

TP 1 Redis Report

CAO Anh Quan
Architectures for Massive Data Management

September 28, 2018

Contents

1	Introduction	1
2	Database Design	2
2.1	Article	2
2.2	User	2
3	Application Design	3
4	Publish and Subscribe detail	4
4.1	Publish in Server	4
4.2	Subscribe in Client	4
5	Result	4
5.1	Guideline	4
5.1.1	Server	4
5.1.2	Client	5
6	Conclusion	7

1 Introduction

In this tp, our objective is to create a **publish-subscribe News system**. I created the system using Java programming language and Redis as a database, and I also add **more functions** to make the application better. There are 2 main parts in my system: **Server** and **Client**

- **Server:**
 - **Save** article into Redis database.
 - **Publish** article id to corresponding channels.
 - **Show all** articles in Redis database.
- **Client:**
 - User can **register** using username and password.
 - User can **log in** using username and password.
 - User can **subscribe** to a channel.

2 Database Design

2.1 Article

In this system, **article** contains **body**, which is the content of the article, and **tags**, which is the corresponding topic of that article. To identify an article, I associate each article with an **unique ID** which is a number. I use **atomic INCR** operation to create **unique ID** for each article. I use the following code to insert an article into the Redis database.

```
INCR next_article_id
```

I assume INCR next_article_id return 1. Then I use this value as an id for the new article and I append this value 1 to string "article:" to create **a key** to store an article as following.

```
HMSET article:1 body articleBody tag sport,politic.
```

In the Server, I need to **show all** articles. Therefore, I need to store all article IDs in a list **users** as following.

```
RPUSH articles 1
```

1 is an example for article id. When I need to get all articles, I need to query all article ids from **articles** list.

2.2 User

I apply the same method for storing an user with username and hashed password. I also use **atomic INCR** for creating unique **user ID** and then append ID to the string "user:" to create **a key** to store that user.

```
INCR next_user_id
```

```
HMSET user:1 username quan password ad4f596c85c6fd9fad1cbd30efb611c2.
```

I also need to create a **hash** in Redis with **username** as key and value is user **ID**. Because, when a user logs in, she will input her *username and password* and our job is to find her **ID** using her **username**.

HSET users quan 1

where 1 is an ID of user with **username** quan.

3 Application Design

In this application, I use the **multitier architecture** to separate presentation, logic handling, and data management functions. I divide our application into **three tier**: Repository (Data Access), Service (Logic Handling), Application (Server and Client - presentation).

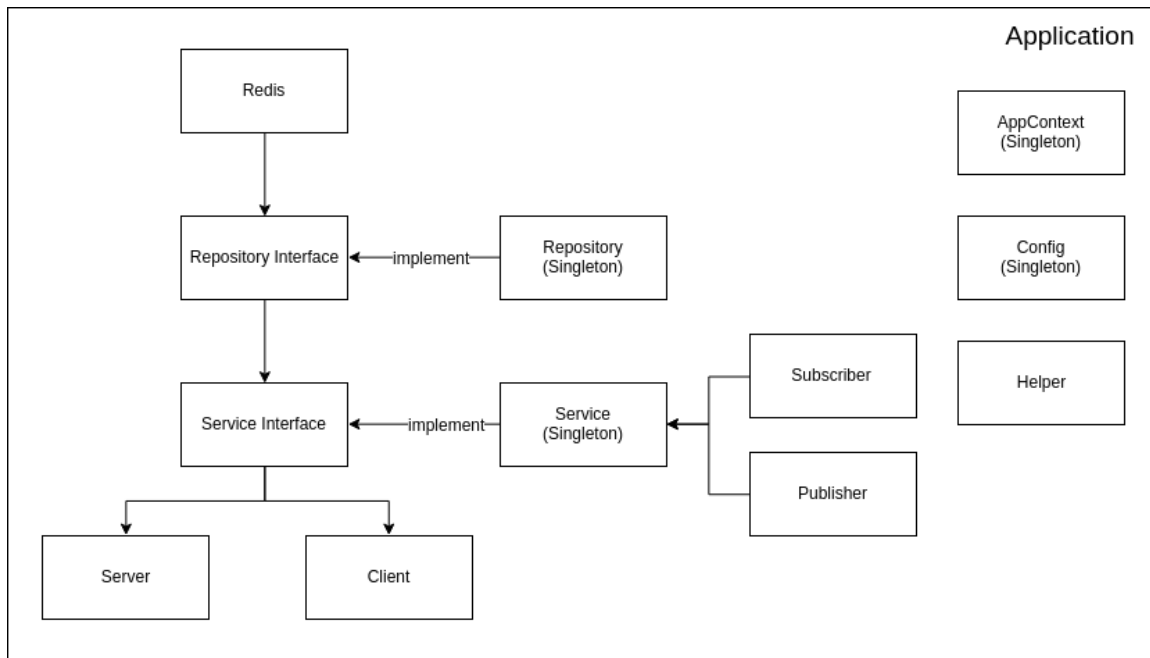


Figure 1: Application Design

I use the **Repository design pattern** for data access tier and **Singleton design pattern** for all Repository and Service class to avoid **redundant resource creation** and allow **share resource** in the application the AppContext and Config class.

For the Repository and Service, I use **interface** instead of directly inject concrete class because when I change the detail implementation of one tier, I will not need to change other layers.

I handle the **Publish** and **Subscribe** functions in the Service tier.

4 Publish and Subscribe detail

4.1 Publish in Server

When we create a new article in Server, we need to add **article body** and article tags.

1. The server **saves** the article into Redis database and get the **article ID**.
2. It split **tags** into separate **words**.
3. It publish the **article ID** to all channels using keyword in the tags. For example: If the tags is: "sport,culture", then the server will publish to channels: "channel:sport", "channel:culture".

4.2 Subscribe in Client

1. User need to **log in or register**.
2. User input the keyword (tag) that she wants to subscribe. For example sport
3. The Client subscribe to the corresponding channel. For example: "channel:sport"

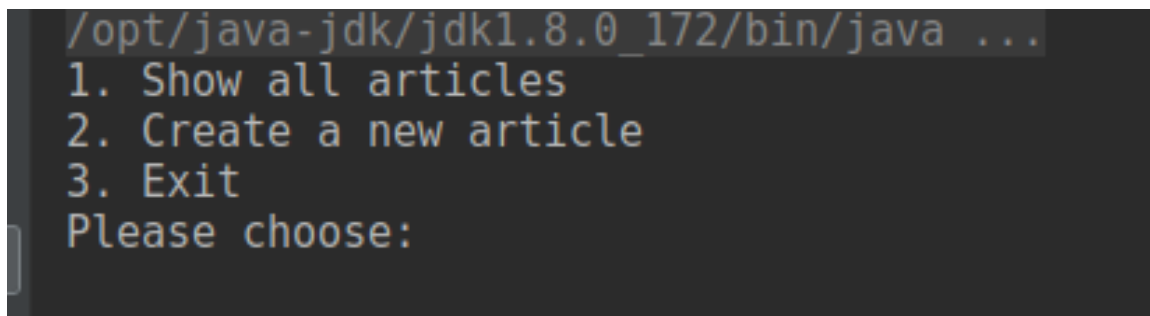
5 Result

5.1 Guideline

First, we need to import this project as **Maven project**.

5.1.1 Server

1. Start the Server's main method.



```
/opt/java-jdk/jdk1.8.0_172/bin/java ...  
1. Show all articles  
2. Create a new article  
3. Exit  
Please choose:
```

Figure 2: Main menu

2. Choose 1 to show all articles.

```
1. Show all articles
2. Create a new article
3. Exit
Please choose: 1
Article{id=1, body='hello world', tags='sport,world'}
Article{id=2, body='hello', tags='sport'}
Article{id=3, body='test', tags='sport'}
Article{id=4, body='test', tags='sport'}
Article{id=5, body='abc', tags='test'}
```

Figure 3: Show all articles in the database

3. Choose 2 to create new article and publish to clients

```
1. Show all articles
2. Create a new article
3. Exit
Please choose: 2
Please input article body
hello world
Please input article keywords separated by comma:
sport,culture
PUBLISH to channel:sport
PUBLISH to channel:culture
```

Figure 4: Save new article into database and publish to corresponding channels

5.1.2 Client

1. Start the client's **main** method.

```
/opt/java-jdk/jdk1.8.0_172/bin/java ...  
1. Log in  
2. Register  
3. Exit  
Please choose:
```

Figure 5: Client's menu

2. We can **register or log in** to the client

<pre>Please choose: 1 Please input your username quan1 Please input your password 123 Log in successfully 1. Subscribe to channel 2. Exit Please choose:</pre>	<pre>1. Log in 2. Register 3. Exit Please choose: 2 Please input your username quan4 Please input your password 123 Log in successfully 1. Subscribe to channel 2. Exit Please choose:</pre>
--	--

Figure 6: Register and Log in

3. We input the **tag** that we want to **subscribe** and waiting for the article creation from the server.

```
1. Subscribe to channel
2. Exit
Please choose: 1
Please input a tag: sport
1. Subscribe to channel
2. Exit
Please choose:
-----SUBSCRIBE Begin-----
SUBSCRIBE Channel: channel:sport
=====
channel:sport
Article id: 15
Article body: This is sport article
```

Figure 7: Client Subscription

6 Conclusion

In this tp, I implemented the **publish-subscribe completely news system** as in the lab requirement. I also add more functions and implement some application best practice such as design patterns and multi-tier architecture to make the application better and easier to maintain.