==========================================

# Spring Boot with Redis Cache Integration

==========================================

==============

## Introduction

==============

-> Every application will interact with database to store and retrieve data

-> In DB we will maintain data in tables (Rows & Columns)

-> DB tables are divided into 2 types

1) Transactional Tables

2) Non-Transactional Tables

-> If our application is perfoming INSERT / UPDATE / DELETE operations in table then it is called as Transactional Table.

-> If our application is performing only SELECT operation in the table then it is called as Non-Transactional table.

Examples :

-> When student register, then our application will insert student details into STUDENT_DTLS table (This is Transactional Table)

-> When student open 'Register For Demo' pop-up it will display COUNTRY Name with Country Code to enter Mobile number. Here our application will read the data from COUNTRY_MASTER table. In our application we don't insert / update / delete data in COUNTRY_MASTER table (This is Non-Transactional table)

-> If we retrieve contries data from table for multiple times also the data will be same and un-necessarily multiple queries will execute and it reduces our application performance.

-> When we have this kind of scenario then its better to go for Cache Memory.

-> Static tables data we should retrieve only one time and store into Cache memory.

-> When our application needs that static we can retrieve from Cache instead of Retrieving from DB.

-> Cache will be created in JVM and our application is also running in same JVM hence retrieval will be faster where as  DB calls are costly

===============

What is Cache ?

===============

-> Cache is a memory which is used to store the data in key-value format.

-> If our application having a requirement to use static data then it is highly recommended to use Cache in that application.

-> Using cache data we can reduce no.of db calls and we can improve performance of the application.

-> If we want to use cache in one application then we will go for Local cache. Local cache means we will implement with in our application.

-> If we want to use cache data in several applications then we will go for Global Cache.

=======================

How to implement cache

=======================

-> Once application got started take data from db and store that into cache in the form of key-value pair.

-> When application needs that data take it from cache memory instead of making db call.

-> By using cache memory we can reduce no.of db calls hence application performance will be improved.

==============================================

Q) What is Local cache & What is distributed cache ?

==============================================

-> If we implement cache with in one application then it called as local cache.

-> If we want to use cache data in multiple applications then we should go for distributed cache (Ex: Redis cache)

=======================

Q) What is Redis?

=======================

-> Redis is an open source cache which is used to store the data in the form key-value pair.

-> Multiple applications can connect to Redis cache to access the data.

=======================

Redis Environment Setup

=======================

1) Download redis-server software

2) Extract downloaded zip file

3) Double click on redis-server.exe file (redis server will start)

4) Double click on redis-cli.exe file

5) Type ping in client cmd it will respond with PONG.

Note : With this redis server setup is ready.

Note: In realtime, redis-server will be installed in remote machine and they will share redis-server details (url, uname, pwd and port)

==============================================

Spring Boot Application with Redis Integration

==============================================

1) Create Spring Boot application with below dependencies

    1) spring-boot-starter-web

    2) spring-boot-starter-redis

3) project lombok

4) devtools

5) swagger & swagger-ui

2) Configure JedisConnectionFactory bean with Redis Server details & inject JedisConnectionFactory bean into RedisTemplate bean

3) Create Binding class class

4) Create Service Class with required methods. Inject RedisTemplate into Repository bean and get HashOperations object to perform redis operations.

5) Create Rest Controller with required methods

6) Run the application and test it.

=====================================================================

Spring Boot with Redis Integration Project Git repo :

https://github.com/ashokitschool/Springboot_Redis_App

=====================================================================

========================================================================

1) Create Entity Class as per requirements to store in Redis DB server

========================================================================

```
package in.api.model;


import java.io.Serializable;


import lombok.Data;


@Data
public class User implements Serializable{


        private String id;
        private String name;
        private String email;


}
```

================================================================================
=========================

2) Create Service class for as per Entity to performed different operations such adding to the Redis DB,

getting single user or collection of Users from Redis DB

================================================================================
=========================

```
package in.api.service;


import java.util.Collection;
import java.util.List;
import java.util.Map;


import org.springframework.data.redis.core.HashOperations;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.stereotype.Service;

import in.api.model.User;

@Service
public class UserService {

	HashOperations<String,String,User> opsForHash = null;
	/*
	 * HashOperations<String,String,User>
	 *		HashName,Key,Value
	 */

	public UserService(RedisTemplate<String,User> redisTemplate) {
		this.opsForHash = redisTemplate.opsForHash();
	}

	public String addUser(User user) {
		opsForHash.put("USERS",user.getId(),user);
		return "User Added";
	}

	public User getUser(String userId) {
		System.out.println(opsForHash.get("USERS",userId));
		return opsForHash.get("USERS",userId);
	}

	public Collection<User> getAllUsers(){
		Map<String, User> entries = opsForHash.entries("USERS");
		return entries.values();
	}

}
```

====================================================================================
================================

3) Create Configuration class for Redis Debug with @Bean for JedisConnectionFactory for connection details

	and passing JedisConnectionFactory object to RedisTemplate for establishing connection with Redis DB Server

====================================================================================
================================

```
package in.api.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.jedis.JedisConnectionFactory;
import org.springframework.data.redis.core.RedisTemplate;

import in.api.model.User;

@Configuration
public class RedisConfig {

	@Bean
	JedisConnectionFactory jedisConnectionFactory () {
        JedisConnectionFactory jcf = new JedisConnectionFactory();
        return jcf;
    }

    @Bean
    RedisTemplate<String,User> redisTemplate() {
        RedisTemplate<String, User> redisTemplate = new RedisTemplate<>();
        redisTemplate.setConnectionFactory(jedisConnectionFactory());
        return redisTemplate;
    }

}
```

==================================================================================

4) Create Controller Class to add entity, get user as per requirement

==================================================================================

```
package in.api.controllers;

import java.util.Collection;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import in.api.model.User;
import in.api.service.UserService;

@RestController
public class UserController {

	private UserService service;

	@Autowired
	public UserController(UserService service) {
		this.service = service;
	}

	@PostMapping("/user")
	public String add(@RequestBody User user) {
		return service.addUser(user);
	}

	@GetMapping("/user/{userId}")
	public User getUser(@PathVariable String userId) {
		return service.getUser(userId);
	}

	@GetMapping("/users")
	public Collection<User> getAllUsers(){
```

```
                return service.getAllUsers();
        }
}
```

==================================================================

5) Dependencies

==================================================================

```
<dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-redis</artifactId>
                        <exclusions>
                                <exclusion>
                                        <groupId>io.lettuce</groupId>
                                        <artifactId>lettuce-core</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
                <dependency>
                        <groupId>redis.clients</groupId>
                        <artifactId>jedis</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>
```

```
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
			<scope>runtime</scope>
			<optional>true</optional>
		</dependency>
		<dependency>
			<groupId>org.projectlombok</groupId>
			<artifactId>lombok</artifactId>
			<optional>true</optional>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>
```

==================================================================