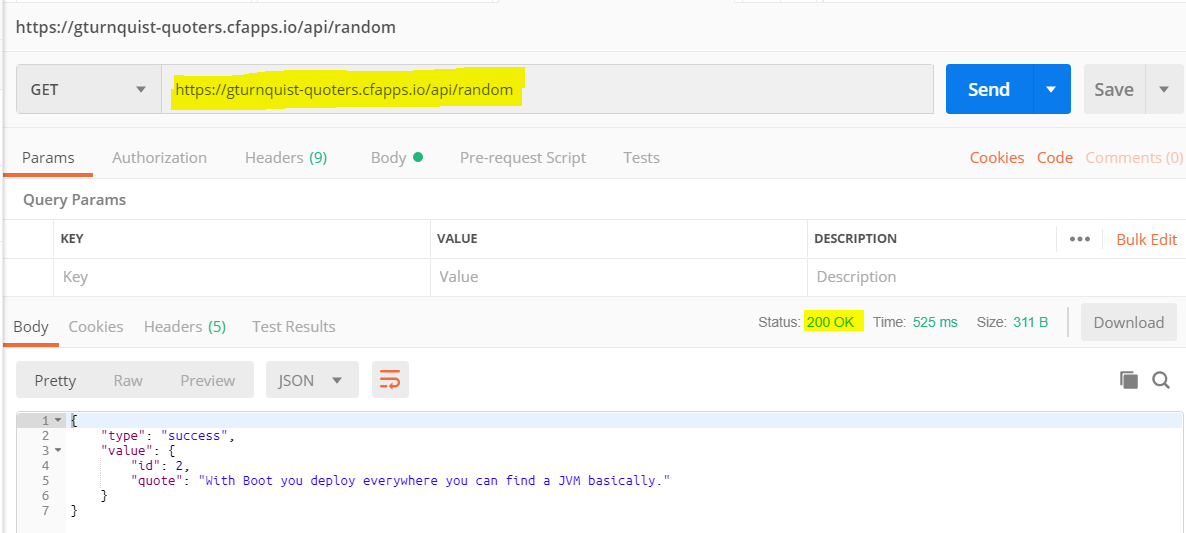
**CONSUMIR UN REST --- Spring.io**



**import com.fasterxml.jackson.annotation.JsonIgnoreProperties;**  
@JsonIgnoreProperties(ignoreUnknown = **true**)  
**public class Quote** {  
  
 **private** String **type**;  
 **private** Value **value**;  
  
 **public Quote**() { }  
 …  
 **@Override** **public** String **toString**() {  
 **return "Quote{"** + **"type='"** + **type** + **'\''** + **", value="** + **value** + **'}'**;  
 }  
}

…

**import com.fasterxml.jackson.annotation.JsonIgnoreProperties;**  
**@JsonIgnoreProperties(ignoreUnknown = true)  
public class Value** {  
  
 **private** Long **id**;  
 **private** String **quote**;  
  
 **public Value**() { }  
  
 **@Override  
 public String toString(){return "Value{" + "id=" + id +", quote='" + quote + '\'' + '}';** }  
}

…

**import** com.example.loggin.com.example.model.Quote;  
**import** org.slf4j.Logger;  
**import** org.slf4j.LoggerFactory;  
**import** org.springframework.web.client.RestTemplate;  
  
**public class App** {  
  
 **private static final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(App.**class**);  
  
 **public static void main**(String[] args) {  
  
 **RestTemplate** restTemplate = **new** RestTemplate();  
 **Quote** forObject = **restTemplate**.**getForObject**(**"https://gturnquist-quoters.cfapps.io/api/random"**, Quote.**class**);  
 ***LOGGER*.info(forObject.toString());** }  
}

**======================= LO MISMO PERO EN REST**

**import** com.example.loggin.com.example.model.Quote;  
**import** org.slf4j.Logger;  
**import** org.slf4j.LoggerFactory;  
**import** org.springframework.boot.CommandLineRunner;  
**import** org.springframework.boot.SpringApplication;  
**import** org.springframework.boot.autoconfigure.SpringBootApplication;  
**import** org.springframework.boot.web.client.RestTemplateBuilder;  
**import** org.springframework.context.annotation.Bean;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
**import** org.springframework.web.client.RestTemplate;  
  
**@RestController  
@SpringBootApplication  
public class LogginApplication {**  
 **private static final Logger *LOGGER* = LoggerFactory.*getLogger*(LogginApplication.class);**  
 **@RequestMapping("/")** **public** String **devuelve**() {

***LOGGER***.info(**"SOMEONE HIT!!!"**);  
 ***LOGGER***.error(**"error"**);  
 ***LOGGER***.warn(**"warn"**);  
 ***LOGGER***.info(**"info"**);

***LOGGER***.debug(**"debug"**); *// no debiera mostrarse* **return "Hello Daniel!"**;  
 }  
  
 **public static void main**(String[] args) {  
 SpringApplication.*run*(LogginApplication.**class**, args);

***LOGGER***.error(**"error"**);  
 ***LOGGER***.warn(**"warn"**);  
 ***LOGGER***.info(**"info"**);  
 ***LOGGER***.debug(**"debug"**);

}  
 **@Bean  
 public RestTemplate restTemplate(RestTemplateBuilder builder) {** **return** builder.build();  
 }  
 **@Bean  
 public CommandLineRunner run(RestTemplate restTemplate) throws Exception {** **return** args -> {  
 Quote quote = restTemplate.getForObject(**"https://gturnquist-quoters.cfapps.io/api/random"**, Quote.**class**);  
 ***LOGGER***.info(quote.toString());  
 };  
 }  
}

15**:**12 **[**main**]** INFO com**.**example**.**loggin**.**LogginApplication **\*\*** Starting LogginApplication on DESKTOP**-**P3FK8PT with PID 3056 **(**C**:**\Users\Daniel\Downloads\loggin\out\production\classes started by Daniel in C**:**\Users\Daniel\Downloads\loggin**)**

15**:**12 **[**main**]** INFO com**.**example**.**loggin**.**LogginApplication **\*\*** No active profile set**,** falling back to **default** profiles**:** **default**

15**:**12 **[**main**]** INFO o**.**s**.**b**.**w**.**e**.**tomcat**.**TomcatWebServer **\*\*** Tomcat initialized with port**(**s**):** 8080 **(**http**)**

15**:**12 **[**main**]** INFO o**.**a**.**coyote**.**http11**.**Http11NioProtocol **\*\*** Initializing ProtocolHandler **[**"http-nio-8080"**]**

15**:**12 **[**main**]** INFO o**.**a**.**catalina**.**core**.**StandardService **\*\*** Starting service **[**Tomcat**]**

15**:**12 **[**main**]** INFO o**.**a**.**catalina**.**core**.**StandardEngine **\*\*** Starting Servlet engine**:** **[**Apache Tomcat**/**9.0.21**]**

15**:**12 **[**main**]** INFO o**.**a**.**c**.**c**.**C**.[**Tomcat**].[**localhost**].[/] \*\* Initializing Spring embedded WebApplicationContext**

15**:**12 **[**main**]** INFO o**.**s**.**web**.**context**.**ContextLoader **\*\*** Root WebApplicationContext**:** initialization completed in 1639 ms

15**:**12 **[**main**]** INFO o**.**s**.**s**.**c**.**ThreadPoolTaskExecutor **\*\*** Initializing ExecutorService 'applicationTaskExecutor'

15**:**12 **[**main**]** INFO o**.**a**.**coyote**.**http11**.**Http11NioProtocol **\*\*** Starting ProtocolHandler **[**"http-nio-8080"**]**

15**:**12 **[**main**]** INFO o**.**s**.**b**.**w**.**e**.**tomcat**.**TomcatWebServer **\*\*** Tomcat started on port**(**s**):** 8080 **(**http**)** with context path ''

15**:**12 **[**main**]** INFO com**.**example**.**loggin**.**LogginApplication **\*\*** Started LogginApplication in 2.623 seconds **(**JVM running **for** 3.228**)**

15**:**12 **[**main**]** INFO com**.**example**.**loggin**.**LogginApplication **\*\*** Quote**{**type**=**'success'**,** value**=**Value**{**id**=**12**,** quote**=**'@springboot with @springframework is pure productivity! Who said in #java one has to write double the code than in other langs? #newFavLib'**}}**

15**:**12 **[**main**]** ERROR com**.**example**.**loggin**.**LogginApplication **\*\*** error

15**:**12 **[**main**]** WARN com**.**example**.**loggin**.**LogginApplication **\*\*** warn

15**:**12 **[**main**]** INFO com**.**example**.**loggin**.**LogginApplication **\*\*** info

15**:**12 **[**http**-**nio**-**8080**-**exec**-**1**]** INFO o**.**a**.**c**.**c**.**C**.[**Tomcat**].[**localhost**].[/] \*\* Initializing Spring DispatcherServlet 'dispatcherServlet'**

15**:**12 **[**http**-**nio**-**8080**-**exec**-**1**]** INFO o**.**s**.**web**.**servlet**.**DispatcherServlet **\*\*** Initializing Servlet 'dispatcherServlet'

15**:**12 **[**http**-**nio**-**8080**-**exec**-**1**]** INFO o**.**s**.**web**.**servlet**.**DispatcherServlet **\*\*** Completed initialization in 7 ms

15**:**12 **[**http**-**nio**-**8080**-**exec**-**1**]** INFO com**.**example**.**loggin**.**LogginApplication **\*\*** SOMEONE HIT**!!!**

15**:**12 **[**http**-**nio**-**8080**-**exec**-**1**]** ERROR com**.**example**.**loggin**.**LogginApplication **\*\*** error

15**:**12 **[**http**-**nio**-**8080**-**exec**-**1**]** WARN com**.**example**.**loggin**.**LogginApplication **\*\*** warn

15**:**12 **[**http**-**nio**-**8080**-**exec**-**1**]** INFO com**.**example**.**loggin**.**LogginApplication **\*\*** info

===

**logback.xml**

*<?***xml version = "1.0" encoding = "UTF-8"***?>*<**configuration**>  
 <**appender name = "STDOUT" class = "ch.qos.logback.core.ConsoleAppender"**>  
 <**encoder**>  
 <**pattern**>%d{HH:mm} [%t] %-5level %logger{36} \*\* %msg%n</**pattern**>  
 </**encoder**>  
 </**appender**>  
 <**root level = "INFO"**>  
 <**appender-ref ref = "STDOUT"**/>  
 </**root**>  
</**configuration**>

===

**applition.properties**

#logging.level.root=info

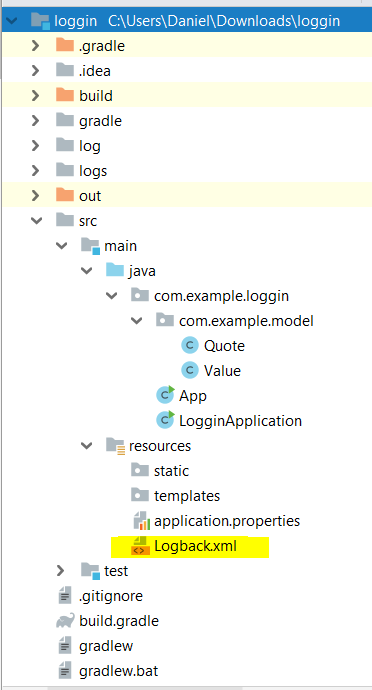
#logging.pattern.console=%d{HH:mm} [%t] %-5level %logger{36} \*\* %msg%n

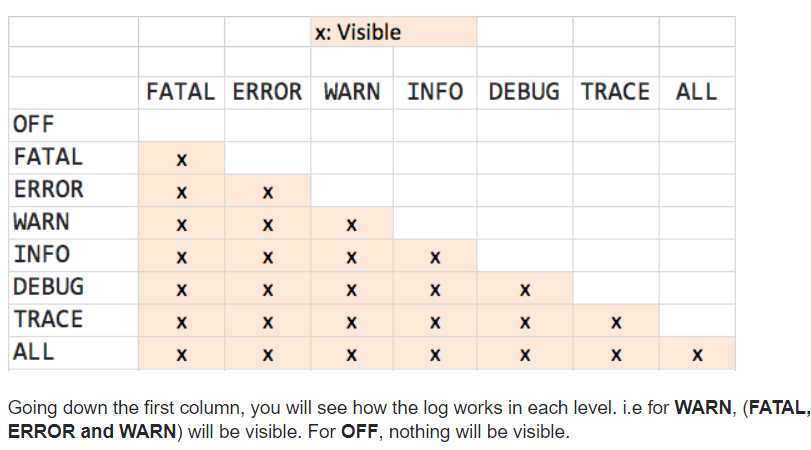
#logging.pattern.console.com.example.loggin.com.example.model.Quote=%d{HH:mm} [%t] %-5level %logger{36} \*\* %msg%nProtocol \*\* Initializing ProtocolHandler ["http-nio-8080"]

15:12 [main] INFO o.a.catalina.core.StandardService \*\* Starting service [Tomcat]

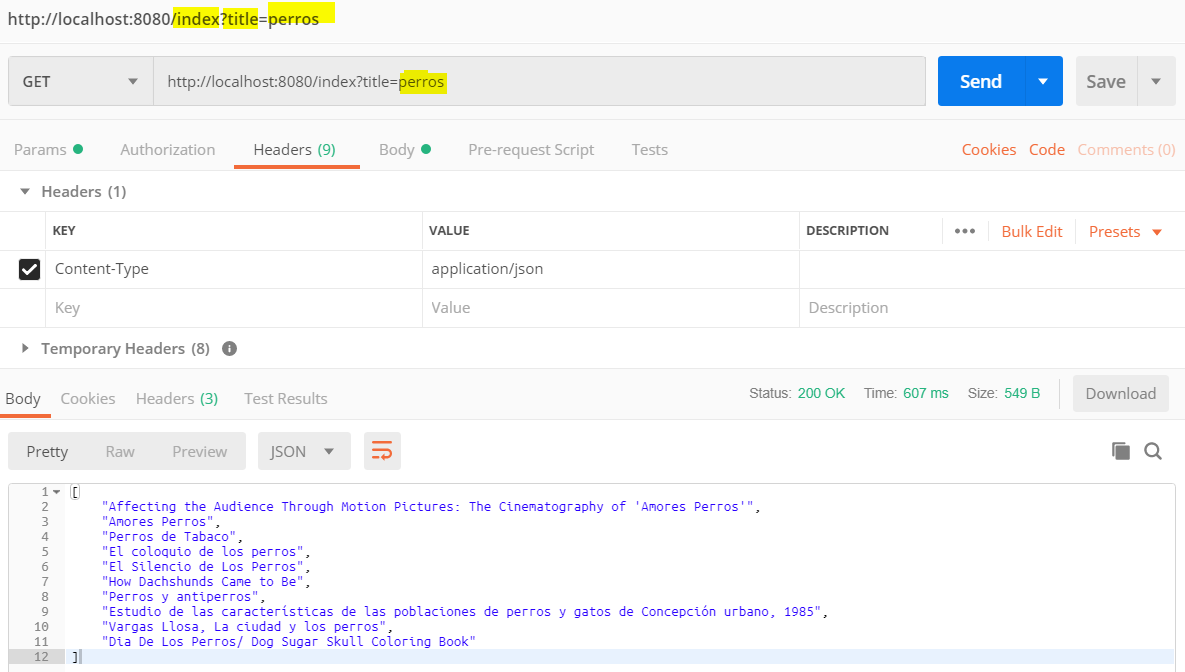
===

**build.gradle**

buildscript {  
 repositories {  
 mavenCentral()  
 }  
 dependencies {  
 classpath(**"org.springframework.boot:spring-boot-gradle-plugin:2.1.4.RELEASE"**)  
 }  
}  
  
*// Binary plugin  
/\*  
You apply plugins by their plugin id, which is a globally unique identifier, or name,   
for plugins. Core Gradle plugins are special in that they provide short names, such   
as 'java' for the core JavaPlugin  
\*/*plugins {  
 id **'java'**}  
  
*// Script plugin*apply **plugin**: **'java'**apply **plugin**: **'eclipse'**apply **plugin**: **'idea'**apply **plugin**: **'org.springframework.boot'**apply **plugin**: **'io.spring.dependency-management'**group **'com.example'**version **'1.0-SNAPSHOT'**sourceCompatibility = 1.8  
targetCompatibility = 1.8  
  
repositories {  
 mavenCentral()  
}  
  
dependencies {  
 compile(**"org.springframework.boot:spring-boot-starter"**)  
 compile(**"org.springframework:spring-web"**)  
 compile(**"com.fasterxml.jackson.core:jackson-databind"**)  
 testCompile(**"junit:junit"**)  
 testCompile(**"org.assertj:assertj-core:3.11.1"**)  
 *// https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-test* testCompile **group**: **'org.springframework.boot'**, **name**: **'spring-boot-starter-test'**, **version**: **'2.1.4.RELEASE'** *// https://mvnrepository.com/artifact/org.slf4j/slf4j-simple* testCompile **group**: **'org.slf4j'**, **name**: **'slf4j-simple'**, **version**: **'1.6.1'**}  
  
*/\*  
Script plugins are additional build scripts that further configure the build and usually   
implement a declarative approach to manipulating the build. They are typically used within   
a build although they can be externalized and accessed from a remote location. Binary plugins   
are classes that implement the Plugin interface and adopt a programmatic approach to manipulating   
the build. Binary plugins can reside within a build script, within the project hierarchy or externally   
in a plugin jar.\*/*



**================================= CON NODES CUANDO NO CONOCEMOS LO QUE HAY ABAJO**



{

"kind": "books#volumes",

"totalItems": 479,

"**items**": [

{

"kind": "books#volume",

"id": "1TMLRbolbZ8C",

"etag": "r1CWxrGq+H4",

"selfLink": "https://www.googleapis.com/books/v1/volumes/1TMLRbolbZ8C",

"**volumeInfo**": {

"**title**": "Affecting the Audience Through Motion Pictures: The Cinematography of 'Amores Perros'",

"authors": [

"Susanne Schwarz"

],

"publisher": "GRIN Verlag",

"publishedDate": "2009-10",

"description": …

**============================ SOLO CONTROLLER**

**import** com.fasterxml.jackson.databind.JsonNode;  
**import** com.fasterxml.jackson.databind.node.ArrayNode;  
**import** com.fasterxml.jackson.databind.node.ObjectNode;  
**import** org.slf4j.Logger;  
**import** org.slf4j.LoggerFactory;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RequestParam;  
**import** org.springframework.web.bind.annotation.RestController;  
**import** org.springframework.web.client.RestTemplate;  
**import** java.util.ArrayList;  
**import** java.util.Collection;  
  
**@RestController  
public class** Application {  
  
 *// Yo creo que cuando uno es feliz puede ser cualquier cosa* Logger **logger** = LoggerFactory.*getLogger*(Application.**class**);  
  
 **@RequestMapping("/index")** **public** Collection<String> getVolumnes(@RequestParam String title) {  
  
 RestTemplate restTemplate = **new** RestTemplate();  
  
 **logger**.info(**"Se buscan libros con el titulo de: {}"**, title);  
ObjectNode data = restTemplate.getForObject(**"https://www.googleapis.com/books/v1/volumes?q=intitle:"** + title,

ObjectNode.**class**);  
 Collection<String> bookTitles = **new** ArrayList<>();  
  
 ArrayNode items = (ArrayNode) data.get(**"items"**);  
  
 **for** (**int** i = 0; i < items.size(); i++) {  
 JsonNode jsonNode = items.get(i);  
 String s = jsonNode.get(**"volumeInfo"**).get(**"title"**).asText();  
 bookTitles.add(s);  
 **logger**.info(s);  
 }

**return bookTitles**;  
 }  
}

…

21:21 [main] INFO o.a.coyote.http11.Http11NioProtocol \*\* Starting ProtocolHandler ["http-nio-8080"]

21:21 [main] INFO o.s.b.w.e.tomcat.TomcatWebServer \*\* Tomcat started on port(s): 8080 (http) with context path ''

21:21 [main] INFO com.example.loggin.LogginApplication \*\* Started LogginApplication in 2.508 seconds (JVM running for 3.108)

21:21 [http-nio-8080-exec-1] INFO o.a.c.c.C.[Tomcat].[localhost].[/] \*\* Initializing Spring DispatcherServlet 'dispatcherServlet'

21:21 [http-nio-8080-exec-1] INFO o.s.web.servlet.DispatcherServlet \*\* Initializing Servlet 'dispatcherServlet'

21:21 [http-nio-8080-exec-1] INFO o.s.web.servlet.DispatcherServlet \*\* Completed initialization in 8 ms

21:21 [http-nio-8080-exec-1] INFO c.e.loggin.controller.Application \*\* Se buscan libros con el titulo de: perros

21:21 [http-nio-8080-exec-1] INFO c.e.loggin.controller.Application \*\* Affecting the Audience Through Motion Pictures: The Cinematography of 'Amores Perros'

21:21 [http-nio-8080-exec-1] INFO c.e.loggin.controller.Application \*\* Amores Perros

21:21 [http-nio-8080-exec-1] INFO c.e.loggin.controller.Application \*\* Perros de Tabaco

21:21 [http-nio-8080-exec-1] INFO c.e.loggin.controller.Application \*\* El Silencio de Los Perros

21:21 [http-nio-8080-exec-1] INFO c.e.loggin.controller.Application \*\* How Dachshunds Came to Be

21:21 [http-nio-8080-exec-1] INFO c.e.loggin.controller.Application \*\* Perros y antiperros

21:21 [http-nio-8080-exec-1] INFO c.e.loggin.controller.Application \*\* Vargas Llosa, La ciudad y los perros