Java Redis Projects

Redis Requirements

Spring Redis requires Redis 2.6 or above and Spring Data Redis integrates with **Lettuce** **and** **Jedis**, two popular open-source Java libraries for Redis.
[crayon-6633b3cc7b7a1832173338/]
For Connection Factories : docs.spring.io/spring-data/data-redis/docs/current/reference/html/#redis:connectors
[crayon-6633b3cc7b7a7868645950/]
Sentinel Support Changes
[crayon-6633b3cc7b7aa589601867/]
Redis Configuration Sentinel :
[crayon-6633b3cc7b7ac781794963/]

Jedis Connector
[crayon-6633b3cc7b7af923272245/]

Working with Objects through RedisTemplate

[crayon-6633b3cc7b7b2667873885/]
[crayon-6633b3cc7b7b4991861794/]

String Focus: String-focused Convenience Classes

[crayon-6633b3cc7b7b6597296007/]
[crayon-6633b3cc7b7b8877292872/]

Hash Mappers

[crayon-6633b3cc7b7bc233293133/]

There are Listeners : Pub / Sub :

##
##

- Redis Project Directory D:\opensource-projects\repos\redis-notes\redisson\redisson-spring-data\redisson-spring-data-25\src
- docs.spring.io/spring-data/data-redis/docs/current/reference/html/#redis:connectors
- redis_kutayzorlu.com : PDF

Encryption and Programming

Redis & Management

Docker container:

- https://hub.docker.com/_/redis

Username :
[crayon-6633b3cc7c809603714848/]
SET

Timeout Record TTL

Clustering

Redis Sentinel Documentation

Redis Sentinel provides high availability for Redis. In practical terms this means that using Sentinel you can create a Redis deployment that resists without human intervention certain kinds of failures.

Redis Sentinel also provides other collateral tasks such as monitoring, notifications and acts as a configuration provider for clients.

This is the full list of Sentinel capabilities at a macroscopic level (i.e. the big picture):

- Monitoring. Sentinel constantly checks if your master and replica instances are working as expected.

- Notification. Sentinel can notify the system administrator, or other computer programs, via an API, that something is wrong with one of the monitored Redis instances.
- Automatic failover. If a master is not working as expected, Sentinel can start a failover process where a replica is promoted to master, the other additional replicas are reconfigured to use the new master, and the applications using the Redis server are informed about the new address to use when connecting.
- Configuration provider. Sentinel acts as a source of authority for clients service discovery: clients connect to Sentinels in order to ask for the address of the current Redis master responsible for a given service. If a failover occurs, Sentinels will report the new address.

Distributed nature of Sentinel

Redis Sentinel is a distributed system:

Sentinel itself is designed to run in a configuration where there are multiple Sentinel processes cooperating together. The advantage of having multiple Sentinel processes cooperating are the following:

- Failure detection is performed when multiple Sentinels agree about the fact a given master is no longer available. This lowers the probability of false positives.
- Sentinel works even if not all the Sentinel processes are working, making the system robust against failures. There is no fun in having a failover system which is itself a single point of failure, after all.

The sum of Sentinels, Redis instances (masters and replicas) and clients connecting to Sentinel and Redis, are also a

larger distributed system with specific properties. In this document concepts will be introduced gradually starting from basic information needed in order to understand the basic properties of Sentinel, to more complex information (that are optional) in order to understand how exactly Sentinel works.

Redis Replication

* https://redis.io/docs/manual/replication/
* Redis-replication-Redis_kzcom

ref: redis.io/topics/sentinel
DE: senti_redis

–
Redis'in High Availability Çözümü: Sentinel
redis_high_availibility_

–

Spring Boot – Annotations

@Transactional
[crayon-6633b3cc7cbe3393025001/]
@EnableScheduling
[crayon-6633b3cc7cbe7231262288/]

[crayon-6633b3cc7cbe9947887870/]

@ –

Reference : https://www.baeldung.com/spring-component-annotation

Spring Boot Modules

Spring modules. Some of the most commonly used ones are:

- *spring-boot-starter-data-jpa*
- *spring-boot-starter-security*
- *spring-boot-starter-test*
- *spring-boot-starter-web*
- *spring-boot-starter-thymeleaf*

———-

Sonar / Coding Quality Tools

Example Maven

Example Exclusions

[crayon-6633b3cc7cfcb395271874/]

Ref: ichi.pro/tr/sonarqube-8-2-toplulugu-ile-spring-boot-2-2-6-kod-kalitesi-123864276582261

TODO

Jenkins

Job: Bir jenkins projesidir. Otomatize etmek istediğimiz işleri burada belirleriz. Örneğin, job config üzerinden şu repository'i çek, şu şartlarda build et, şu testleri çalıştır ve belirlenen kişilere mail at gibi işlemleri burada belirleriz.

Node: job'un üzerinde çalıştığı sunucuyu ifade eder. Testleri başka bir bilgisayarda koşmak istediğimizde node oluşturur ve bağlantı için gerekli şartları gerçekleştirdikten sonra node'da testlerimizi koşabiliriz.

Plugin (eklenti): Jenkins saf haliyle yüklenir, ihtiyacımıza göre plugin yükler ve bunları kullanılırız. Örneğin, Job çalıştıktan sonra mail atması için "Email Extension" eklentisini yüklemeli ve post-build adımında kullanmalıyız.

Pipeline: işlerin ardışık bir sırada yapılması, bir işlemin çıktısının sonraki gelen işlemin girdisi olması anlamına gelir. Ör: Bir test adımının başarısız olması durumunda diğer bir testin hiç başlamaması gibi.

Before Linux Installation

[crayon-6633b3cc7d378481270138/]

Docker Container Management
[crayon-6633b3cc7d37c845910011/]

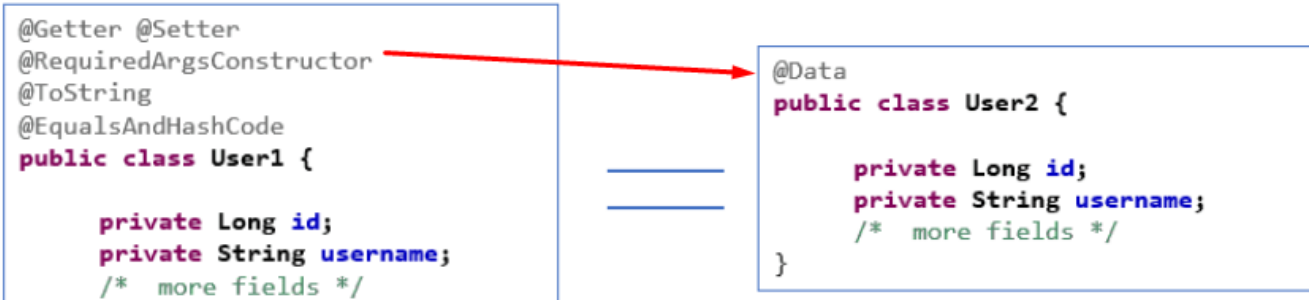
Creation of the Repo

Realtime Build Counts

Ref: www.jenkins.io/doc/book/installing/docker/

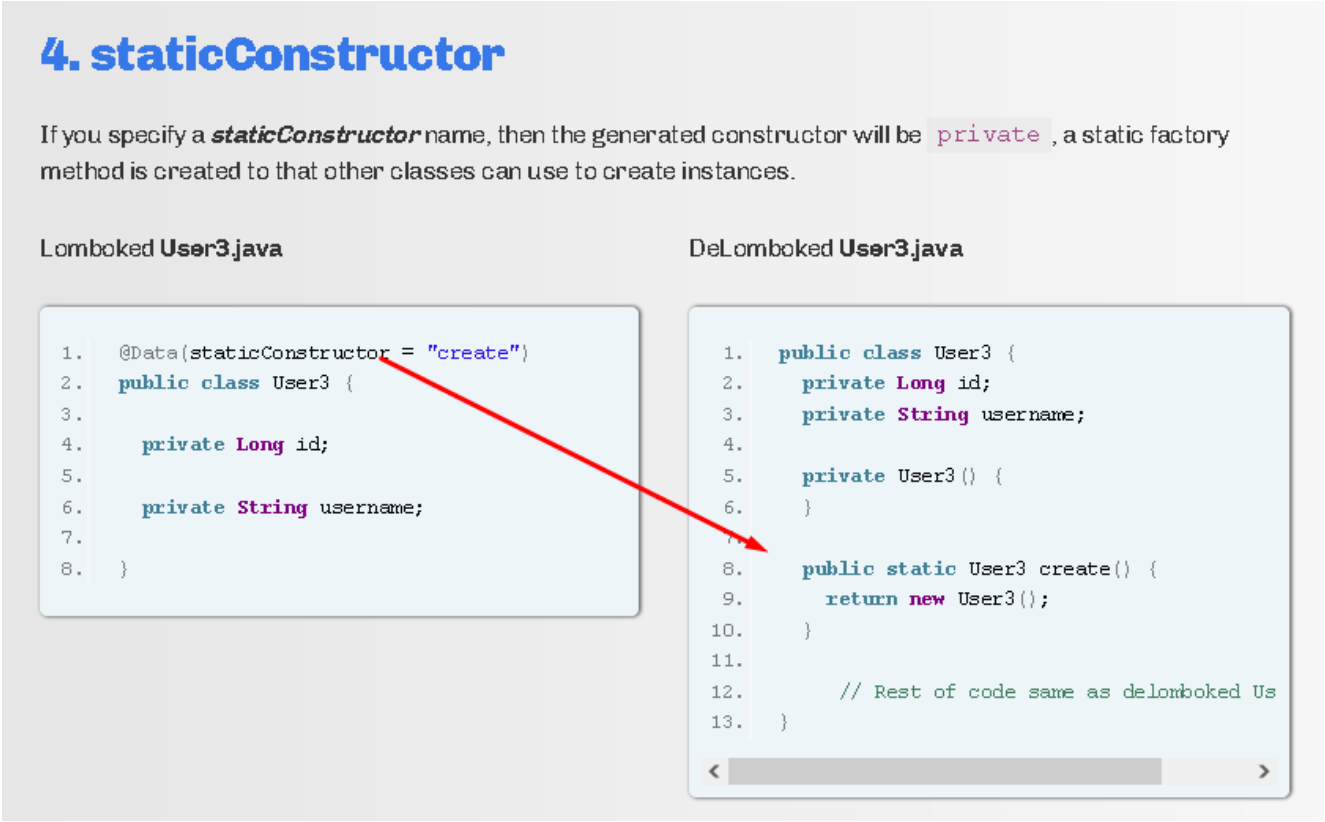
Lombok

[crayon-6633b3cc7d7ca139991887/]



Ref: javabydeveloper.com (All the content!)

[crayon-6633b3cc7d7ce531139302/]
[crayon-6633b3cc7d7d0888720301/]
[crayon-6633b3cc7d7d2076899224/]



[crayon-6633b3cc7d7d4853182195/]

[crayon-6633b3cc7d7d6269206713/]

---

4. staticConstructor

If you specify a ***staticConstructor*** name, then the generated constructor will be `private`, a static factory method is created to that other classes can use to create instances.

Lomboked **User3.java**

```
1.  @Data(staticConstructor = "create")
2.  public class User3 {
3.
4.    private Long id;
5.
6.    private String username;
7.
8.  }
```

DeLomboked **User3.java**

```
1.  public class User3 {
2.    private Long id;
3.    private String username;
4.
5.    private User3() {
6.    }
7.
8.    public static User3 create() {
9.      return new User3();
10.   }
11.
12.     // Rest of code same as delomboked Us
13. }
```

Sometimes you may want to define annotations on top of constructor, for example when you are working with Spring framework or some other third part java libraries, you may need to declare annotations on top of constructor. `onConstructor` attribute of `@AllArgsConstructor` allows us to put annotations on generated all-args constructor.

1. Up to JDK7: `@AllArgsConstructor(onConstructor=@__({@AnnotationsGoHere}))`
2. From JDK8: `@AllArgsConstructor(onConstructor_={@AnnotationsGohere})` // note the underscore after `onConstructor`.

Lomboked ***AllArgsDemo6.java***

```
1.  @AllArgsConstructor(
2.      onConstructor_=
3.      @ConstructorProperties({"id", "usernam
4.  public class AllArgsDemo6 {
5.
6.    private Long id;
7.
8.    private String username;
9.
10. }
```

DeLomboked ***AllArgsDemo6.java***

```
1.  public class AllArgsDemo6 {
2.    private Long id;
3.    private String username;
4.
5.    @ConstructorProperties({"id", "username"
6.    public AllArgsDemo6(final Long id, final
7.      this.id = id;
8.      this.username = username;
9.    }
10. }
```

If we would like to create instance using a `static` factory method, `staticName` attribute of `@AllArgsConstructor` allows us to generates a private all-args constructors and an additional `static` factory method that wraps around the `private` constructor is generated. Let's have a look into following example.

Lomboked ***AllArgsDemo5.java***

```
@AllArgsConstructor(staticName = "getInstance")
public class AllArgsDemo5 {

  private Long id;

  private String username;

}
```

DeLomboked ***AllArgsDemo5.java***

```
public class AllArgsDemo5 {
  private Long id;
  private String username;

  private AllArgsDemo5(final Long id, final Str
    this.id = id;
    this.username = username;
  }

  public static AllArgsDemo5 getInstance(final
    return new AllArgsDemo5(id, username);
  }
}
```

Lombok generates a public all-args constructor by default for the ***`@AllArgsConstructor`***. To generate `private` all-args constructor define `@AllArgsConstructor(access = AccessLevel.PRIVATE)`. `access` attribute of ***`@AllArgsConstructor`*** allows you to change the access modifier of the generated constructor.

Lomboked ***AllArgsDemo4.java***

```
@AllArgsConstructor(access = AccessLevel.PRIVATE
public class AllArgsDemo4 {

  private Long id;

  private String username;

}
```

DeLomboked ***AllArgsDemo4.java***

```
public class AllArgsDemo4 {
  private Long id;
  private String username;

  private AllArgsDemo4(final Long id, final Strin
    this.id = id;
    this.username = username;
  }
}
```

1. Lombok never generates constructor argument for the `static` fields for `@AllArgsConstructor`.
2. For `@AllArgsConstructor` Lombok never generates constructor argument for the `final` fields if they are initialized with value, otherwise an argument will be generated.

Lomboked ***AllArgsDemo3.java***

```
1.  @AllArgsConstructor
2.  public class AllArgsDemo3 {
3.
4.    private Long id;
5.
6.    private static boolean defaultStatus;
7.
8.    private final double minSalary = 10000.00;
9.
0.    private final int defaultRole;
1.  }
```

DeLomboked ***AllArgsDemo3.java***

```
blic class AllArgsDemo3 {
 private Long id;
 private static boolean defaultStatus;
 private final double minSalary = 10000.0;
 private final int defaultRole;

 public AllArgsDemo3(final Long id, final int def
   this.id = id;
   this.defaultRole = defaultRole;
 }
```

The fields marked with `@NonNull` in your class result in null check on those parameter within generated all-args constructor.

Lomboked ***AllArgsDemo2.java***

```
1.  @AllArgsConstructor
2.  public class AllArgsDemo2 {
3.
4.    private Long id;
5.
6.    @NonNull
7.    private String username;
8.  }
```

DeLomboked ***AllArgsDemo2.java***

```
1.  public class AllArgsDemo2 {
2.    private Long id;
3.    @NonNull
4.    private String username;
5.
6.    public AllArgsDemo2(final Long id, @NonN
7.      if (username == null) {
8.        throw new NullPointerException("user
9.      }
10.     this.id = id;
11.     this.username = username;
12.   }
13. }
```

```
1.  @AllArgsConstructor
2.  public class AllArgsDemo1 {
3.
4.    private Long id;
5.
6.    private String username;
7.  }
```

```
1.  public class AllArgsDemo1 {
2.    private Long id;
3.    private String username;
4.
5.    public AllArgsDemo1(final Long id, final
6.      this.id = id;
7.      this.username = username;
8.    }
9.  }
```

Delombok

[crayon-6633b3cc7d7d9858998649/]
—

—

—

—