# Consuming a RESTful Web Service

Fetch a REST resource

With project setup complete, you can create a simple application that consumes a RESTful service.

A RESTful service has been stood up at [http://gturnquist-quoters.cfapps.io/api/random](https://gturnquist-quoters.cfapps.io/api/random). It randomly fetches quotes about Spring Boot and returns them as a JSON document.

If you request that URL through your web browser or curl, you’ll receive a JSON document that looks something like this:

{

type: "success",

value: {

id: 10,

quote: "Really loving Spring Boot, makes stand alone Spring apps easy."

}

}

A more useful way to consume a REST web service is programmatically. To help you with that task, Spring provides a convenient template class called [RestTemplate](http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/client/RestTemplate.html). RestTemplate makes interacting with most RESTful services a one-line incantation. And it can even bind that data to custom domain types.

First, create a domain class to contain the data that you need.

src/main/java/hello/Quote.java

package hello;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@JsonIgnoreProperties(ignoreUnknown = true)

public class Quote {

private String type;

private Value value;

public Quote() {

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public Value getValue() {

return value;

}

public void setValue(Value value) {

this.value = value;

}

@Override

public String toString() {

return "Quote{" +

"type='" + type + '\'' +

", value=" + value +

'}';

}

}

As you can see, this is a simple Java class with a handful of properties and matching getter methods. It’s annotated with @JsonIgnoreProperties from the Jackson JSON processing library to indicate that any properties not bound in this type should be ignored.

In order for you to directly bind your data to your custom types, you need to specify the variable name exact same as the key in the JSON Document returned from the API. In case your variable name and key in JSON doc are not matching, you need to use @JsonPropertyannotation to specify the exact key of JSON document.

An additional class is needed to embed the inner quotation itself.

src/main/java/hello/Value.java

package hello;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@JsonIgnoreProperties(ignoreUnknown = true)

public class Value {

private Long id;

private String quote;

public Value() {

}

public Long getId() {

return this.id;

}

public String getQuote() {

return this.quote;

}

public void setId(Long id) {

this.id = id;

}

public void setQuote(String quote) {

this.quote = quote;

}

@Override

public String toString() {

return "Value{" +

"id=" + id +

", quote='" + quote + '\'' +

'}';

}

}

This uses the same annotations but simply maps onto other data fields.

src/main/java/hello/Application.java

package hello;

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.client.RestTemplate;

@SpringBootApplication

public class Application {

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

public static void main(String args[]) {

SpringApplication.run(Application.class);

}

@Bean

public RestTemplate restTemplate(RestTemplateBuilder builder) {

return builder.build();

}

@Bean

public CommandLineRunner run(RestTemplate restTemplate) throws Exception {

return args -> {

Quote quote = restTemplate.getForObject(

"http://gturnquist-quoters.cfapps.io/api/random", Quote.class);

log.info(quote.toString());

};

}

}

The RestTemplateBuilder is injected by Spring, and if you use it to create a RestTemplatethen you will benefit from all the autoconfiguration that happens in Spring Boot with message converters and request factories. We also extract the RestTemplate into a @Bean to make it easier to test (it can be mocked more easily that way).

### Build an executable JAR

If you are using Maven, you can run the application using ./mvnw spring-boot:run. Or you can build the JAR file with ./mvnw clean package. Then you can run the JAR file:

java -jar target/gs-consuming-rest-0.1.0.jar