```
Load Balancing at the client side
=================================
   Ribbons in Microservices

=> By using Ribbon we can achieve client side LoadBalancing.

Note:
 -Dserver.port = 7777 (By suppling VM arguments to jvm we can run our application)
                          url : http://localhost:7777/welcome

 -Dserver.port = 6060
                       url : http://localhost:6060/welcome

                 refer: instances of application.png
                             SB-DEMO-REST-APP


Dependencies to get RibbonClient
===============================
<dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-ribbon</artifactId>
</dependency>

Code to Write RibbonClient
==========================
@FeignClient(name = "CURRENCY-EXCHANGE-SERVICE")
@RibbonClient(name = "CURRENCY-EXCHANGE-SERVICE")
public interface CurrencyExchangeClient {

      @GetMapping(value = "/getCost/from/{from}/to/{to}/")
      public CurrencyExchangeBean invokeCeApi(@PathVariable("from") String from,
@PathVariable("to") String to);

}

Distrubuted logging using Sleuth and Zipkin
===========================================
   logs => logging is a process of storing application execution details in a file
to monitor in future.
           By using logs we can understand where is the problem and what is the
problem in the application.

Steps to implement Distrubuted logging using Zipkin and Sleuth
  a. Download zipkin server (it is a jar file)
  b. run the zipkin server jar file from the command prompt
            syntax: java -jar <zipkin-jar-filname.jar>
       run the zipkin server using the following url : http://localhost:9411/

  c. Create one SpringBoot application with the following methods
            1. spring-boot-starter-web
            2. sleuth
            3. zipkin
            4. devtools
  d. Create one rest controller with the following methods
  e. Implement logging in rest controller methods
  f. Configure port no and run the application

                             Refer: Springboot-Sleuth-Zipkin-App
```

CacheImplementation

=> It is a temporary storage.
=> Whenever we need to access same data frequently,then its better to maintain that data in cache to improve performance.
=> If we try to read the same data from DB multiple times, then application performance will be decreased becoz db interaction is costly operation.
=> To reduce the burden on the application or to improve the performance we use "Caching".
=> In DB we have 2 types of table
a. Transactional tables
If the table data is getting inserted/update/deleted from the application then it is called as "Transactional tables"
b. NonTransactional tables
If the application is performing only read operation on the table, then that table is called as "NonTransactional tables".

Note: We should implement cache for "NonTransactional table".

=> If we implement cache in one project, then it is called as local cache,only that project can access the data from the local cache.
It is suitable only for monolith architecture based project.

=> When we implement project using Microservices architecture we will have multiple microservices,if we want cache data in all services we should go for global cache,It is called as "Distrubuted cache".

=> In microservcies "Distrubuted cache" can be implemented using "RedisCache".

What is RedisCache?
Redis is an in-memory data structure store, used as a distributed, in-memory key–value database, cache and message broker, with optional durability.
Redis supports different kinds of abstract data structures, such as strings, lists, maps, sets, sorted sets, HyperLogLogs, bitmaps, streams, and spatial indices.

Setting up Redis Server in Machine

1. Download Redis Server from the below url
   url : https://github.com/microsoftarchive/redis/releases
2. Download the zip file and open the downloaded folder
3. click on RedisServer.exe file to start the Redis server
4. click on Rediscli.exe file to start the Redis client
5. Once Redis Client is started just execute 'ping' it should given PONG as a response

Note: with the above steps Redis Server set up is done in our machine.

Commands to Store data in Redis cache in the form of key-value
```
  set key "value"
      eg: set name "sachin"
```

Commands to Get the value based on data in Redis cache
```
  get key
      eg: get name
```

```
commands to list all the key value pair data
  keys *

            refer:: SB-Redis-Cache-App

Thursday :: 7.00PM to 9.00PM (last session)
```