Spring Boot Consume RESTful API

Description: Spring Boot is an open source, micro service-based Java web framework. The Spring Boot framework creates a fully production-ready environment that is completely configurable using its prebuilt code within its own codebase.

Project: Build a program that consumes a RESTful API at “https://quoters.apps.pcfone.io/api/random”. The demo will use Spring’s RestTemplate to retrieve a random Spring Boot quotation at the above website. This project is based on Consuming a RESTful Web Service and just expands on the screenshots and step by step instructions.

Technology: This project uses the following technology:

Integrated Development Environment (IDE):

Spring Tool Suite 4 (Version: 4.15.0.RELEASE)

Java Development Kit (JDK):

Oracle's JDK 8 (1.8)

This project is incomplete. After build the program it did not run without errors. Additional research and updates are required to completed this document.

Table of Contents

Glossary of Terminology

For a list of key terms and definitions used throughout this and various Spring Boot demo documents see the document titled “Appendix 01 Glossary”.

Generate Spring Boot Download

Follow the instructions in the document title “Appendix 02 Spring Initializr” to generate a spring boot download for this project.

When the document talks about the items in the “**Project Metadata**” use the values shown below:

> “**Group**” use “**com.example**”
> “**Artifact**” use “**consuming-rest**”
> “**Name**” use “**consuming-rest**”
> “**Package name**” use “**com.example.consuming-rest**”

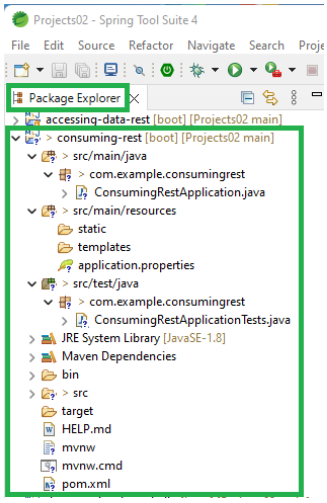
For other items in the “**Project Metadata**” and dependency use the defaults. Follow the instructions to extract the files from the zip file into the Sprint Tool Suite 4 workspace.

Import the Spring Boot Download

Open the Spring Tool Suite 4 IDE.

Import the project: “rest-service”

Follow the instructions in the document title “Appendix 03 Import Spring Tool Suite Project” and import the “**consuming-rest**” project that was created with Spring Initializr. After importing the project, it should look like the following using the IDE “Package Explorer”.



Project consuming-rest Discussion

The project builds an application using Spring's RestTemplate to retrieve a random quotation at https://quoters.apps.pcfone.io/api/random.

Main Class: ConsumingRestApplication

The ConsumingRestApplication class created when using the Spring Initializr is used with modification as the last section before testing. This class contains the "`public static void main(String[] args) {}`" method that starts the application listening when the internal web server is started.

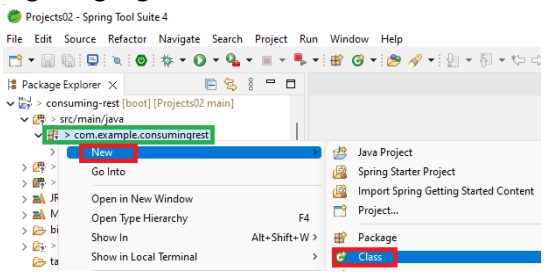
Create Model Classes

A Model class is a generic class that defines an object, something tangible.

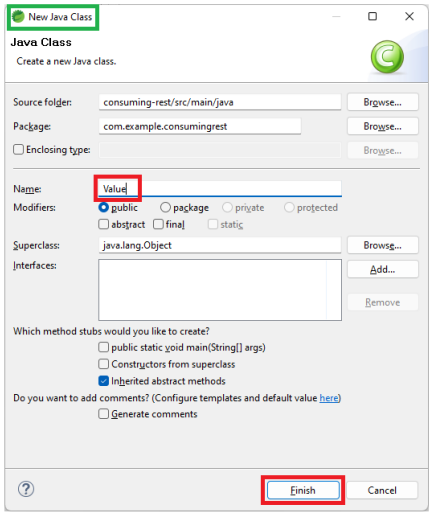
Create Model Class: Value

This class is used to embed the inner quotation from the API being consumed. This class is built first to eliminate errors in the Quote class.

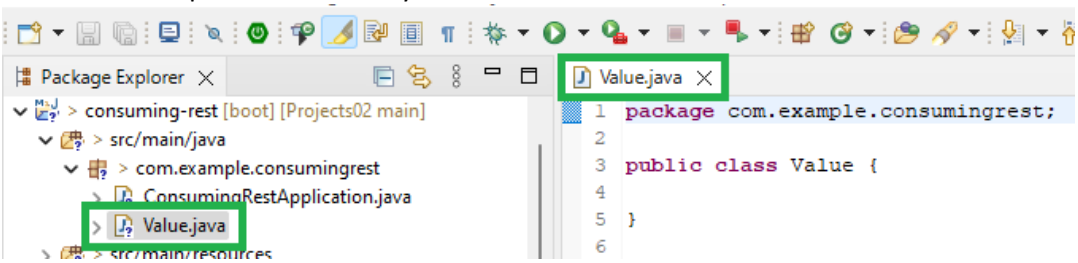
Right click on service package "com.example.consumingrest"
Right highlight "New" ➔ select "Class"



Enter class "Name:" "Value"
Click "Finish"



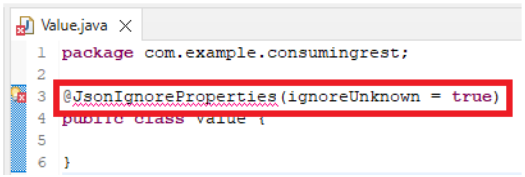
The new class opens automatically in an edit window.



Update model class: Value

Start modifying the class by adding the following annotation which, mark a property or list of properties to be ignored. The properties that are declared in this annotation will not be mapped to the JSON content.

```
@JsonIgnoreProperties(ignoreUnknown = true)
```

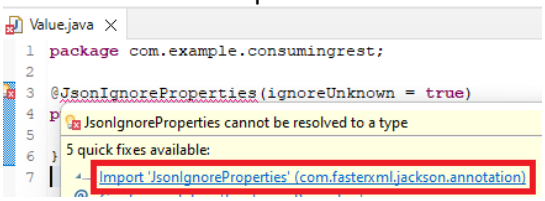


Correct the import error:

Use the IDE Wizard to help correct the error.

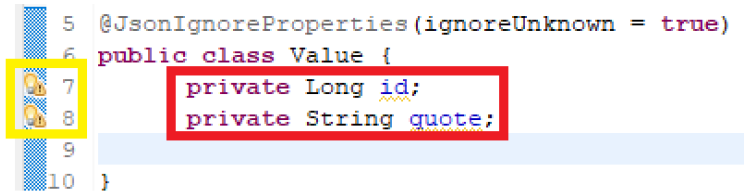
Hover the mouse over the error.

Select the import link shown below.



Add the two private class variables.

```
    private Long id;
    private String quote;
```



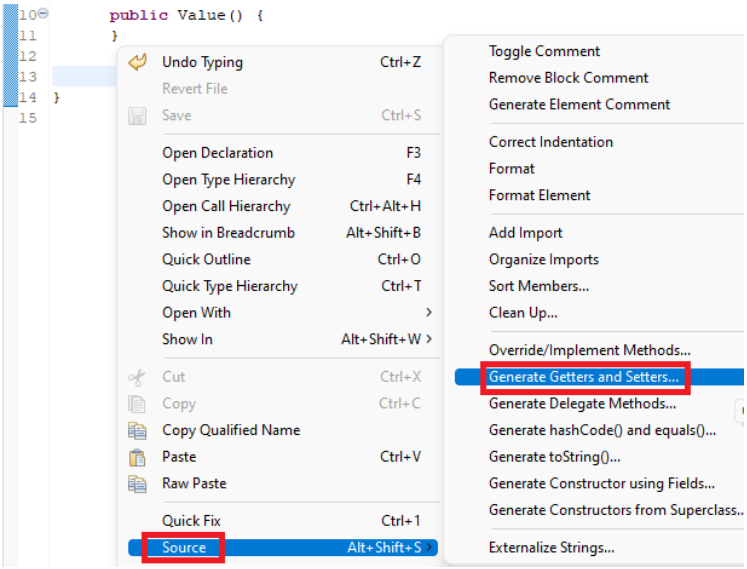
Notice the warning to the left. Basically, they indicate that the class variables are not being used. The warnings will go away once the variables are referenced in the code.

Add the no argument constructor.

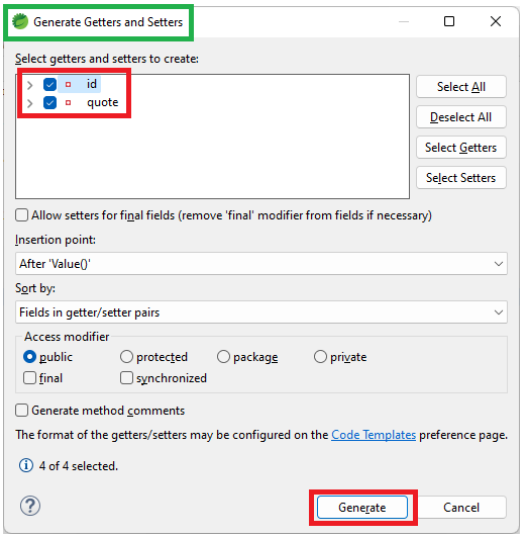
```
    public Value() {
    }
```

```
5  @JsonIgnoreProperties(ignoreUnknown = true)
6  public class Value {
7        private Long id;
8        private String quote;
9
10       public Value() {
11       }
12
13
14 }
```

Use the IDE Wizard to generate the getters and setters for the class variables. The source Wizard is a handy way to save time writing code by letting the Wizard generate the code for us.
Right click inside the class at line 13.
Highlight “Source” ➔ click “Generate Getters and Setters…”



Select both class variables.
Click “Generate”.



The getters and setters are generated by the Wizard.

```
13      public Long getId() {
14          return id;
15      }
16
17      public void setId(Long id) {
18          this.id = id;
19      }
20
21      public String getQuote() {
22          return quote;
23      }
24
25      public void setQuote(String quote) {
26          this.quote = quote;
27      }
```

Add code to override the toString() method inherited from the Object class, which all classes are derived from. The @Override annotation does as its namesake meaning that it performs this method and not the toString() method from its parent the Object class.

```
@Override
public String toString() {
  return "Value{" +
       "id=" + id +
       ", quote='" + quote + '\'' +
       '}';
}
```

```
29      @Override
30      public String toString() {
31          return "Value{" + "id=" + id + ", quote='" + quote + '\'' + '}';
32      }
```

```
29      @Override
30  overrides java.lang.Object.toString
```

The completed class.

Value.java

```
1  package com.example.consumingrest;
2
3  import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
4
5  @JsonIgnoreProperties(ignoreUnknown = true)
6  public class Value {
7      private Long id;
8      private String quote;
9
10     public Value() {
11     }
12
13     public Long getId() {
14         return id;
15     }
16
17     public void setId(Long id) {
18         this.id = id;
19     }
20
21     public String getQuote() {
22         return quote;
23     }
24
25     public void setQuote(String quote) {
26         this.quote = quote;
27     }
28
29     @Override
30     public String toString() {
31         return "Value{" + "id=" + id + ", quote='" + quote + '\'' + '}';
32     }
33
34 }
```

Copy and paste source code for class: Value

As an alternative you can copy the code below and replace all code in the class if you used the same package name.

```
package com.example.consumingrest;

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 + '\'' +
        '}';
  }
}
```

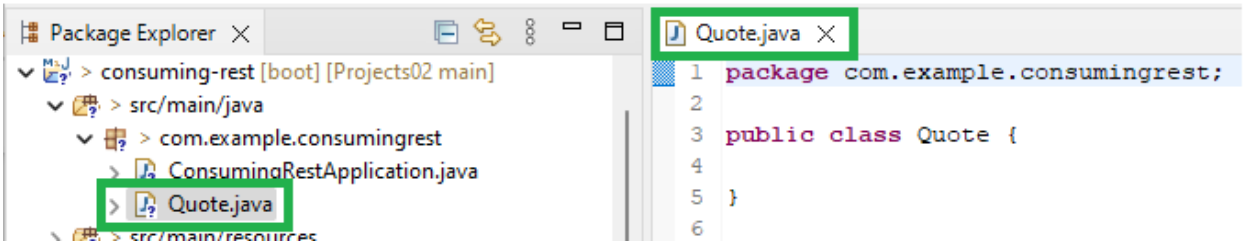
Create Model Class: Quote

"This simple Java class has a handful of properties and matching getter methods. It is annotated with @JsonIgnoreProperties from the Jackson JSON processing library to indicate that any properties not bound in this type should be ignored.

To directly bind your data to your custom types, you need to specify the variable name to be exactly the same as the key in the JSON document returned from the API. In case your variable name and key in JSON doc do not match, you can use @JsonProperty annotation to specify the exact key of the JSON document. (This example matches each variable name to a JSON key, so you do not need that annotation here.)" -- Consuming a RESTful Web Service

Right click on service package "com.example.consumingrest"
Right highlight "New" ➔ select "Class"
Enter class "Name:" "Quote"
Click "Finish"

The new class opens automatically in an edit window.



Update model class: Quote

Items of interest where covered when creating the Value class. For the Quote class simple copy the code below and replace all code generated by the Wizard that generated class shell:

```
package com.example.consumingrest;

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 +
        '}';
  }
}
```

The completed class. Since we created the Value class first the reference in the Quote class did not shown an error.

Quote.java

```
1  package com.example.consumingrest;
2
3  import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
4
5  @JsonIgnoreProperties(ignoreUnknown = true)
6  public class Quote {
7
8    private String type;
9    private Value value;
10
11   public Quote() {
12   }
13
14   public String getType() {
15     return type;
16   }
17
18   public void setType(String type) {
19     this.type = type;
20   }
21
22   public Value getValue() {
23     return value;
24   }
25
26   public void setValue(Value value) {
27     this.value = value;
28   }
29
30   @Override
31   public String toString() {
32     return "Quote{" +
33         "type='" + type + '\'' +
34         ", value=" + value +
35         '}';
36   }
37 }
```

Update Main Class: ConsumingRestApplication

Like mentioned earlier the ConsumingRestApplication class created when using the Spring Initializr is used with modification. This section walks through the modification of this class.

> "Now you need to add a few other things to the `ConsumingRestApplication` class to get it to show quotations from our RESTful source. You need to add:
>
> - A logger, to send output to the log (the console, in this example).
> - A `RestTemplate`, which uses the Jackson JSON processing library to process the incoming data.
> - A `CommandLineRunner` that runs the `RestTemplate` (and, consequently, fetches our quotation) on startup."
>
> -- Consuming a RESTful Web Service

The class Wizard generated the following basic main class.

ConsumingRestApplication.java

```
1  package com.example.consumingrest;
2
3  import org.springframework.boot.SpringApplication;
5
6  @SpringBootApplication
7  public class ConsumingRestApplication {
8
9      public static void main(String[] args) {
10         SpringApplication.run(ConsumingRestApplication.class, args);
11     }
12
13 }
14
```

Add a class logging variable.

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

```
6  @SpringBootApplication
7  public class ConsumingRestApplication {
8      private static final Logger log = LoggerFactory.getLogger(ConsumingRestApplication.class);
9
```

Correct the import errors:

Use the IDE Wizard to help correct the errors.

Hover the mouse over the error.

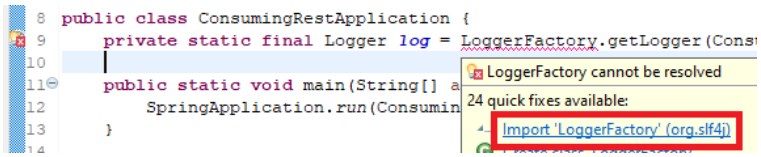
Select the import link shown below.

```
7  public class ConsumingRestApplication {
8      private static final Logger log = LoggerFactory.
9
10     public static void ma
11         SpringApplication
12     }
13
14 }
15
```

Logger cannot be resolved to a type

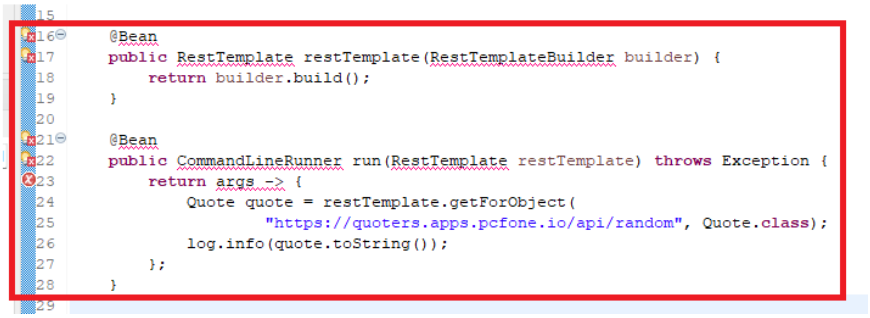
23 quick fixes available:

- Import 'Logger' (ch.qos.logback.cla
- Import 'Logger' (java.util.logging)
- Import 'Logger' (org.apache.loggin
- Import 'Logger' (org.slf4j)



Add two methods with @Bean annotation.
The annotation is applied on a method to specify that it returns a bean. The bean is managed by Spring context. The method is responsible for creating the instance.

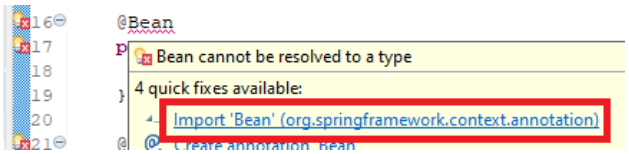
```
@Bean
public RestTemplate restTemplate(RestTemplateBuilder builder) {
        return builder.build();
}

@Bean
public CommandLineRunner run(RestTemplate restTemplate) throws Exception {
        return args -> {
                Quote quote = restTemplate.getForObject(
                                "https://quoters.apps.pcfone.io/api/random", Quote.class);
                log.info(quote.toString());
        };
}
```

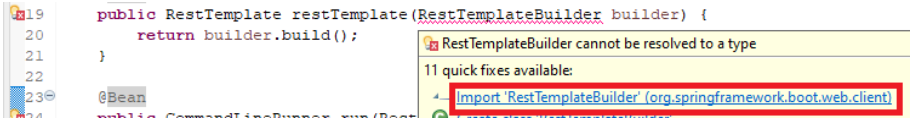


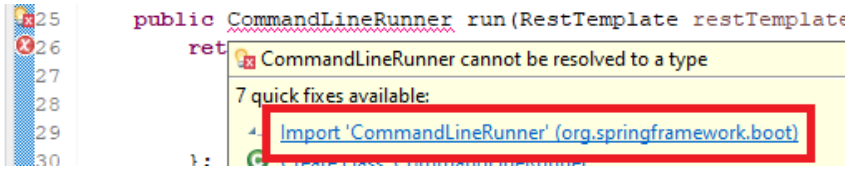
Use the IDE Wizard to help correct the errors.

Hover the mouse over the error.

Select the import link shown below.







The completed class code.

ConsumingRestApplication.java

```
1  package com.example.consumingrest;
2
3  import org.slf4j.Logger;
4  import org.slf4j.LoggerFactory;
5  import org.springframework.boot.CommandLineRunner;
6  import org.springframework.boot.SpringApplication;
7  import org.springframework.boot.autoconfigure.SpringBootApplication;
8  import org.springframework.boot.web.client.RestTemplateBuilder;
9  import org.springframework.context.annotation.Bean;
10 import org.springframework.web.client.RestTemplate;
11
12 @SpringBootApplication
13 public class ConsumingRestApplication {
14     private static final Logger log = LoggerFactory.getLogger(ConsumingRestApplication.class);
15
16     public static void main(String[] args) {
17         SpringApplication.run(ConsumingRestApplication.class, args);
18     }
19
20     @Bean
21     public RestTemplate restTemplate(RestTemplateBuilder builder) {
22         return builder.build();
23     }
24
25     @Bean
26     public CommandLineRunner run(RestTemplate restTemplate) throws Exception {
27         return args -> {
28             Quote quote = restTemplate.getForObject(
29                     "https://quoters.apps.pcfone.io/api/random", Quote.class);
30             log.info(quote.toString());
31         };
32     }
33
34 }
```

Copy and paste source code for class: ConsumingRestApplication

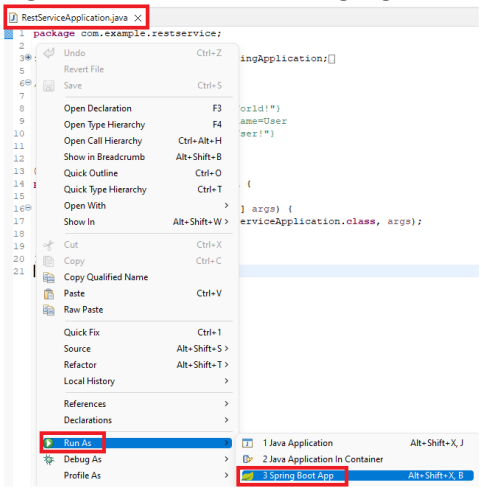
As an alternative you can copy the code below and replace all code in the class if you used the same package name.

```
package com.example.consumingrest;

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 ConsumingRestApplication {

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

          public static void main(String[] args) {
                    SpringApplication.run(ConsumingRestApplication.class, args);
          }

          @Bean
          public RestTemplate restTemplate(RestTemplateBuilder builder) {
                    return builder.build();
          }

          @Bean
          public CommandLineRunner run(RestTemplate restTemplate) throws Exception {
                    return args -> {
                              Quote quote = restTemplate.getForObject(
                                                  "https://quoters.apps.pcfone.io/api/random", Quote.class);
                              log.info(quote.toString());
                    };
          }
}
```

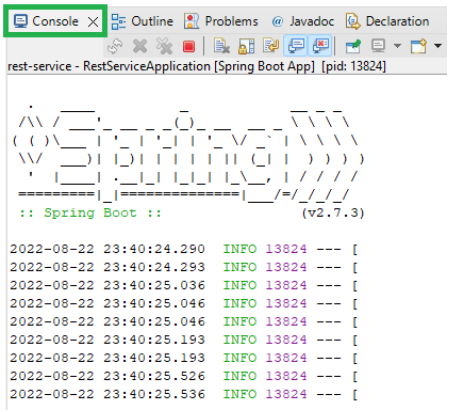
Quick Project Test

To do a quick test of the project use the Spring Boot bundled server. Later “Appendix 04 Run Spring Initializr Project” can be used to use a command line server version and to build an executable jar.

Open class RestServiceApplication.
Right click inside the class highlight “Run As” ➔ “3 Spring Boot App”



The Spring Boot server status is shown in the “Console” tab.

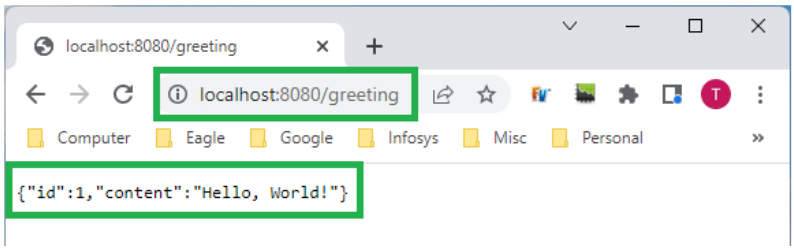


Test the Application’s URLs

Use the following URL in a browser.

```
http://localhost:8080/greeting
```

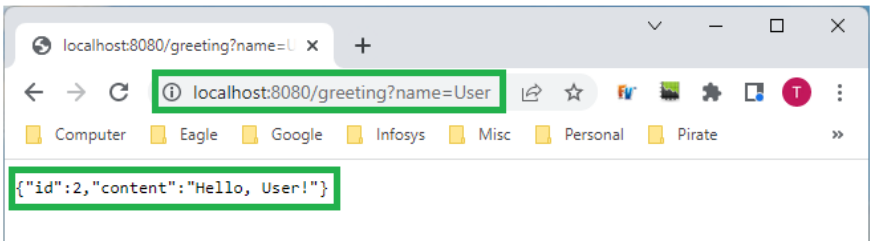
Expected results.



Use the next URL in a browser.

```
http://localhost:8080/greeting
```

Expected results.



Build and Run the Project

This project introduces using an executable Java Archive (JAR) file. The "Appendix 04 Run Spring Initializr Project" document section title "Executable JAR File Lifecycle" should be followed to learn the new concept. At a minimum follow section "Executable JAR File Lifecycle" in the Appendix 04 Run Spring Initializr Project" document.

Using the executable JAR file that is built for this project, the URLs in the previous section "Test the Application's URLs" should be used and the results should be the same.