Instant Messaging service with database

Juan Luis Gorricho

Third approach

- This is a second remote approach implementing an instant messaging service.
- This approach focuses on using a web application service, including **persistence at the server**; and accessing the service using a REST API.
- The approach uses Swing clients instead of Internet browsers, and the **asynchronous behavior** is implemented with a pushing strategy using a WebSockets library. The client has no persistence.

Third approach

- On considering persistence there is a substantial change on the implementation approach.
- The exercise must be solved focusing almost **exclusively** on handling tables of a ddbb. This is consistent with the common use of RESTful web services, only managing resources in a stateless fashion, not looking for an implement of an RPC tool.
- Consequently, for this approach the client implementation turns into a sequence of send/receive of records (rows) of those tables.

Third approach

- Now the close functionality at SwingClient **will not** unsubscribe the client from all her/his subscriptions and **will not** remove her/his publishing profile, if that was the case.
- Consequently, after logging into the system, the SwingClient must retrieve the user profile: the topic she/he was publisher of, if that's the case; her/his subscriptions, and all the messages from those subscriptions as well.
- **Note**: the login is done without user input to preserve the UI from previous exercises.

Third approach

- Finally, the original mechanism for broadcasting the messages from a publisher to all its subscribers, as developed on our first two approaches, here is moved entirely to the WebSocketServer java class.
- Pay attention to the methods notifyNewMessage() and notifyTopicClose() of WebSocketServer.
- Nevertheless, from the client's perspective this is something unnoticeable.

Exercise

- Import the instantmessaging.sql file.
- Open the web application project: InstantMessagingRemote_server_with_ddbb. Add the instantmessaging connection to the data base if necessary. Import the GSON library if necessary.
- Run the web application project.
- Try to understand the coding of the REST facades and the WebSocketServer.

Exercise

- Open the InstantMessagingRemote_client_with_ddbb_exercise.
- Try to understand the coding of the Java classes included in apiREST.
- Complete the code of the SwingClient as in the previous version except for:
  - The additional programming of the method: clientSetup() where you have to restore the user profile,
  - Avoid removing the user as publisher or subscriber of any topic on closing the application.
- Complete the code of PublisherStub and TopicManagerStub in a similar way as in the previous version, although some additional methods are necessary to restore the user profile.

Exercise

- Complete the code of the WebSocketClient in a similar way as you did for the previous version.
- Review the schema of classes of the following slide for a better understanding of the exercise.
- Run separately two or three clients with different login/password values to test the code.

Schema of classes for a remote service