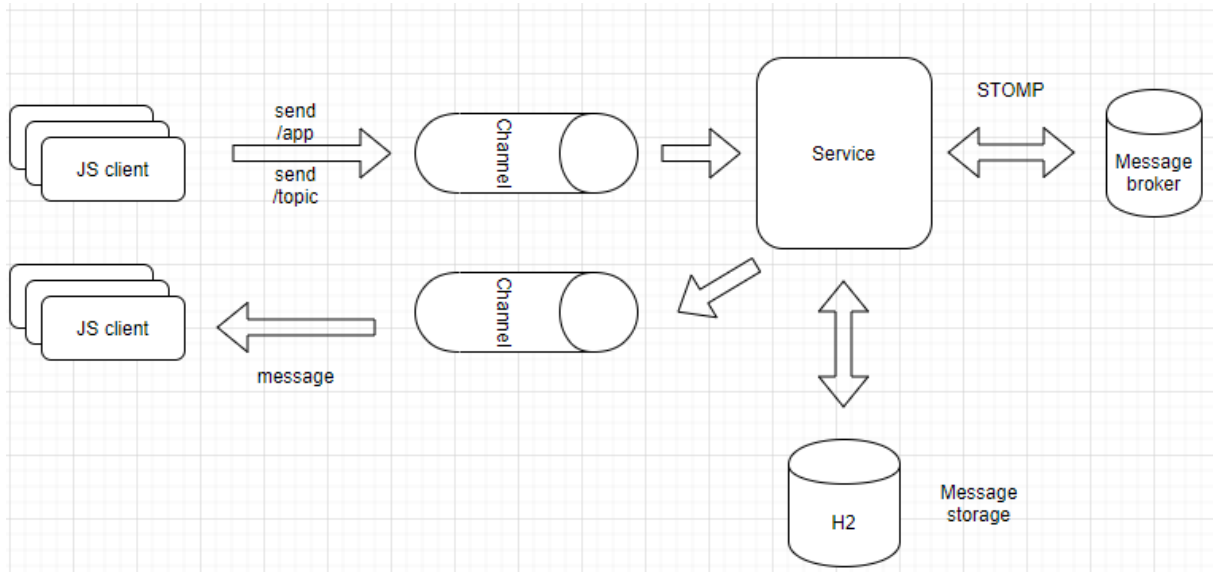


Action Monitor

Summary: The application can receive messages via websocket connection and client can subscribe to a particular topic (topic/public). When one user sends out a message it is broadcasted to all of the clients on this topic.

Achitecture:



Components:

- 1.) Service: Java application using spring boot
- 2.) Database: H2 in-memory database in order to store the messages
- 3.) JSClient: javascript client connecting to the service via websocket and STOMP protocol

User guide

- 1.) Start the server: „mvnw spring-boot:run”

[illegible]

- 2.) Open the client and type in the user name than click on „Connect to the server“:

- a. <http://localhost:8080/>

Steps:

- 1.) Type your username then click on connect**

- 2.) Type your message then click on send message**

| | |
|-------------------|-----------------------|
| Username | Connect to the server |
| Type a message... | Send |

If client could connect to the server the „Connect to the server” button background will become green

| | | |
|-------------------|-----------------------|--|
| first | Connect to the server | |
| Type a message... | Send | |

- first joined!

- 3.) Repeat the same thing with another client, after that the first one should see that the second one joined:

| | |
|-------------------|-----------------------|
| first | Connect to the server |
| Type a message... | Send |

- first joined!
- second joined!

4.) After this users are able to send messages:

| | |
|----------|------------------------------|
| first | Connect to the server |
| message1 | |
| | Send |

- first joined!
- second joined!
- Message from first:message1
- Message from second:message2

5.) Stop the server with CTRL+C

| | |
|----------|------------------------------|
| first | Connect to the server |
| message1 | |
| | Send |

- first joined!
- second joined!
- Message from first:message1
- Message from second:message2

REST endpoint

1.) Get all of the messages from database:

URI: <http://localhost:8080/storedMessages>

Output:

```
{
  "messages":
  [{
    "id":1,
    "content":"message1",
    "sender":"first",
    "type":null
  },
  {
    "id":2,
    "content":"message12",
    "sender":"first",
    "type":null
  },
  {
    "id":3,
    "content":"message123",
    "sender":"first",
    "type":null...
  }
]
```

2.) Get all of the users who used the application since startup:

URI: <http://localhost:8080/users>

Output:

```
{
  "users": [
    "first",
    "second"
  ]
}
```

3.) Get the server uptime since it was started:

- a. Human readable:

URI: <http://localhost:8080/uptime?format=true>

Output: `{"uptime": "455 seconds"}`

- b. In msec:

URI: <http://localhost:8080/uptime?format=true>

Output: `{"uptime": 561819}`

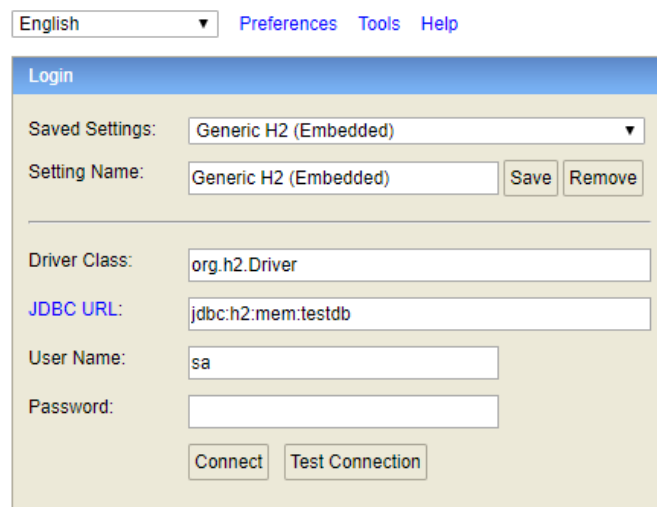
- 4.) Get service version:

URI: <http://localhost:8080/version>

Output: `{"version": "0.0.1"}`

Database Access

- 1.) Open in browser: <http://localhost:8080/h2-console>
- 2.) On the page click on connect (if it is not working check if all the parameters are the same)



The screenshot shows the H2 Console Login page. At the top, there is a language dropdown menu set to 'English' and navigation links for 'Preferences', 'Tools', and 'Help'. The main section is titled 'Login' and contains the following fields and buttons:

- Saved Settings:** A dropdown menu showing 'Generic H2 (Embedded)'.
- Setting Name:** A text input field containing 'Generic H2 (Embedded)', with 'Save' and 'Remove' buttons to its right.
- Driver Class:** A text input field containing 'org.h2.Driver'.
- JDBC URL:** A text input field containing 'jdbc:h2:mem:testdb'.
- User Name:** A text input field containing 'sa'.
- Password:** An empty text input field.
- Buttons:** 'Connect' and 'Test Connection' buttons at the bottom.

- 3.) On the next page select the „MESSAGE” table and type in the following query (if not prepopulated): `SELECT * FROM MESSAGE`, then click on Run
On this page the user should see all the messages sent since the server start up.

The screenshot shows a database client interface. On the left, a tree view displays the database structure: jdbc:h2:mem:testdb, MESSAGE, INFORMATION_SCHEMA, Sequences, Users, and H2 1.4.193 (2016-10-31). The main area contains a query editor with the text `SELECT * FROM MESSAGE`. Above the editor are buttons for 'Run', 'Run Selected', 'Auto complete', and 'Clear'. Below the editor, the query result is displayed as a table with 6 rows and 3 columns: ID, CONTENT, and SENDER. The data is as follows:

| ID | CONTENT | SENDER |
|----|------------|--------|
| 1 | message1 | first |
| 2 | message12 | first |
| 3 | message123 | first |
| 4 | m1 | second |
| 5 | m12 | second |
| 6 | m123 | second |

Below the table, it indicates '(6 rows, 4 ms)' and there is an 'Edit' button.