**Redis Cache**

Redis is an in-memory data structure store that can be used as a cache. Redis Cache is a popular use case for Redis, and it offers several features that make it suitable for caching data in various applications. Here are key aspects of using Redis as a cache:

* In-Memory Storage:

Redis stores data in-memory, which allows for extremely fast read and write operations. This is ideal for caching frequently accessed data to improve application performance.

* Key-Value Store:

Redis is a key-value store, and each piece of data in the cache is associated with a unique key. This simplicity makes it easy to store and retrieve data quickly.

* High Throughput and Low Latency:

Redis is known for its high throughput and low latency, making it suitable for applications that require quick access to cached data.

**Steps to run the Redis server on Windows**

* Download Redis:

Visit the official Redis download page: <https://redis.io/download>

* Download the latest stable release for Windows.
* Extract the Archive:

Extract the downloaded archive to a directory of your choice.

* Run Redis Server:

Open a command prompt or PowerShell window.

Navigate to the directory where you extracted Redis.

Run the following command to start the Redis server:

***redis-server.exe***

* Open another command prompt or PowerShell window.
* Navigate to the Redis directory.
* Run the following command to start the Redis command-line interface:

***redis-cli.exe***

You can see the logs whenever the Api of the application is hit by the user

Redis Cache was implemented for the Demo Project given.

**Dependencies Used:**

<dependency>

<groupId>org.springframework. boot</groupId>

<artifactId>spring-boot-starter-data-redis</artifactId>

</dependency>

**Configuration in application properties**

redis:

host: localhost

port: 6379

cache:

type: redis

**Annotations used:**

@Cacheable

The **@Cacheable** annotation is part of the Spring Framework's caching support and is used to declare a caching behavior for a method

@CacheEvict

The **@CacheEvict** annotation in Spring is used to remove entries from a cache.

Redis Implementation

* Write a method where it will call the other method. Annotate with @Cacheable.
* This will make this method store in the cache.
* To verify that, print the simple statement to verify it.
* Use the same value and write the other method but make sure the value variable should be same for both the methods with the @CacheEvict annotation.
* Hit the Api, if you got the output along with statement printed, its coming from the cache.
* If you got the output with no statement printed, it is from the database.

*public interface EmpRepository extends MongoRepository<Employee, Integer>*

*{*

*Employee findByEmpFirstName(String firstName);*

*@org.springframework.cache.annotation.Cacheable(value="firstName")*

*default Employee myTest(String firstName) {*

*System.out.println("Cache is working");*

*return findByEmpFirstName(firstName);*

*}*

*@CacheEvict(value="firstName",allEntries = true)*

*@Override*

*<S extends Employee> S save(S entity);*

*}*