



JAVA SPRING BOOT



Czym jest Spring Boot?



- Spring Boot to framework oparty na Javie, który służy do tworzenia samodzielnych aplikacji, które są łatwe do zbudowania i uruchomienia. Spring Boot to rozszerzenie Springa.

Czym jest Spring Boot?



- Spring Boot to framework oparty na Javie, który służy do tworzenia samodzielnych aplikacji, które są łatwe do zbudowania i uruchomienia. Spring Boot to rozszerzenie Springa.
- Spring Boot oferuje:

Czym jest Spring Boot?



- Spring Boot to framework oparty na Javie, który służy do tworzenia samodzielnych aplikacji, które są łatwe do zbudowania i uruchomienia. Spring Boot to rozszerzenie Springa.
- Spring Boot oferuje:
 - Łatwiejsze uruchamianie aplikacji

Czym jest Spring Boot?



- Spring Boot to framework oparty na Javie, który służy do tworzenia samodzielnych aplikacji, które są łatwe do zbudowania i uruchomienia. Spring Boot to rozszerzenie Springa.
- Spring Boot oferuje:
 - Łatwiejsze uruchamianie aplikacji
 - Automatyczną konfigurację

Czym jest Spring Boot?

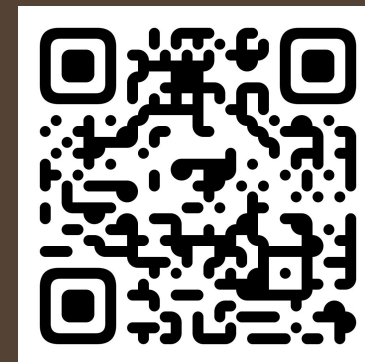


- Spring Boot to framework oparty na Javie, który służy do tworzenia samodzielnych aplikacji, które są łatwe do zbudowania i uruchomienia. Spring Boot to rozszerzenie Springa.
- Spring Boot oferuje:
 - Łatwiejsze uruchamianie aplikacji
 - Automatyczną konfigurację
 - Szybsze tworzenie aplikacji

Czym jest Spring Boot?



- <https://start.spring.io/>

A screenshot of the Spring Initializr web application interface. The interface is dark-themed and contains several sections for configuring a new project. At the top left is the "spring initializr" logo. Below it, the "Project" section has radio buttons for "Gradle - Groovy" (selected), "Gradle - Kotlin", and "Maven". The "Language" section has radio buttons for "Java" (selected), "Kotlin", and "Groovy". The "Spring Boot" section has radio buttons for "3.0.2 (SNAPSHOT)", "3.0.1" (selected), "2.7.8 (SNAPSHOT)", and "2.7.7". The "Project Metadata" section includes input fields for "Group" (com.example), "Artifact" (demo), "Name" (demo), and "Description" (Demo project for Spring Boot). The "Package name" field contains com.example.demo. The "Packaging" section has radio buttons for "Jar" (selected) and "War". At the bottom, there are three buttons: "GENERATE CTRL + G", "EXPLORE CTRL + SPACE", and "SHARE...". A "Dependencies" section on the right is currently empty, showing "No dependency selected".

Jak narodziła się koncepcja Spring Boota?



```
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

def __init__(self, *args, **kwargs):
    self.file = None
    self.fingerprints = self()
    self.logdupes = True
    self.debug = debug
    self.logger = logging.getLogger(__name__)
    if path:
        self.file = open(os.path.join(path, 'requests.log'),
                        'a')
        self.file.seek(0)
        self.fingerprints.update(self)

    @classmethod
    def from_settings(cls, settings):
        debug = settings.getbool('supersite.debug')
        return cls(job_dir(settings), debug)

    def request_seen(self, request):
```


Jak narodziła się koncepcja Spring Boota?



- Sam Spring ma pewne... Niedogodności:

Jak narodziła się koncepcja Spring Boota?



- Sam Spring ma pewne... Niedogodności:
 - szeroko bazuje na konfiguracji

Jak narodziła się koncepcja Spring Boota?



- Sam Spring ma pewne... Niedogodności:
 - szeroko bazuje na konfiguracji
 - kompatybilne biblioteki i ich konfiguracje

Jak narodziła się koncepcja Spring Boota?



- Sam Spring ma pewne... Niedogodności:
 - szeroko bazuje na konfiguracji
 - kompatybilne biblioteki i ich konfiguracje



Jak narodziła się koncepcja Spring Boota?



- Sam Spring ma pewne... Niedogodności:
 - szeroko bazuje na konfiguracji
 - kompatybilne biblioteki i ich konfiguracje
 - wymagany serwer

Jak narodziła się koncepcja Spring Boota?




- Sam Spring ma pewne... Niedogodności:
 - szeroko bazuje na konfiguracji
 - kompatybilne biblioteki i ich konfiguracje
 - wymagany serwer
 - boilerplate code


Jak narodziła się koncepcja Spring Boota?



- Sam Spring ma pewne... Niedogodności:
 - szeroko bazuje na konfiguracji
 - kompatybilne biblioteki i ich konfiguracje
 - wymagany serwer
 - boilerplate code

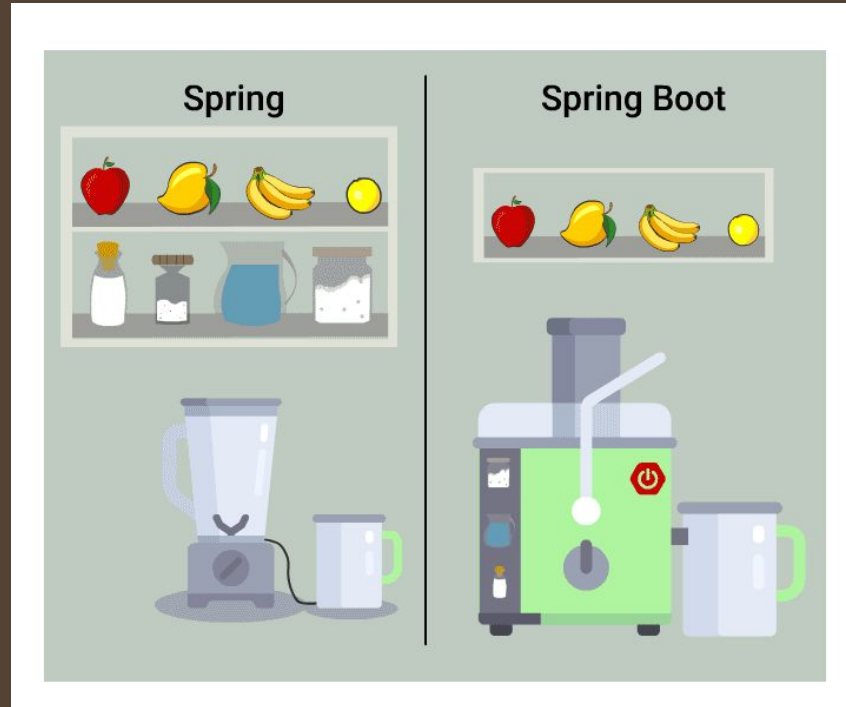


```
1 class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println("Hello, World!");
4     }
5 }
```



```
1 print('Hello, World!')
```

Spring vs Spring Boot



<https://devrant.com>

Jak narodziła się koncepcja Spring Boota?



| Spring | Spring Boot |
|--|---|
| Konieczność ręcznego tworzenia konfiguracji | Konfiguracje tworzone są automatycznie (ale mogą być edytowane) |
| Pozwala tworzyć aplikacje oparte na mikroservisach | Pozwala tworzyć aplikacje stand-alone |
| Wymaga zewnętrznego serwera | Posiada wbudowany i skonfigurowany serwer (Tomcat lub Jetty) |
| Wymaga dużej ilości boilerplate code | Znacząco redukuje ilość potrzebnego kodu |
| Wymaga ręcznego zarządzania bibliotekami | Odpowiednie biblioteki są dołączane automatycznie |

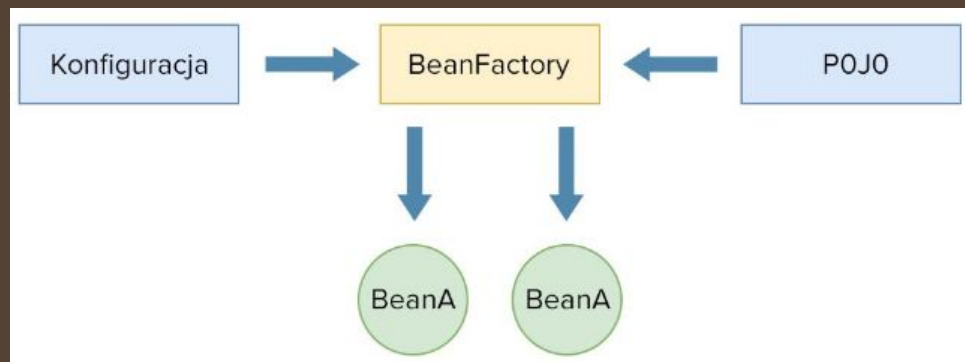
Techniczna strona Spring Boot



Obiekt zarządzany - Bean



- Context and Dependency Injection
- Główne zalety
- Sposoby zdefiniowania w kodzie
 - Adnotacje
 - Instancje w metodach klasy konfiguracyjnej
 - XML
- Kiedy stosować Beany

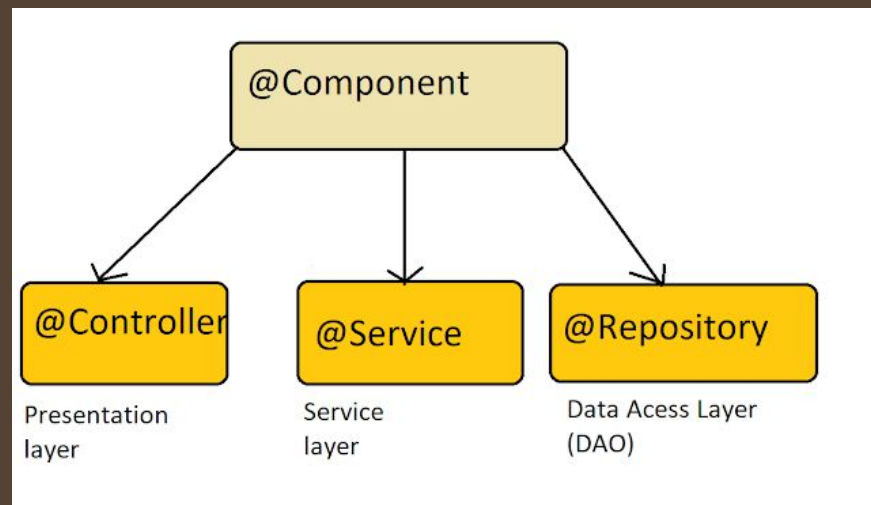


Schemat tworzenia Beanów w Spring

Spring Boot - Adnotacje



- Najpopularniejsze adnotacje
 - @Component
 - @Repository
 - @Service
 - @Controler/@RestController



<https://www.bdabek.pl/wp-content/uploads/2020/12/stereotypes.png>

Mechanizm wstrzykiwania zależności



- Sposoby wstrzykiwania zależności
 - Konstruktor
 - Pole
 - Właściwości / metody

```
public class Car {  
    | usage  
    private CarService carService;  
  
    @Autowired  
    public void setCarService(CarService carService){  
        this.carService = carService;  
    }  
}
```

Wstrzykiwanie przez właściwości / metody

```
public class Car {  
  
    | usage  
    private CarService carService;  
  
    @Autowired  
    public Car(CarService carService) {  
        this.carService = carService;  
    }  
}
```

Wstrzykiwanie przez konstruktor

```
public class Car {  
    @Autowired  
    private CarService carService;  
}
```

Wstrzykiwanie przez pole

Konfiguracja i profilowanie



- Sposoby zarządzania konfiguracją
 - Nadpisanie istniejącej konfiguracji
 - Definiowanie własnych zmiennych
- Wczytywanie konfiguracji
 - @Value
- Profilowanie
 - application.properties
 - application-prod.properties
 - application-dev.properties

The screenshot shows a dark-themed web page titled "Common Application Properties". On the left is a table of contents with 16 items, where "1. Core Properties" is highlighted. The main content area contains an introductory paragraph, a "Tip" section with a lightbulb icon, and a "Note" section with an information icon. The "Tip" section discusses conversion mechanisms and value formatting. The "Note" section states that property contributions can come from additional JAR files on the classpath. At the bottom of the main content area, the heading "1. Core Properties" is visible.

<https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.htm>

Przykładowy projekt “Hello World”



Spring Initializr



spring initializr

Project
 Gradle - Groovy
 Gradle - Kotlin Maven

Language
 Java Kotlin Groovy

Spring Boot
 3.0.2 (SNAPSHOT) 3.0.1 2.7.8 (SNAPSHOT) 2.7.7

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging Jar War

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

<https://start.spring.io>

Project: Maven

Language: Java

Spring Boot: 2.2.8

Packaging: JAR

Java: 8

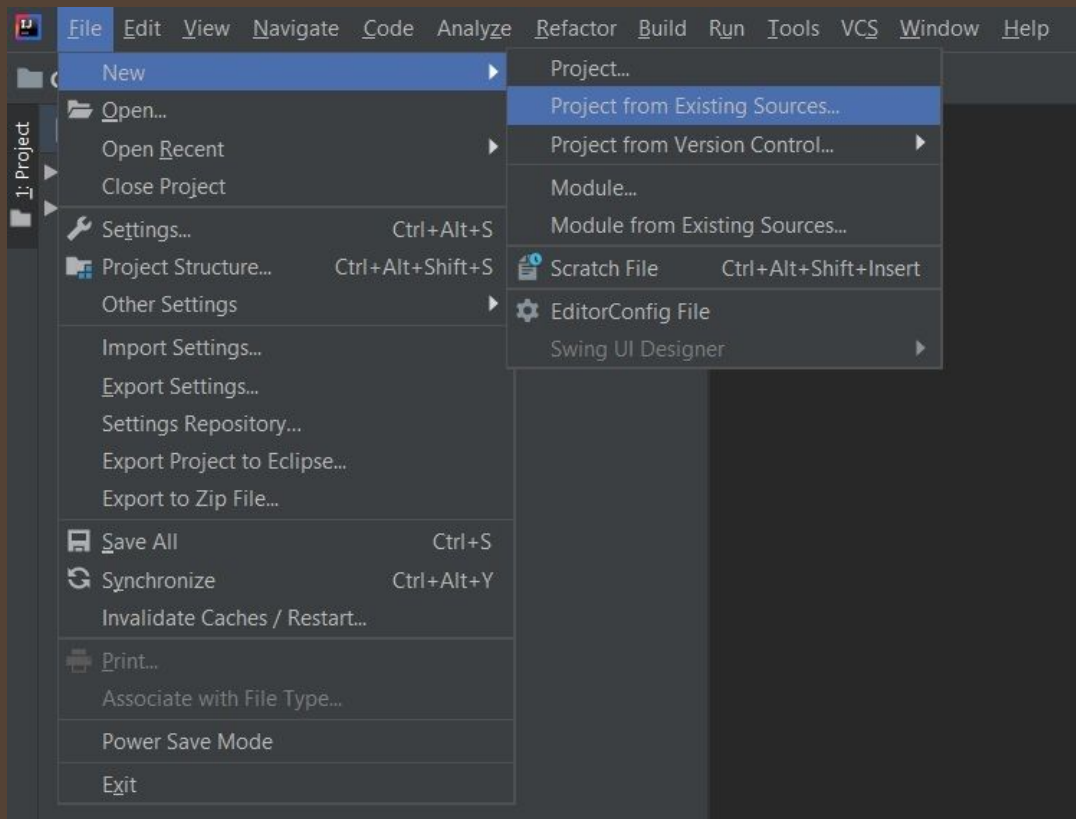
Dependencies: Spring
Web

GENERATE

EXPLORE

SHARE...

Otworzenie projektu w IDE



Wypakuj pobrany plik ZIP.
W IntelliJ wybierz File > New >
Project from Existing Sources...

Dodanie pliku z głównymi funkcjami



```
6 @SpringBootApplication
7
8 // Main class
9 // Implementing CommandLineRunner interface
10 public class SpringBootApplication implements CommandLineRunner
11 {
12     // Method 1
13     // run() method for springBootApplication to execute
14     @Override
15     public void run(String args[]) throws Exception
16     {
17         // Print statement when method is called
18         System.out.println("Hello world!");
19     }
20
21     // Method 2
22     // Main driver method
23     public static void main(String[] args)
24     {
25         // Calling run() method to execute
26         // SpringBootApplication by
27
28         // invoking run() inside main() method
29         SpringApplication.run(SpringBootApplication.class, args);
30     }
31 }
```

Plik src > main > java >
com.example.demo >
SpringBootApplication.java

Test uruchomienia



```
Console Actuator
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...
Java HotSpot(TM) 64-Bit Server VM warning: Options -Xverify:none and -noverify were deprecated in JDK 13 and will likely be removed in a future release.

  ____  _
 / ___|| | | |
 \___ \| |_| |
  ___) | | | |
 |___) | |_| |
      |_|_|_|

:: Spring Boot ::      (v3.0.1)

2022-12-29T17:11:54.659+01:00 INFO 15240 --- [main] c.example.demo.SpringBootApplication : Starting SpringBootApplication using Java 17.0.1 with PID 15240 (C:\Users\Arkadiusz\Downloads\demo\target\classes st
2022-12-29T17:11:54.673+01:00 INFO 15240 --- [main] c.example.demo.SpringBootApplication : No active profile set, falling back to 1 default profile: "default"
2022-12-29T17:11:55.746+01:00 INFO 15240 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-12-29T17:11:55.760+01:00 INFO 15240 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-12-29T17:11:55.760+01:00 INFO 15240 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.4]
2022-12-29T17:11:55.867+01:00 INFO 15240 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-12-29T17:11:55.868+01:00 INFO 15240 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1089 ms
2022-12-29T17:11:56.255+01:00 INFO 15240 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-12-29T17:11:56.265+01:00 INFO 15240 --- [main] c.example.demo.SpringBootApplication : Started SpringBootApplication in 2.308 seconds (process running for 3.578)

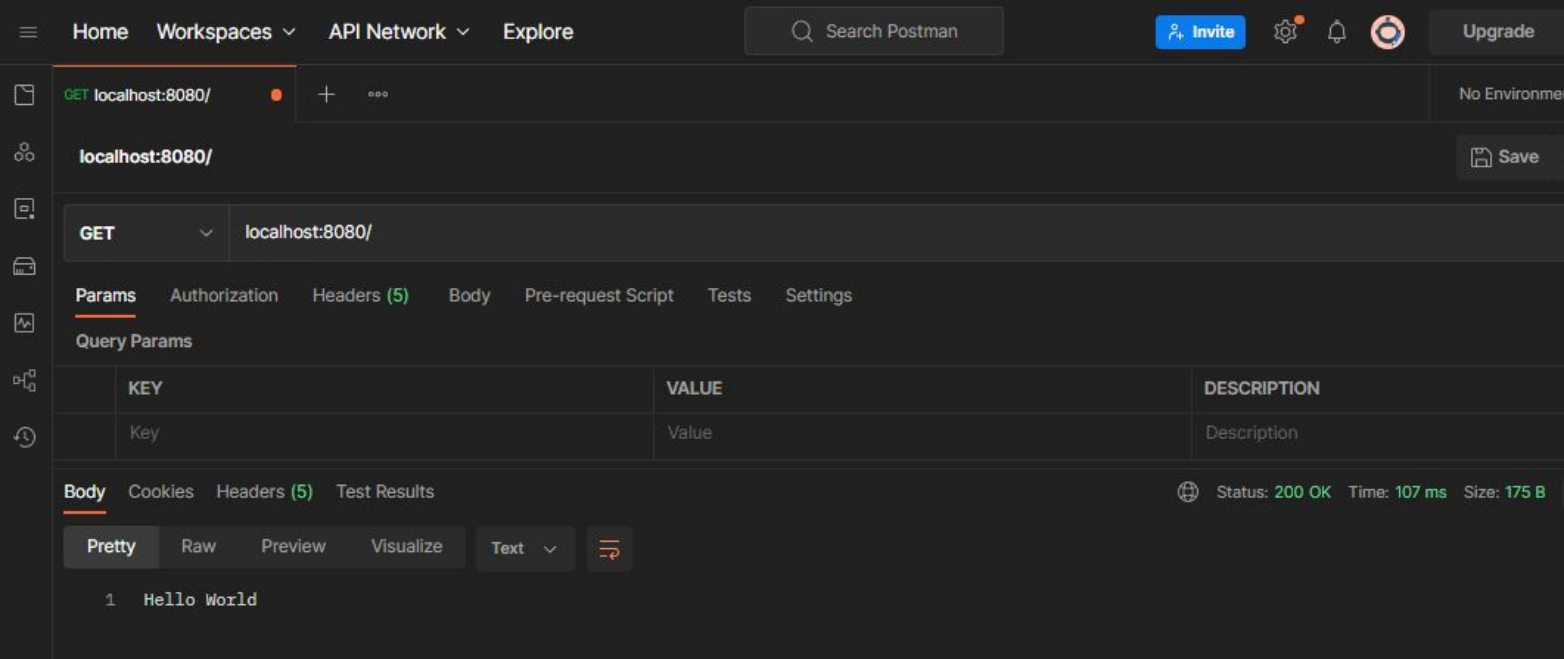
Hello world!
```

Hello World w REST API



```
SpringBootApplication.java x controller.java x
1 package com.example.demo;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RestController
7 public class controller {
8     @GetMapping("/")
9     String returnHelloWorld(){
10         return "Hello World";
11     }
12 }
```

Plik src > main > java > com.example.demo > controller.java



The screenshot shows the Postman interface for a GET request to localhost:8080. The request is configured with the following parameters:

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

The response is displayed in the Body tab, showing a status of 200 OK, a time of 107 ms, and a size of 175 B. The response content is:

```
1 Hello World
```

Zapytanie GET do localhost:8080 i odpowiedź aplikacji

Zapytanie



```
Console Actuator
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...
Java HotSpot(TM) 64-Bit Server VM warning: Options -Xverify:none and -noverify were deprecated in JDK 13 and will likely be removed in a future release.

  ____
 /\  / ___'  _  (C)  _  _  \ \ \ \
 ( ) \__  | '  | '  | '  \  /  \ \ \ \
 \V ___| |_) | | | | | | | | | | | |
 '  |___| |__| | | | | | | | | | | |
 =====|_|=====|_/_/=_/_/_/

:: Spring Boot ::                (v3.0.1)

2022-12-29T17:23:29.318+01:00 INFO 31552 --- [main] c.example.demo.SpringBootApplication : Starting SpringBootApplication using Java 17.0.1 with PID 31552 (C:\Users\Ar
2022-12-29T17:23:29.324+01:00 INFO 31552 --- [main] c.example.demo.SpringBootApplication : No active profile set, falling back to 1 default profile: "default"
2022-12-29T17:23:30.364+01:00 INFO 31552 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-12-29T17:23:30.373+01:00 INFO 31552 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-12-29T17:23:30.373+01:00 INFO 31552 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.4]
2022-12-29T17:23:30.485+01:00 INFO 31552 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-12-29T17:23:30.486+01:00 INFO 31552 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1083 ms
2022-12-29T17:23:30.833+01:00 INFO 31552 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-12-29T17:23:30.843+01:00 INFO 31552 --- [main] c.example.demo.SpringBootApplication : Started SpringBootApplication in 2.154 seconds (process running for 3.209)
Hello world!
2022-12-29T17:24:28.338+01:00 INFO 31552 --- [nio-8080-exec-4] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-12-29T17:24:28.338+01:00 INFO 31552 --- [nio-8080-exec-4] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-12-29T17:24:28.339+01:00 INFO 31552 --- [nio-8080-exec-4] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
```

Tworzenie testów



```
1 package com.example.demo;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.test.context.SpringBootTest;
6 import org.springframework.boot.test.web.client.TestRestTemplate;
7 import org.springframework.http.ResponseEntity;
8 import static org.assertj.core.api.AssertionsForClassTypes.assertThat;
9
10
11 @SpringBootTest(classes = SpringBootTestApplication.class, webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
12 class controllerTest {
13
14     @Autowired
15     private TestRestTemplate template;
16
17     @Test
18     void returnHelloWorld() throws Exception {
19         ResponseEntity<String> response = template.getForEntity( url: "/", String.class);
20         assertThat(response.getBody()).isEqualTo("Hello World!");
21     }
22 }
23
24
```

Test > java > com.example.demo > controllerTest.java

Wynik przeprowadzonego testu



```
Hello world!  
2022-12-29T18:36:35.791+01:00 INFO 5880 --- [o-auto-1-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/] : Ini  
2022-12-29T18:36:35.791+01:00 INFO 5880 --- [o-auto-1-exec-2] o.s.web.servlet.DispatcherServlet : Ini  
2022-12-29T18:36:35.793+01:00 INFO 5880 --- [o-auto-1-exec-2] o.s.web.servlet.DispatcherServlet : Com
```

```
org.opentest4j.AssertionFailedError:
```

```
expected: "Hello World!"
```

```
but was: "Hello World"
```

```
Expected : "Hello World!" ←
```

```
Actual   : "Hello World" ←
```

```
<Click to see difference>
```

```
☐ <3 internal lines>
```

```
at java.base/java.lang.reflect.Constructor.newInstanceWithCaller(Constructor.java:499)
```

```
☐ at com.example.demo.controllerTest.returnHelloWorld(controllerTest.java:23) <31 internal lines>
```

```
☐ at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
```

```
☐ at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <28 internal lines>
```

```
Process finished with exit code -1
```




Koniec

Finis coronat opus