Constructors | |
| Constructor | Description |
| AResponse | default constructor |
| AResponse(String type) | Construct a response with specific type |

|  |  |  |
| --- | --- | --- |
| Methods | | |
| Modifier and Type | Method | Description |
| public String | toJson() | Convert the object to json string |

Method Detail

* toJson

public Integer toJson()

Convert the object to json string

Returns:

The json encoding of object itself

1. DispatcherAdapter

the DispatcherAdapter class helps handle web socket messages