**CRON JOBS:**

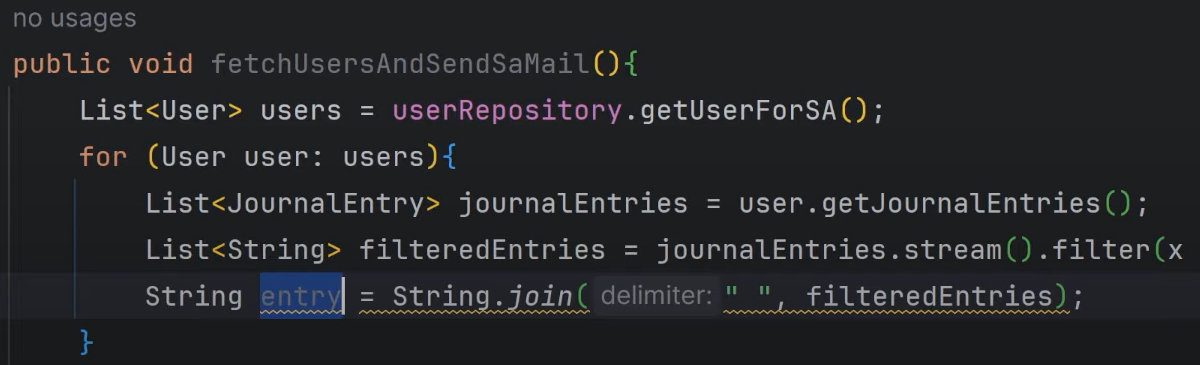
For the date data type, we would be filtering the content from 7 days old using the following expression:

**A computer screen with white text

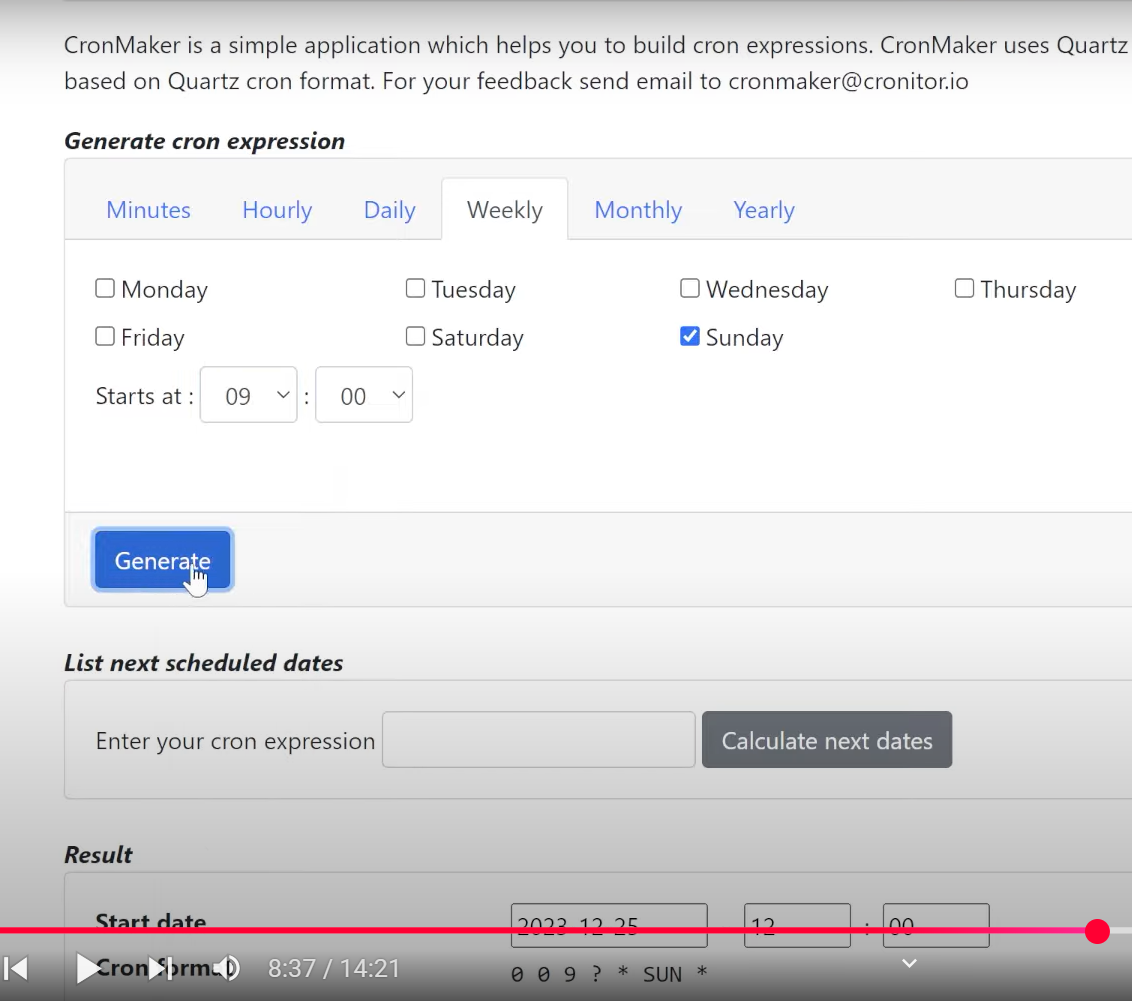
AI-generated content may be incorrect.**

And also we will be returning the content from that part.

We can also join this using the String.join function:



There is cron maker expression generate, where we can generate the expression and use that in the @Scheduled annotation as shown below:

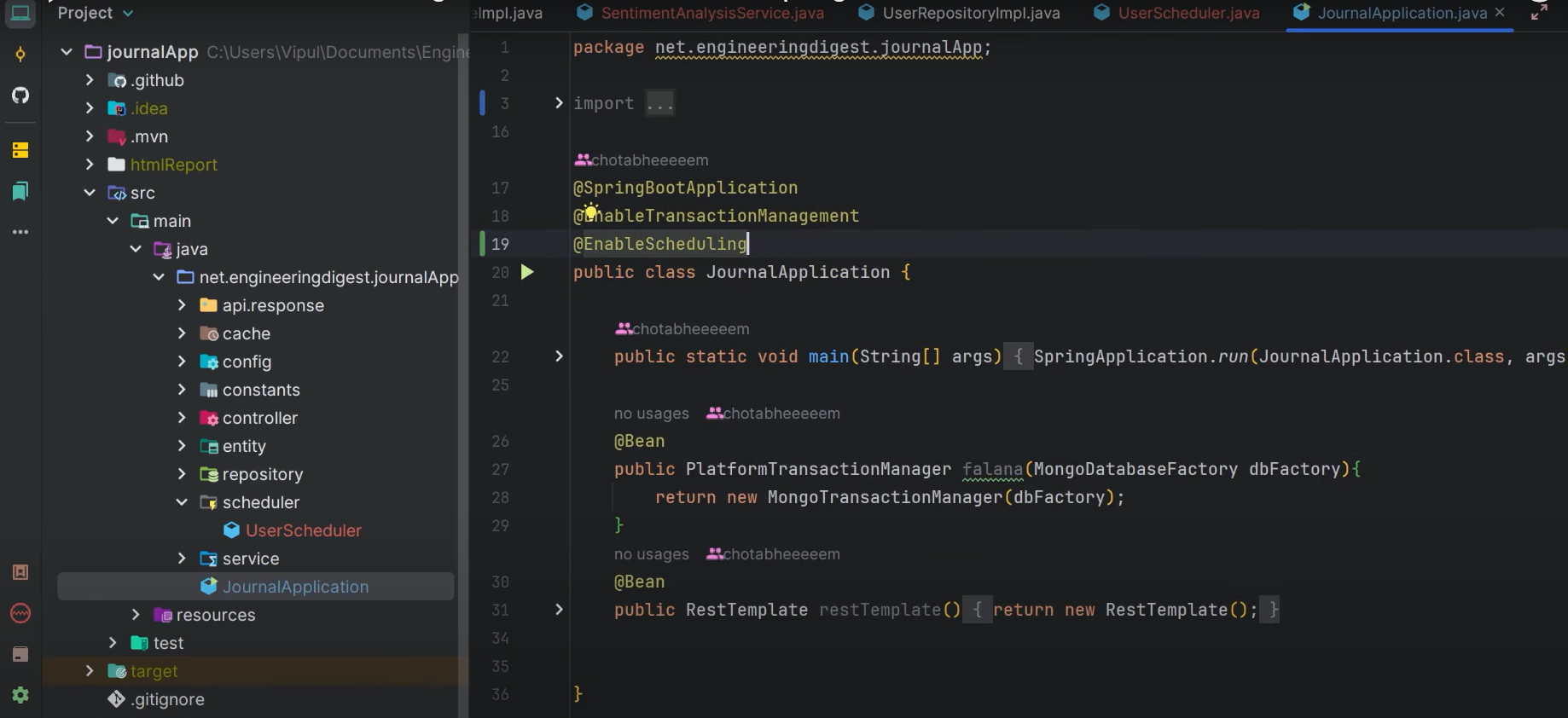


A screen shot of a computer program

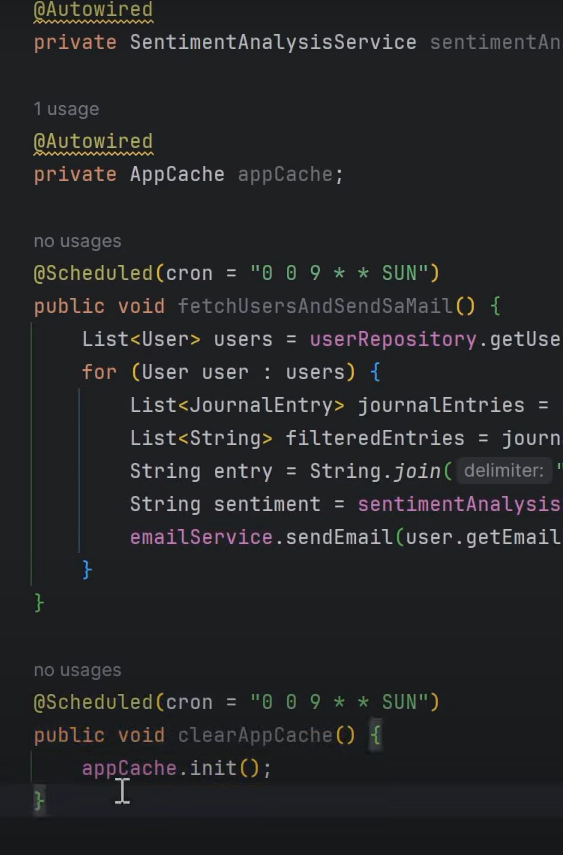
AI-generated content may be incorrect.

(0 0 9 \* \* SUN) -> first field is second, second is minute, third is hour, first \* is day of the month, second \* is month and SUN is day of the week.

@EnableScheduling annotation needs to be added in the main class for enabling the cron jobs.



**Special case:** If we want to clear the cache every 5 minutes using the cron job, we can also do that adding the @Schduled annotation on the method and calling the init() method of the App cache.



For every ten minutes:

A screenshot of a computer

AI-generated content may be incorrect.

**REDIS:**

A screenshot of a computer

AI-generated content may be incorrect.

Redis is in memory cache, data is getting read from the main memory rather than hard disk.

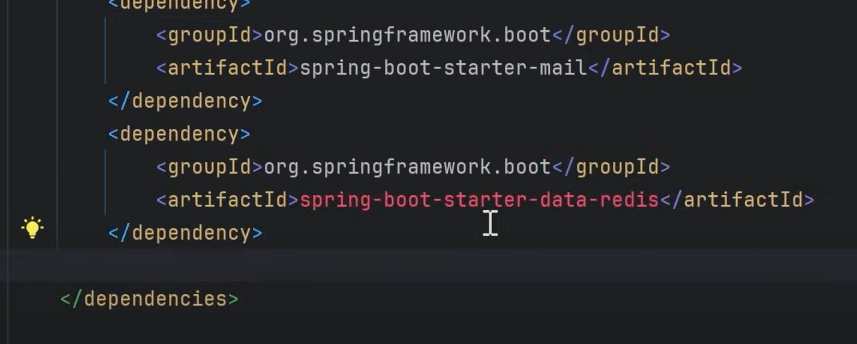
To access the data from the RAM, it takes the nanoseconds, while the data stored in the hard disk takes milli seconds.

IMPORTANT: if you are searching for the plans for the Netflix, so if the user is trying to access the plans for the same country, without login, so the response is same for every user, and if more than 1 crore users are hitting the API, and that hit is querying the database, then we unnecessarily hitting the DB, API and wasting the resources.

So, we can add the layer of REDIS in between, but if we are already using the cache, then what is the need to REDIS.

As REDIS is providing additional functionality of data structures, cache eviction strategy. Also you can limit the cache time for which the data needs to be read from the cache and set the time interval for which again DB needs to be hit for updating the response.

INTEGRATION OF REDIS IN SPRING BOOT:



Also, we need to use the REDIS host and REDIS port same as per the MongoDB.

Firstly, we need to install REDIS on windows, as it does not operate on windows directly, we need to use the windows subsystem for Linux to use the REDIS on windows.

* wsl –install

To install the linux subsystem for windows. After restart, click on the wsl to start and type the following command to check for the wsl installation.

* wsl -l -v

Command to install REDIS:



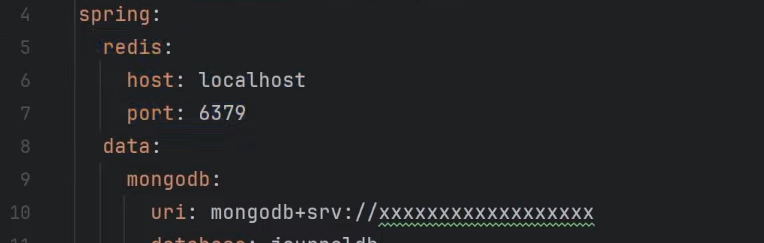
Following are the REDIS commands needs to be executed:

* curl -fsSL [https://packages.redis.io/gpg](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa2dwXzZRMWRkY3FPYTRjN2dmc2JfRUdDT1ZzZ3xBQ3Jtc0tuSTkyVVhrdldZcUl2UHg3N3hhQWNGMS14NWs2MzFqV1NhamU2SG5xOFoybjJkRW00SmFHZFg1NWhCODhDNkdNc2trVzR5ZUJJa1A1bEZTVjRpYTdHTFNPSl9qR0t2Z0NYTy1fS0Rpc1h5UTJ4YU9FOA&q=https%3A%2F%2Fpackages.redis.io%2Fgpg&v=2srQ-RiJHps) | sudo gpg --dearmor -o /usr/share/keyrings/redis-archive-keyring.gpg
* echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg] [https://packages.redis.io/deb](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbDRJMGh4eE5ZNVMydHFsVmRkZy10Znc3d3h4QXxBQ3Jtc0tuYVFfcERQNDFUVzZDeUQtRWtqb1IwWnBBXzNNNDhVWnJMRTMxRnZFMVZIWU1ESGtzTUt5cG9QS1FUaVpEbWVjdjBNVEc2MTlZTUQ3ZW5ZMmlZZjQyTlhtQ0laWmc1Q3c1eXNKaFlzRnJkV1JqM1FoZw&q=https%3A%2F%2Fpackages.redis.io%2Fdeb&v=2srQ-RiJHps) $(lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/redis.list
* sudo apt-get update
* sudo apt-get install redis
* sudo service redis-server start
* redis-cli

redis-cli command would give the details of the URL which gives the localhost and the port on which the redis is running.

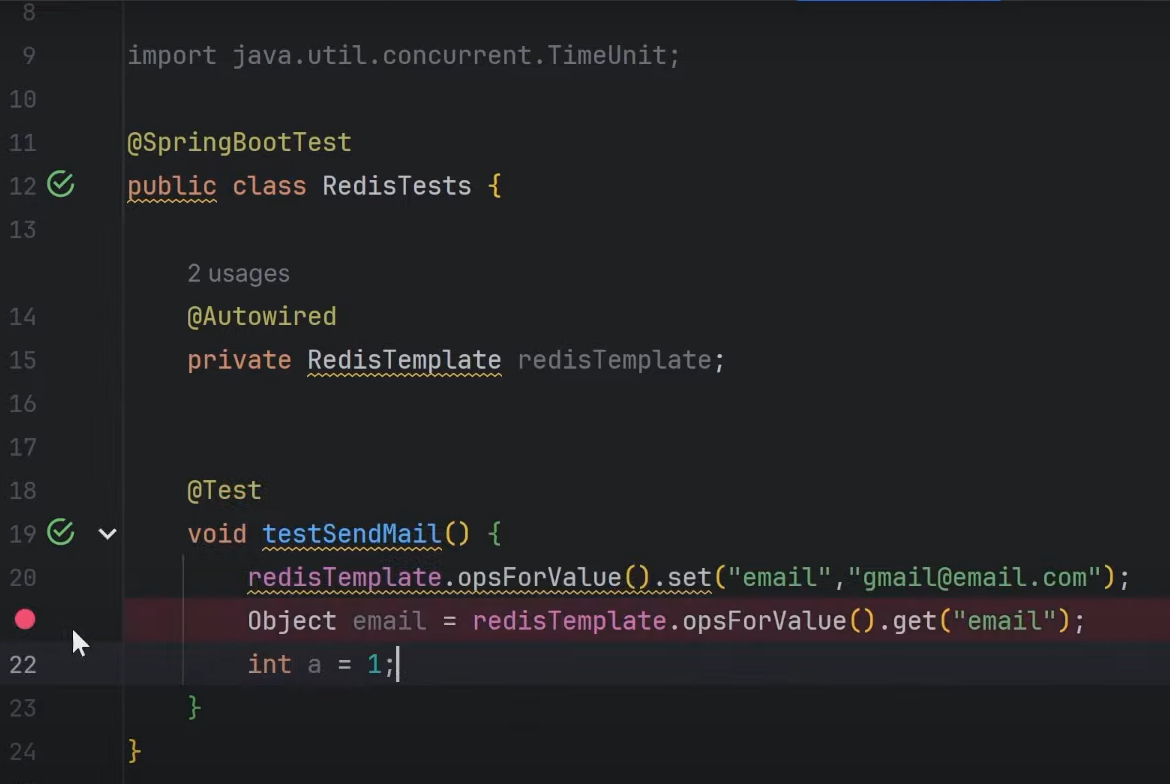
As when you type the command, ping you’ll receive the command pong.

To give the REDIS port and host, we can provide the following entry in application.yml file:



To interact with REDIS, we need to use the REDIS Template.

And now if we are setting up the value of the email and getting the same in the spring boot, we can get that, as shown below:

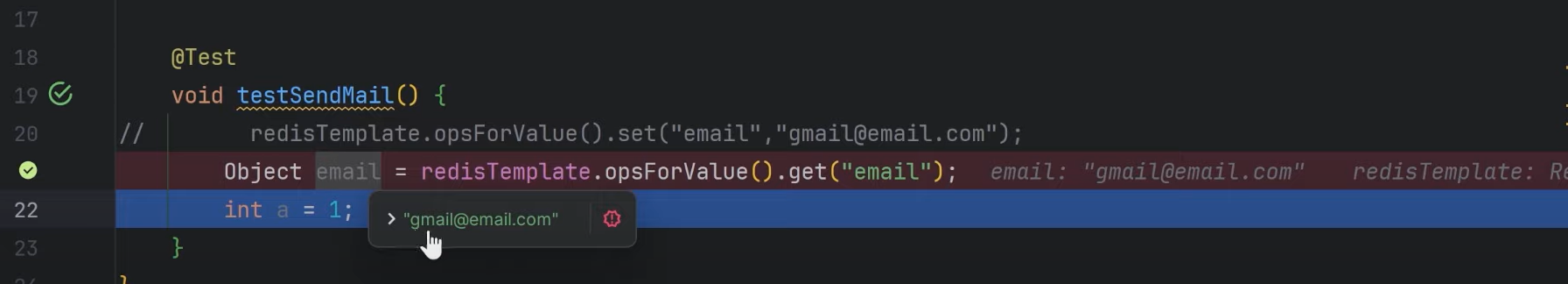


It is getting set in the project as shown:

A computer screen with text and symbols

AI-generated content may be incorrect.

Now if we comment out first line and try to get the email again, we’ll be getting the value of the email as:



Now, if we try to get the same in the command line, using the same key for the email, we’ll not be getting any value.

A computer screen with white text

AI-generated content may be incorrect.

And now if try to get the same salary in the spring boot, we would not be able to get that as shown below:

A computer screen with text on it

AI-generated content may be incorrect.

The reason for that is, REDIS is not using the same serializer and deserialization, which is using in the command line, so we need to set the serializer and deserializer in the spring boot and will create a new config class as RediConfig as shown below:

A computer screen shot of a program code

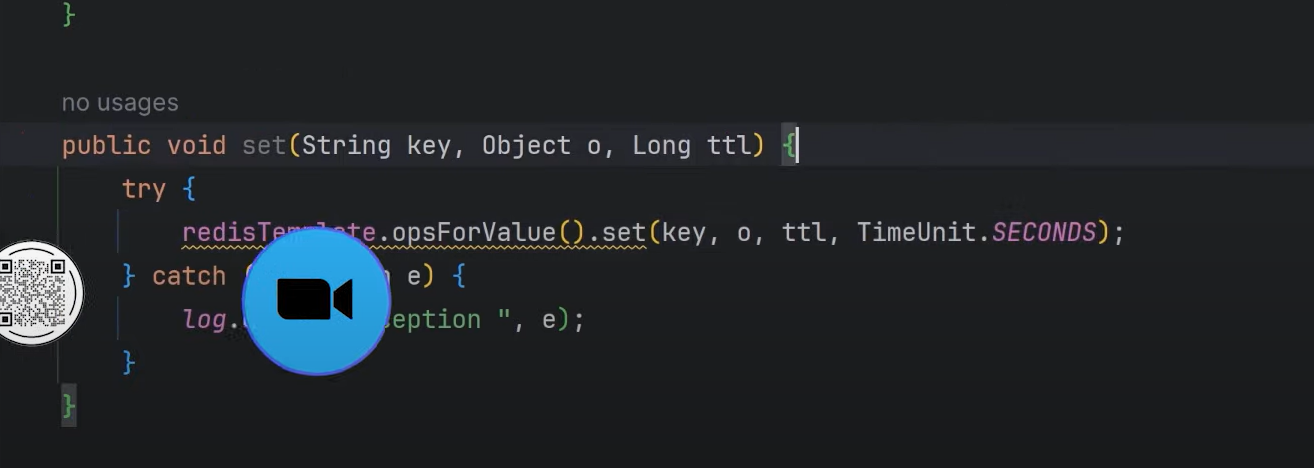
AI-generated content may be incorrect.

Now we’ll be using the sending out the response using REDIS cache. And we’ll be creating the REDIS service.

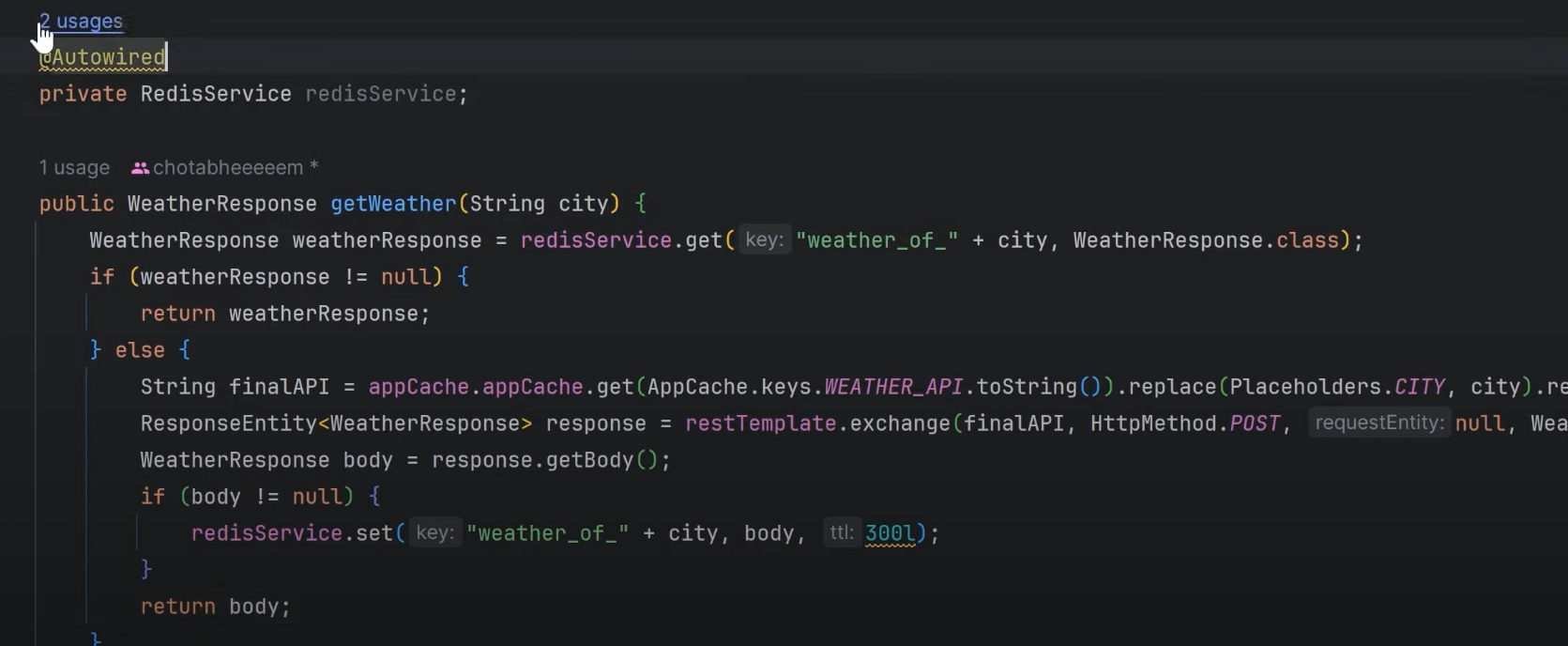
The following is the get() method:



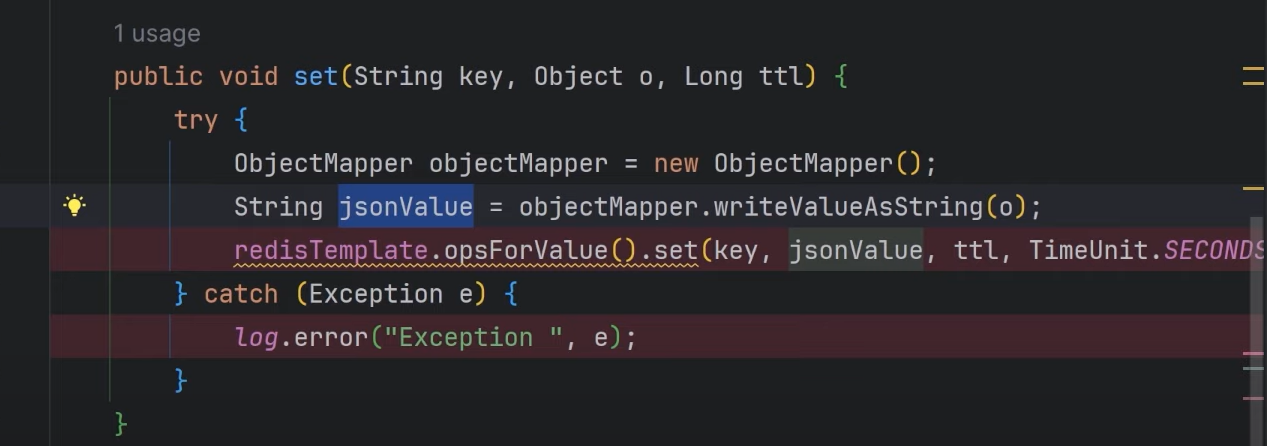
The following is the set() method:



Returning the response from REDIS cache if available, otherwise read from the database, when ttl expires.



There might be issue with the serializer used, so we need to use the ObjectMapper to convert the object to String as shown below:



**Integrate the spring boot with redis cloud: PENDING**

Need to login to redis cloud to manage the connection with redis. With REDIS, we need to provide the specific vendor which can be AWS, Google cloud or Microsoft Azure.

NEED TO IMPLEMENT THE SAME IN THE LOCAL LAPTOP, then will get the proper idea.

**KAFKA IN SPRING BOOT:**

A black background with white text

AI-generated content may be incorrect.

**A black background with white text

AI-generated content may be incorrect.**

Now if users are doing the like, comment or subscribing the email service and there is no kafka and direct communication between the services, so if millions of users are liking, then like\_service API would be hit millions of times, which would make flow very slow, and may be is the service stops, then would be issue and data would be lost.

Now when we introduce the kafka in between the services, then there would be asynchronous communication between the services and kafka would distribute data between multiple servers and data would be consumed parallelly, so SCALIBILITY would be achieved, FAULT TOLERENECE, DECOUPLING would be achieved.