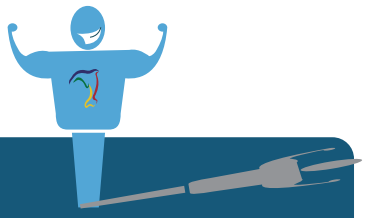
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

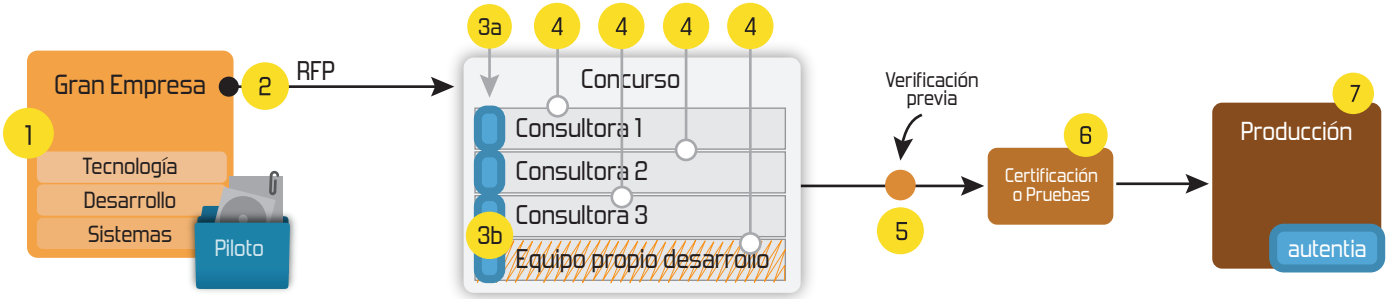
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Juan Alonso Ramos

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero en Informática, especialidad en Ingeniería del Software

Puedes encontrarme en Autentia: Ofrecemos de servicios soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2014-11-11

Tutorial visitado 1 veces Descargar en PDF

Soporte Redis con Spring: RedisTemplate

0. Indice de contenidos.

1. Introducción.

Spring Redis pertenece a la familia Spring Data. Nos proporciona utilidades para que de forma más sencilla podamos configurar el acceso a la base de datos NoSQL Redis y un template para realizar las operaciones más comunes sobre los datos almacenados.

En este tutorial veremos cómo instalar Redis en Mac OS y cómo configurar con Maven un proyecto para utilizar la clase RedisTemplate que nos da soporte para utilizar Redis. También crearemos un clase Repository que proporcionará las operaciones más comunes sobre una base de datos Redis.

2. Entorno.

El tutorial se ha realizado con el siguiente entorno:

- MacBook Pro 15' (2.4 GHz Intel Core i5, 8GB DDR3 SDRAM).
- Sistema Operativo: Mac OS Mavericks 10.9.5
- Oracle Java SDK 1.7.0_60
- Redis 2.8.11

3. Instalar Redis

Este tutorial está escrito con un MacBook con el sistema operativo Mac OS Mavericks por lo que si dispones de un Mac lo más fácil es utilizar el gestor de paquetes **brew**. Si no siempre puedes descargarlo de la web oficial. El comando para instalar Redis desde Mac es el siguiente:

```
1 brew install redis
```

Si todo ha ido bien ejecuta por consola la instrucción **redis-server** y se arrancará Redis.

```
jalonso@MacBook-Pro-Juan-Alonso:/usr/local/etc$ redis-server
[51780] 26 Oct 19:26:53.603 # Warning: no config file specified, using the default co
[51780] 26 Oct 19:26:53.606 * Increased maximum number of open files to 10032 (it was
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 2.8.11 (00000000/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in stand alone mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 6379
      |    `-._   `._    /     _.-'    |     PID: 51780
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

[51780] 26 Oct 19:26:53.611 # Server started, Redis version 2.8.11
[51780] 26 Oct 19:26:53.612 * The server is now ready to accept connections on port 6
```

Una vez arrancado el servidor desde otra consola podemos arrancar el cliente con **redis-cli** con el que poder lanzar nuestros comandos Redis, pero antes vamos a explicar un poco en qué consiste esta base de datos.

4. Pero qué es Redis

- Redis es un motor de almacenamiento de datos en formato clave-valor opensource.
- El modelo de datos en el que se basa es de tipo diccionario o tabla de hashes relacionando una clave con una estructura donde se almacena el valor asociado en diferentes formatos: strings, listas, sets, sorted sets y hashes.
- Se conoce principalmente porque su almacenamiento es muy rápido debido a que mantiene los datos en memoria pero también se persisten en disco.
- Puede atender cientos de miles de operaciones por segundo y es escalable.
- Redis soporta replicación maestro-esclavo y permite publicación-subscripción.
- Tiene una amplia lista de clientes diferentes con el que conectarnos al servidor Redis: C, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala, etc.

5. Configurar pom.xml

Ahora que conocemos un poquito más Redis, vamos a crear nuestro primero proyecto para realizar operaciones sobre los datos. Crearemos para empezar el pom.xml del proyecto Maven ayudados por spring-boot-starter-redis

El pom.xml quedaría así:

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.autentia.tutoriales</groupId>
    <artifactId>spring-redis</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.1.8.RELEASE</version>
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java.version>1.7</java.version>
    </properties>

    <build>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>${java.version}</source>
                    <target>${java.version}</target>
                    <encoding>${project.build.sourceEncoding}</encoding>
                </configuration>
            </plugin>
        </plugins>
    </build>

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

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

        <dependency>
            <groupId>org.twitter4j</groupId>
            <artifactId>twitter4j-core</artifactId>
            <version>4.0.2</version>
        </dependency>

        <dependency>
            <groupId>org.twitter4j</groupId>
            <artifactId>twitter4j-stream</artifactId>
            <version>4.0.2</version>
```


todos los plugins de forma automática

» Generación de vistas HTML5 con el soporte de JSF2: pass through

Últimos Tutoriales del Autor

» Monitorización de Apache Kafka

» Monta fácilmente tu proyecto con Spring Boot Starter POMs

» Primeros pasos con Apache Kafka

» Trident, un compañero de viaje para tratar con Storm

» Introducción a Apache Storm

Categorías del Tutorial

Spring

BBDD

```
        </dependency>
    </dependencies>
</project>
```

6. Configurar RedisTemplate

Spring Data nos da soporte para abstraernos del driver de conexión a Redis. A través de RedisTemplate nos proporciona una capa de alto nivel para tratar con las operaciones de lectura, escritura, búsqueda, actualización, borrado, etc. Por debajo se puede utilizar cualquier otra librería para trabajar con Redis ya que soporta las librerías Jedis, JRedis, Lettuce y SRP.

Vamos a crear una clase de configuración de nuestra aplicación donde levantar los beans JedisConnectionFactory encargado de ofrecernos una conexión a Redis, la clase RedisTemplate y nuestra clase Repository.

```java
package com.autentia.tutoriales;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.jedis.JedisConnectionFactory;
import org.springframework.data.redis.core.StringRedisTemplate;

import com.autentia.tutoriales.redis.StringRedisRepository;

@Configuration
@EnableAutoConfiguration
public class Application {

    private static final Logger LOGGER = LoggerFactory.getLogger(Application.class);

    @Bean
    JedisConnectionFactory connectionFactory() {
        return new JedisConnectionFactory();
    }

    @Bean
    StringRedisTemplate stringRedisTemplate(RedisConnectionFactory connectionFactory)
        return new StringRedisTemplate(connectionFactory);
    }

    @Bean
    StringRedisRepository stringRedisRepository(StringRedisTemplate template) {
        return new StringRedisRepository(template);
    }

    public static void main(String[] args) throws InterruptedException {
        LOGGER.debug("Initializating app...");

        SpringApplication.run(Application.class, args);
    }
}
```

Por defecto JedisConnectionFactory configura una conexión en localhost en el puerto por defecto de Redis 6379. Si tuvieras otra configuración en Redis debes configurar también tu JedisConnectionFactory.

Nuestro Repository será el encargado de encapsularnos las operaciones sobre Redis que efectúen operaciones de almacenamiento de tipo cadena. Algunas de las operaciones más utilizadas son:

- **add**: Añade una cadena asociada a una clave.
- **getBy**: Recupera una cadena por su clave.
- **getKeys**: Recupera las claves que cumplen determinado patrón.
- **getAllValuesBy**: A partir de un patrón devuelve todas los valores cuya clave lo cumplen.
- **delete**: Elimina una cadena por su clave.

```java
package com.autentia.tutoriales.redis;

import java.util.HashSet;
import java.util.Set;

import org.springframework.data.redis.core.StringRedisTemplate;

public class StringRedisRepository {

    private final StringRedisTemplate template;

    public StringRedisRepository(StringRedisTemplate template) {
        this.template = template;
    }

    public void add(String key, String value) {
        template.opsForValue().set(key, value);
    }

    public String getBy(String key) {
        return template.opsForValue().get(key);
    }

    public Set<String> getKeys(String patternKey) {
        return template.keys(patternKey);
    }

    public Set<String> getAllValuesBy(String patternKey) {
        final Set<String> keys = getKeys(patternKey);
        final Set<String> values = new HashSet<String>(keys.size());

```

```
        for (String key : keys) {
            values.add(getBy(key));
        }

        return values;
    }

    public void delete(String key) {
        template.opsForValue().getOperations().delete(key);
    }
}
```

Estas son unas pocas operaciones de las decenas de ellas que se pueden realizar si fueran necesarias pero para una pequeña prueba de concepto son suficientes.

Ya tenemos la aplicación montada, sólo nos quedan datos para meter en Redis. Para mostrar la velocidad de esta base de datos vamos a sacar datos de Twitter. Nos conectaremos con nuestras credenciales de Twitter y usaremos Twitter4j para descargar datos sobre un tema, por ejemplo vamos a buscar los tweets que contengan el hashtag #ebola y los vamos a insertar en Redis. La clave del registro a insertar será el identificador de tweet y su contenido asociado será el JSON completo del tweet.

```
package com.autentia.tutoriales.tweets;

import javax.annotation.PostConstruct;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import twitter4j.Query;
import twitter4j.QueryResult;
import twitter4j.Status;
import twitter4j.Twitter;
import twitter4j.TwitterException;
import twitter4j.TwitterFactory;
import twitter4j.TwitterObjectFactory;

import com.autentia.tutoriales.redis.StringRedisRepository;

@Component
public class TweetIngestor {

    private static final Logger logger = LoggerFactory.getLogger(TweetIngestor.class)

    private final Twitter twitter;

    public static final String TWEET_KEY = "tweet_";

    @Autowired
    private StringRedisRepository redisRepository;

    public TweetIngestor() {
        this.twitter = new TwitterFactory().getInstance();
    }

    @PostConstruct
    public void searchByHashtag() {
        new Thread() {
            @Override
            public void run() {
                try {
                    Query query = new Query("#ebola");

                    int numTweets = 0;
                    long init = System.currentTimeMillis();
                    try {
                        QueryResult result;

                        do {
                            result = twitter.search(query);
                            for (Status status : result.getTweets()) {
                                if (!status.isRetweet()) {
                                    redisRepository.add(TWEET_KEY + String.valueOf(st
                                    numTweets++;
                                }
                            }

                        } while ((query = result.nextQuery()) != null);

                        logger.info(String.format("%s tweets received %s millis", num
                    } catch (TwitterException e) {
                        throw new RuntimeException("Something was wrong retrieving tw
                    }

                } catch (Exception e) {
                    logger.error("Error", e);
                }
            }
        }.start();
    }
}
```

Para arrancar la aplicación lo hacemos con **mvn spring-boot:run**. Pasados unos segundos nos estarán llegando tweets y quedarán almacenados en Redis. Podemos verlos conectando con **redis-cli**. Para ver el contenido de todas las claves almacenadas se utiliza el comando **keys ***. Si queremos ver el contenido de una clave en concreto: **get 'la_clave'**:

```
221) "tweet_528971655981174785"
222) "tweet_528971280960086017"
223) "tweet_528971522149711873"
224) "tweet_528970784778514433"
225) "tweet_528970481345777664"
226) "tweet_528973232922451968"
227) "tweet_528973470953390080"
228) "tweet_528971535571496961"
229) "tweet_528971601178787840"
230) "tweet_528971813255012352"
231) "tweet_528972775059652610"
232) "tweet_528970667405094912"
127.0.0.1:6379> get tweet_528972185910525952
"{\"contributors\":null,\"text\":\"#Yellowstone enzymes helping diagnose #ebola ! http://t.co/lSBJjh1Peg\",\"
geo\":null,\"retweeted\":false,\"in_reply_to_screen_name\":null,\"possibly_sensitive\":false,\"truncated\":fa
lse,\"lang\":\"en\",\"entities\":{\"symbols\":[],\"urls\":[{\"expanded_url\":\"http://yellowstoneinsider.com/
2014/11/02/yellowstone-enzymes-key-ebola-dianosis-tool/\",\"indices\":[47,69],\"display_url\":\"yellowstonein
sider.com/2014/11/02/yel\\u2026\",\"url\":\"http://t.co/lSBJjh1Peg\"}],\"hashtags\":[{\"text\":\"Yellowstone\
",\"indices\":[0,12]},{\"text\":\"ebola\",\"indices\":[38,44]}],\"user_mentions\":[]},\"in_reply_to_status_id
_str\":null,\"id\":528972185910525952,\"source\":\"<a href=\\\"http://twitter.com\\\" rel=\\\"nofollow\\\">Tw
itter Web Client<\\/a>\",\"in_reply_to_user_id_str\":null,\"favorited\":false,\"in_reply_to_status_id\":null,
\"retweet_count\":0,\"created_at\":\"Sun Nov 02 18:09:31 +0000 2014\",\"in_reply_to_user_id\":null,\"favorite
_count\":0,\"id_str\":\"528972185910525952\",\"place\":null,\"user\":{\"location\":\"University of Illinois\"
,\"default_profile\":true,\"profile_background_tile\":false,\"statuses_count\":350,\"lang\":\"en\",\"profile_
link_color\":\"0084B4\",\"profile_banner_url\":\"https://pbs.twimg.com/profile_banners/1467050958/1369836153\
",\"id\":1467050958,\"following\":false,\"protected\":false,\"profile_location\":null,\"favourites_count\":4,
\"profile_text_color\":\"333333\",\"description\":\"Professor of Curmudgeon at the University of Illinois.\",
\"verified\":false,\"contributors_enabled\":false,\"profile_sidebar_border_color\":\"C0DEED\",\"name\":\"Robe
rt Pahre\",\"profile_background_color\":\"C0DEED\",\"created_at\":\"Wed May 29 11:49:14 +0000 2013\",\"is_tra
nslation_enabled\":false,\"default_profile_image\":false,\"followers_count\":70,\"profile_image_url_https\":\
"https://pbs.twimg.com/profile_images/3727072717/738d500db85ad7fd640b18b1169d8720_normal.jpeg\",\"geo_enabled
\":false,\"profile_background_image_url\":\"http://abs.twimg.com/images/themes/theme1/bg.png\",\"profile_back
ground_image_url_https\":\"https://abs.twimg.com/images/themes/theme1/bg.png\",\"follow_request_sent\":false,
\"entities\":{\"description\":{\"urls\":[]},\"url\":{\"urls\":[{\"expanded_url\":\"http://publish.illinois.ed
u/pahre/\",\"indices\":[0,22],\"display_url\":\"publish.illinois.edu/pahre/\",\"url\":\"http://t.co/Y4YDvfv4B
X\"}]}},\"url\":\"http://t.co/Y4YDvfv4BX\",\"utc_offset\":null,\"time_zone\":null,\"notifications\":false,\"p
rofile_use_background_image\":true,\"friends_count\":84,\"profile_sidebar_fill_color\":\"DDEEF6\",\"screen_na
me\":\"RPahre\",\"id_str\":\"1467050958\",\"profile_image_url\":\"http://pbs.twimg.com/profile_images/3727072
717/738d500db85ad7fd640b18b1169d8720_normal.jpeg\",\"listed_count\":2,\"is_translator\":false},\"coordinates\
":null,\"metadata\":{\"result_type\":\"recent\",\"iso_language_code\":\"en\"}}"
127.0.0.1:6379>
```

Si queremos recuperar los tweets de Redis nos podemos crear una clase que a trav&eactue; de nuestro StringRedisRepository podamos pedirle los tweets a partir del patrón común que hemos utilizado (tweet_) llamando al método getAllValuesBy:

```java
package com.autentia.tutoriales.tweets;

import java.util.Set;

import javax.annotation.PostConstruct;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import com.autentia.tutoriales.redis.StringRedisRepository;

@Component
public class TweetsRetriever {

    private static final Logger logger = LoggerFactory.getLogger(TweetsRetriever.clas

    @Autowired
    private StringRedisRepository redisRepository;

    @PostConstruct
    public void retrieve() {
        final Set<String> tweets = redisRepository.getAllValuesBy(TweetIngestor.TWEET

        for (String tweet : tweets) {
            logger.info(tweet);
        }
    }
}
```

7. Referencias

- Código fuente del tutorial
- Documentación oficial: http://redis.io/documentation
- Comandos Redis http://redis.io/commands
- The Little Redis Book en español: http://raulexposito.com/documentos/redis/

8. Conclusiones.

Hemos pasado muy por encima sobre Redis pero cabe destacar su enorme potencial como almacén de datos por su simplifidad, facilidad de uso y versatilidad.

Cada vez hay más librerías que nos dan soporte y una forma de conectar y usar Redis que aún lo hacen más sencillo. Spring Data Redis es una buena muestra de ello por lo que seguro será una buena elección para tus aplicaciones Java si quieres almacenar y consumir datos en Redis.

Espero que te haya sido de ayuda.

Un saludo.

Juan

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Share | 0 0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» **Registrate** y accede a esta y otras ventajas «