mongoDB para Desarrolladores

Practica 6 Creando Microservicios con Dropwizard y MongoDB

Objetivo

- Aprender a utilizar Dropwizard para crear Microservicios interactuando con MongoDB

Procedimiento

1. Inicia el servidor mongoDB mediante el demonio mongod.exe y conéctate al servidor usando mongo CLI mongo.exe

2. Crea la base de datos devmongodb si es que no existe.

3. Crea la colección people si esta no existe en devmongodb para este propósito ejecuta la siguiente sentencia.

```
db.people.insertMany([
 {name:"Mary",gender:"female",size:1.72,weight:54,phone:"+51 2345679", age:25
,email:"mary.smith@gmail.com",company:"AWS",isActive:true,address:[{primary:"100
Boulevard Miami",secondary:"303 St. Geneva Rome"}]},
 {name:"Charles",gender:"male",size:1.86,weight:90,phone:"+86 7345674", age:35
,email:"charles.slate@yahoo.com",company:"Redhat",isActive:true},
 {name:"Danny",gender:"male",size:1.91,weight:102,phone:"+1 8445663", age:25
,email:"danny.lasalle@growing.com",company:"AWS",isActive:false,address:[{primary:"10
2 Bronco Texas",secondary:"404 Borbon Street Lousiana"}]},
 {name:"Richard",gender:"male",size:1.82,weight:83,phone:"+86 2545671", age:35
,email:"richard.jhonson@gmail.com",company:"Open cloud",isActive:true},
 {name:"Yenny",gender:"female",size:1.75,weight:56,phone:"+51 2345459", age:29
,email:"yenny.sullivan@gmail.com",company:"AWS",isActive:false,address:[{primary:"505
Renfer Madrid",secondary:"345 Republica Barcelona"}]},
 {name:"Rob",gender:"male",size:1.79,weight:85,phone:"+51 7145679", age:35
,email:"rob.sax@mshaw.com",company:"Microsoft Inc",isActive:false},
 {name:"Brain",gender:"male",size:1.90,weight:92,phone:"+1 8947679", age:45
,email:"brain.dawner@yahoo.com",company:"AWS",isActive:true} ,
 {name:"Jane",gender:"male",size:1.56,weight:55,phone:"+1 8345663", age:25
,email:"jane.gross@growing.com",company:"MongoDB Inc",isActive:true}
]);
```

4. Crea el proyecto maven con soporte de Dropwizard, ejecuta el siguiente comando:

```
mvn archetype:generate      -DarchetypeGroupId=io.dropwizard.archetypes
                            -DarchetypeArtifactId=java-simple
                            -DarchetypeVersion=1.3.16
                             -DgroupId=com.example.helloworld
                             -DartifactId=HelloworldService
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.1.2:generate (default-cli) > generate-sources @
standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.1.2:generate (default-cli) < generate-sources @
standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.1.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
[INFO] Archetype repository not defined. Using the one from [io.dropwizard.archetypes:java-
simple:2.0.0-rc9] found in catalog remote
[INFO] Using property: groupId = com.example.helloworld
[INFO] Using property: artifactId = HelloworldService
Define value for property 'version' 1.0-SNAPSHOT: :
[INFO] Using property: package = com.example.helloworld
[INFO] Using property: description = null
Define value for property 'name': HelloWorld
[INFO] Using property: shaded = true
```

Confirm properties configuration:

groupId: com.example.helloworld

artifactId: HelloworldService

version: 1.0-SNAPSHOT

package: com.example.helloworld

description: null

name: HelloWorld

shaded: true

Y: : y

[INFO] ----------

[INFO] Using following parameters for creating project from Archetype: java-simple:1.3.16

[INFO] ----------

[INFO] Parameter: groupId, Value: com.example.helloworld

[INFO] Parameter: artifactId, Value: HelloworldService

[INFO] Parameter: version, Value: 1.0-SNAPSHOT

[INFO] Parameter: package, Value: com.example.helloworld

[INFO] Parameter: packageInPathFormat, Value: com/example/helloworld

[INFO] Parameter: version, Value: 1.0-SNAPSHOT

[INFO] Parameter: package, Value: com.example.helloworld

[INFO] Parameter: name, Value: HelloWorld

[INFO] Parameter: groupId, Value: com.example.helloworld

[INFO] Parameter: description, Value: null

[INFO] Parameter: shaded, Value: true

[INFO] Parameter: artifactId, Value: HelloworldService

[INFO] Project created from Archetype in dir: C:\Users\Daddy\workspace\HelloworldService

[INFO] ----------

[INFO] BUILD SUCCESS

[INFO] ----------

[INFO] Total time: 29.237 s

[INFO] Finished at: 2019-10-25T14:47:38-05:00

[INFO] --

C:\Users\Daddy\workspace>

5. Importa el Proyecto y revisa el archivo de configuración del proyecto, archivo pom.xml identifica: La versión de Dropwizard, la versión del compilador java requerido, la clase principal entre otras características del proyecto.

Nota: Si hay alguna alerta, cambia las preferencias del IDE respecto a JRE, cámbialo por JDK. En muchas ocasiones es conveniente definir la variable JAVA_HOME y modificar la variable PATH para indicar el cambio a los binario de la versión de java de trabajo.

6. Modifica la clase HelloWorldConfiguration.java para que contenga lo siguiente:

```
package com.example.helloworld;

import io.dropwizard.Configuration;
import com.fasterxml.jackson.annotation.JsonProperty;
import org.hibernate.validator.constraints.*;
import javax.validation.constraints.*;

public class HelloWorldConfiguration extends Configuration {
       @NotEmpty
    private String template;

    @NotEmpty
    private String defaultName = "Stranger";

    @JsonProperty
    public String getTemplate() {
        return template;
    }

    @JsonProperty
    public void setTemplate(String template) {
        this.template = template;
    }

    @JsonProperty
    public String getDefaultName() {
        return defaultName;
    }

    @JsonProperty
    public void setDefaultName(String name) {
        this.defaultName = name;
```

```
    }
}
```

7. Crea el archivo example.yml, que contenga lo siguiente:

```
template: Hello, %s!
defaultName: Stranger
```

8. Crea en el paquete com.example.helloworld.api la clase Saying que representa el contenido

```
package com.example.helloworld.api;

import com.fasterxml.jackson.annotation.JsonProperty;
import org.hibernate.validator.constraints.Length;

public class Saying {
    private long id;

    @Length(max = 3)
    private String content;

    public Saying() {
        // Jackson deserialization
    }

    public Saying(long id, String content) {
        this.id = id;
        this.content = content;
    }

    @JsonProperty
    public long getId() {
        return id;
    }

    @JsonProperty
    public String getContent() {
        return content;
    }
}
```

9. Crea el recurso HelloWorldResource en el paquete com.example.helloworld.resource con el siguiente código.

```
package com.example.helloworld.resources;
import com.example.helloworld.api.Saying;
```

```java
import com.codahale.metrics.annotation.Timed;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;
import java.util.concurrent.atomic.AtomicLong;
import java.util.Optional;

@Path("/hello-world")
@Produces(MediaType.APPLICATION_JSON)
public class HelloWorldResource {
    private final String template;
    private final String defaultName;
    private final AtomicLong counter;

    public HelloWorldResource(String template, String defaultName) {
        this.template = template;
        this.defaultName = defaultName;
        this.counter = new AtomicLong();
    }

    @GET
    @Timed
    public Saying sayHello(@QueryParam("name") Optional<String> name) {
        final String value = String.format(template, name.orElse(defaultName));
        return new Saying(counter.incrementAndGet(), value);
    }
       }
```

10. En la clase HelloWorldApplication modifica el código para agregar lo siguiente:

```java
@Override
    public void run(HelloWorldConfiguration configuration,
                    Environment environment) {
        final HelloWorldResource resource = new HelloWorldResource(
            configuration.getTemplate(),
            configuration.getDefaultName()
        );
        environment.jersey().register(resource);
    }
```

Nota: Importa las clases necesarias

11. Crea la clase `TemplateHealthCheck` en el paquete `com.example.helloworld.health`

```java
package com.example.helloworld.health;

import com.codahale.metrics.health.HealthCheck;
```

```java
public class TemplateHealthCheck extends HealthCheck {
    private final String template;

    public TemplateHealthCheck(String template) {
        this.template = template;
    }

    @Override
    protected Result check() throws Exception {
        final String saying = String.format(template, "TEST");
        if (!saying.contains("TEST")) {
            return Result.unhealthy("template doesn't include a name");
        }
        return Result.healthy();
    }
}
```

12. Modifica el método run de la clase HelloWorldApplication para que contenga lo siguiente:

```java
@Override
public void run(HelloWorldConfiguration configuration,
                Environment environment) {
    final HelloWorldResource resource = new HelloWorldResource(
        configuration.getTemplate(),
        configuration.getDefaultName()
    );
    final TemplateHealthCheck healthCheck =
        new TemplateHealthCheck(configuration.getTemplate());
    environment.healthChecks().register("template", healthCheck);
    environment.jersey().register(resource);
  }
```

13. Modifica el pom.xml para agregar el tag
    `<addDefaultImplementationEntries>true</addDefaultImplementationEntries>`

```xml
76          <plugin>
77              <artifactId>maven-jar-plugin</artifactId>
78              <version>3.0.2</version>
79              <configuration>
80                  <archive>
81                      <manifest>
82                          <addClasspath>true</addClasspath>
83                          <addDefaultImplementationEntries>true</addDefaultImplementationEntries>
84                          <mainClass>${mainClass}</mainClass>
85                      </manifest>
86                  </archive>
87              </configuration>
88          </plugin>
```

14. Usando maven compila y empaqueta la aplicación

15. Ejecuta la aplicación

Java -jar target/hello-world-0.0.1-SNAPSHOT.jar server example.yml

16. Consulta las siguientes url

    http://localhost:8080/hello-world

    http://localhost:8080/hello-world?name=Successful+Dropwizard+User

    http://localhost:8081/metrics