

# Język Kotlin na platformie Spring

Programowanie aplikacji internetowych



Packt

Miloš Vasić

Helion 

Tytuł oryginału: Building Applications with Spring 5 and Kotlin

Tłumaczenie: Andrzej Watrak

ISBN: 978-83-283-5183-7

Copyright © Packt Publishing 2018. First published in the English language under the title 'Building Applications with Spring 5 and Kotlin – (9781788394802)'

Polish edition copyright © 2019 by Helion SA  
All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Helion SA dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Helion SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion SA  
ul. Kościuszki 1c, 44-100 Gliwice  
tel. 32 231 22 19, 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Dodatkowe materiały do książki można znaleźć pod adresem: <ftp://ftp.helion.pl/przyklady/jkotsp.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/jkotsp>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

# Spis treści

|  |           |
|--|-----------|
| <b>O autorze</b>                                     | <b>7</b>  |
| <b>O recenzencie</b>                                 | <b>8</b>  |
| <b>Przedmowa</b>                                     | <b>9</b>  |
| <b>Rozdział 1. Przygotowanie</b>                     | <b>13</b> |
| <b>Jaka jest Twoja misja?</b>                        | <b>13</b> |
| <b>Dzielenie kodu na osobne jednostki</b>            | <b>14</b> |
| Opis jednostek                                       | 14        |
| <b>Planowanie pracy</b>                              | <b>15</b> |
| <b>Przygotowanie środowiska roboczego</b>            | <b>16</b> |
| Instalacja narzędzia Git                             | 16        |
| Kompilacja kodu źródłowego narzędzia Git             | 18        |
| Instalacja pakietu JDK                               | 19        |
| Instalacja środowiska programistycznego              | 20        |
| Pierwsze uruchomienie środowiska IntelliJ IDEA       | 21        |
| Instalacja platformy Spring 5                        | 23        |
| Instalacja programu Postman                          | 28        |
| <b>Utworzenie repozytorium Git</b>                   | <b>29</b> |
| <b>Podsumowanie</b>                                  | <b>31</b> |
| <b>Rozdział 2. Pierwsze kroki z platformą Spring</b> | <b>33</b> |
| <b>Czym jest platforma Spring?</b>                   | <b>33</b> |
| Jakie funkcjonalności oferuje platforma Spring?      | 34        |
| Jak jest obsługiwany język Kotlin?                   | 38        |
| <b>Tworzenie projektu Spring</b>                     | <b>38</b> |
| Czym jest narzędzie Spring Initializr?               | 39        |
| Utworzenie projektu                                  | 39        |
| <b>Tworzenie projektu w środowisku IntelliJ IDEA</b> | <b>48</b> |
| <b>Podsumowanie</b>                                  | <b>50</b> |

|   |            |
|---|------------|
| <b>Rozdział 3. Twoja pierwsza usługa Spring REST w języku Kotlin</b>  | <b>53</b>  |
| <b>Zdefiniowanie zależności w projekcie</b>                           | <b>54</b>  |
| <b>Utworzenie klasy kontrolera</b>                                    | <b>55</b>  |
| Co jeszcze można uzyskać?   | 56         |
| <b>Utworzenie klasy danych</b>  | <b>57</b>  |
| <b>Obsługa innych rodzajów zapytań</b>                                | <b>58</b>  |
| <b>Uruchomienie aplikacji</b>   | <b>65</b>  |
| Ścieżki udostępniane przez bibliotekę Actuator                        | 71         |
| <b>Dodanie komponentu Service</b>                                     | <b>73</b>  |
| <b>Podsumowanie</b>   | <b>78</b>  |
| <b>Rozdział 4. Interfejs Spring Data JPA i baza danych MySQL</b>      | <b>79</b>  |
| <b>Wprowadzenie do interfejsu Spring Data JPA</b>                     | <b>79</b>  |
| Co oferuje biblioteka Spring Data?                                    | 80         |
| Jakie moduły Spring Data będą potrzebne?                              | 80         |
| Interfejs Spring Data JPA   | 80         |
| <b>Instalacja bazy MySQL</b>  | <b>80</b>  |
| Instalacja bazy MySQL w systemie macOS                                | 81         |
| Instalacja bazy MySQL w systemie Windows                              | 83         |
| Instalacja bazy MySQL w systemie Linux                                | 84         |
| Utworzenie schematu bazy danych                                       | 86         |
| Dodanie nowych zależności   | 88         |
| <b>Implementacja operacji CRUD</b>                                    | <b>89</b>  |
| Dodawanie danych  | 95         |
| Modyfikacja danych  | 95         |
| Odczytywanie danych   | 97         |
| Usuwanie danych   | 98         |
| Więcej o modyfikowaniu danych   | 98         |
| Obiekty DTO   | 100        |
| <b>Tworzenie zapytań SQL</b>  | <b>106</b> |
| Nazwane zapytania SQL   | 108        |
| <b>Podsumowanie</b>   | <b>109</b> |
| <b>Rozdział 5. Zabezpieczanie aplikacji za pomocą Spring Security</b> | <b>111</b> |
| <b>Wprowadzenie do platformy Spring Security</b>                      | <b>111</b> |
| <b>Definiowanie ról użytkowników</b>                                  | <b>112</b> |
| Implementacja klas reprezentujących role                              | 112        |
| <b>Definiowanie obiektów DTO dla użytkowników</b>                     | <b>119</b> |
| <b>Uwierzytelnianie i autoryzowanie użytkowników</b>                  | <b>123</b> |
| <b>Co jeszcze oferuje platforma Spring Security?</b>                  | <b>129</b> |
| <b>Podsumowanie</b>   | <b>130</b> |
| <b>Rozdział 6. Platforma Spring Cloud</b>                             | <b>131</b> |
| <b>Architektura SOA</b>   | <b>131</b> |
| <b>Architektura mikrousługowa</b>                                     | <b>132</b> |
| <b>Mikrousługi w platformie Spring Cloud</b>                          | <b>133</b> |
| Platforma Spring Cloud w praktyce                                     | 134        |
| Serwer konfiguracji   | 135        |

|   |            |
|---|------------|
| Wykrywanie serwerów                                   | 137        |
| Brama   | 140        |
| Modyfikacja interfejsu API                            | 143        |
| <b>Zabezpieczanie usług Spring Cloud</b>              | <b>148</b> |
| <b>Podsumowanie</b>                                   | <b>158</b> |
| <b>Rozdział 7. Projekt Reactor</b>                    | <b>159</b> |
| <b>Dlaczego należy stosować projekt Reactor?</b>      | <b>159</b> |
| <b>Co to jest projekt Reactor?</b>                    | <b>160</b> |
| <b>Korzystanie z projektu Reactor</b>                 | <b>160</b> |
| <b>Podsumowanie</b>                                   | <b>168</b> |
| <b>Rozdział 8. Praktyki programistyczne</b>           | <b>171</b> |
| <b>Krytyczna ocena praktyk programistycznych</b>      | <b>171</b> |
| Wstrzykiwane zależności                               | 171        |
| Otwartość klas  | 172        |
| Mutowalne zmienne                                     | 172        |
| Wielowątkowość  | 172        |
| Poprawność danych                                     | 173        |
| Zakres testów   | 173        |
| Konfiguracja XML                                      | 173        |
| <b>Dobre praktyki w tworzeniu kodu</b>                | <b>173</b> |
| Nie wstrzykuj zbyt wiele                              | 173        |
| Ograniczaj widoczność                                 | 173        |
| Problem wielowątkowości                               | 174        |
| Weryfikacja danych                                    | 174        |
| <b>Podsumowanie</b>                                   | <b>175</b> |
| <b>Rozdział 9. Testy</b>                              | <b>177</b> |
| <b>Dlaczego testy są tak ważne w programowaniu?</b>   | <b>177</b> |
| Co trzeba testować?                                   | 178        |
| Najczęściej stosowane praktyki i metodyki testowania  | 178        |
| <b>Przygotowanie projektu</b>                         | <b>179</b> |
| <b>Pierwszy test w języku Kotlin</b>                  | <b>183</b> |
| <b>Uruchamianie testów w środowisku IntelliJ IDEA</b> | <b>189</b> |
| <b>Testowanie aplikacji Spring REST</b>               | <b>191</b> |
| <b>Uruchamianie zestawu testów</b>                    | <b>198</b> |
| <b>Podsumowanie</b>                                   | <b>199</b> |
| <b>Rozdział 10. Wdrażanie aplikacji</b>               | <b>201</b> |
| <b>Co będziemy wdrażać?</b>                           | <b>201</b> |
| <b>Opcje wdrożeniowe</b>                              | <b>208</b> |
| Wdrożenie aplikacji na serwerze Tomcat                | 208        |
| Wdrożenie aplikacji na serwerze Java EE               | 210        |
| <b>Wdrożenie aplikacji w chmurze AWS</b>              | <b>214</b> |
| <b>Podsumowanie</b>                                   | <b>233</b> |
| <b>Skorowidz</b>                                      | <b>349</b> |



# Pierwsze kroki z platformą Spring

W poprzednim rozdziale skonfigurowałeś środowisko programistyczne. Teraz jesteś gotów do utworzenia swojego pierwszego projektu w języku Kotlin opartego na platformie Spring. Zanim to jednak zrobisz, musisz poznać podstawy tej platformy i dowiedzieć się, co można za jej pomocą uzyskać. Wtedy będziesz mógł wystartować z projektem. Przygotuj się!

W tym rozdziale opisane są następujące zagadnienia:

- czym jest platforma Spring,
- najważniejsze funkcjonalności platformy Spring,
- obsługa języka Kotlin w platformie Spring,
- generowanie projektów,
- tworzenie projektu w środowisku IntelliJ IDEA.

---

## Czym jest platforma Spring?

Spring jest platformą programistyczną opracowaną przez firmę Pivotal. Platforma oferuje funkcjonalności do tworzenia interfejsu użytkownika i modeli konfiguracyjnych nowoczesnych aplikacji. Dzięki tym funkcjonalnościom programiści mogą skupić się na implementowaniu algorytmu aplikacji. Dzisiaj niemal wszystkie nowoczesne aplikacje tworzy się przy użyciu platformy Spring. Jest to jedna z najczęściej stosowanych i najpopularniejszych platform programistycznych.

## Jakie funkcjonalności oferuje platforma Spring?

Platforma Spring jest wyposażona w mnóstwo nowoczesnych funkcjonalności niezbędnych każdemu programiście. W tej części rozdziału opisanych zostało kilka najważniejszych.

### Wstrzykiwanie zależności

Pierwsza funkcjonalność, o której należy wspomnieć i która jest wykorzystywana w codziennym programowaniu, to wstrzykiwanie zależności (ang. *dependency injection*). Każdy dobry programista stara się tworzyć klasy jak najbardziej uniezależnione od siebie. Są to tzw. klasy luźno sprzężone, czyli „nieświadome” istnienia innych klas. Wstrzykiwanie zależności służy wiązaniu klas ze sobą z zachowaniem niezależności.

Jak to się dzieje?

Wstrzykiwanie zależności to technika nowoczesnego programowania polegająca na udostępnianiu obiektu A obiektowi B. Zależnością jest obiekt A, natomiast obiekt B wykorzystuje wstrzykniętą zależność. Dobrym przykładem jest udostępnianie zależności w testach jednostkowych.

Zależność najczęściej definiujemy, tworząc **instancję zależności** lub **obiekt fabrykujący** generujący instancję. Dzięki wstrzykiwaniu zależności instancja jest przekazywana zewnętrznemu **klientowi**. Sposób wykorzystania instancji to problem zewnętrznego kodu, a nie Twojego! Co to jest ów zewnętrzny kod? Jest nim na przykład:

- obiekt znajdujący się wyżej w hierarchii zależności,
- wstrzykiwarka zależności (platforma) generująca graf zależności.

Poniżej znajduje się kilka przykładów wstrzykiwania zależności. Wykorzystane są w nich konstruktor i obiekt fabrykujący. W obu przypadkach rzecz się komplikuje, jeżeli trzeba dokładnie przetestować kod. Jednak dzięki platformie Spring jest to bardzo łatwe! W tej książce dowiesz się, jak to się robi.

Wstrzykiwanie zależności za pomocą konstruktora wygląda następująco:

```
class MyExampleClass(val parameter: Any){
    private val dependency: Any
    init {
        dependency = parameter
        // Kod wykorzystujący zależność.
    }
}
```

Wstrzykiwanie zależności za pomocą obiektu fabrykującego wygląda następująco:

```
class MyExampleClass2 {
    private val dependency = Factory().create()
}
class Factory {
    fun create(): Any {
```



```

        // Kod tworzący instancję zależności.
        return Any()
    }
}

```

Jak wspomniałem, dobrym przykładem wstrzykiwania zależności są testy kodu. Dzięki zależnościom testowanie jest o wiele prostsze! Jaki problem pojawia się w powyższych dwóch przykładach? Załóżmy, że chcemy przetestować klasę `MyExampleClass2`. Potrzebna jest jej imitacja (ang. *mock*) lub atrapa (ang. *stub*). Sprawę dodatkowo komplikuje to, że nie mamy wpływu na klasę `Factory`! Proszym sposobem byłoby wstrzykiwanie konstruktora, ale nie rozwiązałyby to problemu. Dlaczego? Wzajemne uzależnianie tworzonych obiektów jest w praktyce nie do przyjęcia. Tego problemu nie ma w przypadku wstrzykiwania zależności za pomocą setterów. Wkrótce przekonasz się jednak, że takie podejście ma więcej wad niż zalet.

## Odwrócenie sterowania

Najważniejszą techniką luźnego sprzęgania klas jest odwrócenie sterowania (ang. *Inversion of Control* — **IoC**). Polega ono na udostępnianiu zależności przez obiekt podrzędny zamiast tworzenia lub szukania obiektów zależnych. Sprzęganie obiektów następuje w trakcie działania kodu. Celem zarówno odwrócenia sterowania, jak i wstrzykiwania zależności jest redukcja liczby zależności w kodzie.

Przeanalizujmy kilka przykładów ilustrujących powyższą technikę. Wyobraź sobie, że tworzysz aplikację do odtwarzania muzyki i potrzebny jest mechanizm do sterowania głośnością. Początkowy kod wygląda następująco:

```

class VolumeControl
class MusicPlayer {
    val volumeControl = VolumeControl()
}

```

Teraz zastosujmy odwrócenie sterowania:

```

abstract class VolumeControlAbstract
class MusicPlayerIOC(
    private val volumeControl: VolumeControlAbstract
)

```

W pierwszym kodzie tworzona jest instancja klasy:

```
val volumeControl = VolumeControl()
```

Zatem klasa `MusicPlayer` jest bezpośrednio uzależniona od klasy `VolumeControl`!

Jak to wygląda w drugim kodzie? Tworzona jest w nim klasa abstrakcyjna. W sygnaturze konstruktora klasy `MusicPlayer` zdefiniowana jest klasa zależna `VolumeControl`. Nie jest jednak tworzona jej instancja. Zależność jest tworzona, a następnie przekazywana klasie `MusicPlayer` w następujący sposób:

```

//Zainicjowanie zależności.
val vc = VolumeControlImpl()

```

```
//Przekazanie zależności.  
val player = MusicPlayerIOC(vc)
```

W tym przypadku kod kliencki tworzący instancję klasy `MusicPlayer` ma kontrolę nad implementacją klasy `VolumeControl`. Jak widać, zależność jest wstrzykiwana w sygnaturze klasy `MusicPlayer`!

### Programowanie aspektowe

Platforma Spring oferuje jeszcze jedną ważną funkcjonalność: programowanie aspektowe (ang. *aspect oriented programming* — **AOP**). Programowanie aspektowe stosuje się do tworzenia obiektowych interfejsów API. Jest to nowy styl tworzenia struktury kodu. W programowaniu obiektowym podstawową jednostką jest klasa. Natomiast w programowaniu AOP jest to aspekt. Wstrzykiwanie zależności ma na celu rozprzęgnięcie obiektów tworzących aplikację. Natomiast programowanie AOP pozwala rozdzielać zagadnienia przekrojowe (ang. *cross-cutting concerns*) od obiektów, których te zagadnienia dotyczą. Zagadnienie przekrojowe jest to funkcjonalność wpływająca na wiele obszarów aplikacji. Dobrym przykładem jest zabezpieczanie kodu. Dlaczego? Ponieważ w wielu metodach może być wymagane zaimplementowane reguł bezpieczeństwa.

W programowaniu aspektowym każdy moduł implementuje aspekty. W następnych rozdziałach zastosujesz ten styl programowania w praktyce.

### Kontener

Ważną cechą platformy Spring jest możliwość tworzenia obiektów aplikacyjnych i zarządzania ich cyklem życia oraz konfiguracją. Do tego celu służy interfejs `org.springframework.context.ApplicationContext`, który jest odpowiedzialny za tworzenie instancji klas, ich konfigurowanie i kompletowanie. Platforma Spring oferuje kilka gotowych implementacji interfejsu `ApplicationContext`.

### Wzorzec MVC

Platforma Spring pozwala tworzyć aplikacje zgodnie ze wzorcem **MVC** (ang. *Model-View-Controller* — model-widok-kontroler). Co to oznacza? We wzorcu MVC również stosuje się interfejsy i różne technologie tworzenia widoków. Wzorzec ten wykorzystuje się do pisania interfejsów użytkownika. Dzięki niemu powstały kod jest prosty i czytelny. Zgodnie z tym wzorcem kod aplikacji dzieli się na trzy części: model, widok i kontroler.

#### Model

Model to centralna część aplikacji odpowiedzialna za jej działanie. Jest niezależny od widoku i ma bezpośredni wpływ na dane, algorytm i stosowane zasady.

#### Widok

Widok służy do prezentowania informacji przetwarzanych przez aplikację. Przykładem widoku jest strona HTML tworzona po wysłaniu zapytania do interfejsu API. Aplikacja nie musi mieć jednego widoku. We wzorcu MVC można prezentować tę samą informację w wielu widokach.

## Kontroler

Kontroler odbiera dane wejściowe i przetwarza je na polecenia dla modelu lub widoku. Jak widać, jest to kolejny przykład rozprzęgania klas w platformie Spring!

## Zarządzanie transakcjami

Platforma Spring oferuje ogólną warstwę abstrakcyjną do zarządzania transakcjami. Zarządzanie to obejmuje nie tylko środowisko J2EE. W odróżnieniu od mechanizmu EJB CMT, powiązanego z interfejsem JTA, deklaratywne zarządzanie transakcjami w platformie Spring można stosować w każdym środowisku. Zarządzać można transakcjami JTA, jak również transakcjami lokalnymi, jeżeli zajdzie taka potrzeba, przy użyciu następujących technik:

- JDBC,
- JPA,
- Hibernate,
- JDO.

Wystarczy jedynie odpowiednio zmienić pliki konfiguracyjne!

## Inne funkcjonalności

Wyjątki w warstwie abstrakcji JDBC w platformie Spring tworzą hierarchiczną strukturę, dzięki której kodowanie obsługi błędów jest łatwiejsze.

Platforma Spring umożliwia tworzenie komercyjnych aplikacji za pomocą obiektów **POJO** (ang. *Plain Old Java Object* — zwykły, stary obiekt Java). Co to oznacza? Że nie jest potrzebny kontener EJB, na przykład serwer aplikacji. Programista może użyć nawet podstawowego kontenera serwetów, na przykład Tomcata lub innego komercyjnego produktu. Jednym z możliwych rozwiązań jest wykorzystanie środowiska AWS.

Warto zaznaczyć, że platforma Spring ma modułową budowę. Oznacza to, że w aplikacji można wykorzystywać jedynie niezbędne pakiety, a inne pominąć.

Razem z platformą można stosować inne popularne technologie, na przykład platformę ORM, platformy logujące, środowisko JEE, czasomierze Quartz i JDK.

Jak wygląda testowanie aplikacji? Dzięki platformie Spring testy przeprowadza się szybko i łatwo. Kod uzależniony od otaczającego go środowiska przenosi się do platformy. Za pomocą obiektów POJO można łatwo wstrzykiwać zależności i dane potrzebne do wykonania testów.

Platforma Spring oferuje doskonały interfejs API do przekształcania wyjątków charakterystycznych dla różnych technologii na spójne wyjątki niekontrolowane.

Kontenery IoC w platformie Spring są bardzo małe. Pod tym względem znacznie różnią się od kontenerów EJB i w mniejszym stopniu obciążają pamięć i procesor komputera.

To są najpopularniejsze funkcjonalności platformy Spring. Wykorzystasz je podczas pracy z tą książką. Spotkasz się również z innymi funkcjonalnościami, które nie zostały tu wymienione. Gdy dogłębnie poznasz platformę Spring, na pewno przez długi, bardzo długi czas, a być może już nigdy, nie skorzystasz z innych platform!

## Jak jest obsługiwany język Kotlin?

Począwszy od wersji 5 platforma Spring oficjalnie obsługuje język Kotlin. To świetna wiadomość! Jedną z największych zalet Kotlina jest jego doskonała kompatybilność z bibliotekami języka Java. Ale to nie wszystko! Tworząc aplikacje przy użyciu platformy Spring, można w pełni korzystać z zalet składni języka Kotlin. Dzięki temu osiąga się zupełnie inny poziom wydajności i elastyczności kodu! Dlatego począwszy od wersji 5 platformy Spring język Kotlin jest oficjalnie obsługiwany.

Podstawową funkcjonalnością języka Kotlin, umożliwiającą stosowanie go razem z platformą Spring, są rozszerzenia. Dzięki nim można rozbudowywać istniejące klasy platformy, co wprowadza najnowszą wersję platformy w zupełnie nowy wymiar!

Należy podkreślić, że rozszerzenia w języku Kotlin są rozwiązywane statycznie. Oznacza to, że trzeba je importować.

Poniżej wymienione są najważniejsze korzyści płynące ze stosowania platformy Spring i języka Kotlin:

- Platforma Spring może wykorzystywać charakterystyczne dla języka Kotlin zabezpieczenia przed zerowymi wskaźnikami.
- Do kierowania zapytań można wykorzystywać język DSL.
- Dzięki urzeczowionym parametrom typowanym można uniknąć problemów wynikających z wymazywania typów w maszynie JVM.
- Narzędzie Gradle można konfigurować w języku Kotlin.
- Można stosować szablony skryptów w języku Kotlin.
- Stosując platformę Spring z językiem Kotlin, można tworzyć krótszy, prostszy, bardziej efektywny i czytelny kod.

Język Kotlin to przyszłościowe rozwiązanie dla wszystkich programistów korzystających z platformy Spring!

## Tworzenie projektu Spring

Przyszł czas na utworzenie i uruchomienie pierwszego projektu opartego na platformie Spring. Twoim podstawowym językiem programowania będzie oczywiście Kotlin. Do zainicjowania projektu wykorzystasz narzędzie Spring Initializr.

## Czym jest narzędzie Spring Initializr?

W skrócie: Spring Initializr jest internetowym generatorem projektów opartych na platformie Spring. Narzędzie to jest dostępne pod adresem <https://start.spring.io>. Ponieważ jest to otwarty projekt, jego kod znajduje się w repozytorium GitHub pod adresem <https://github.com/spring-io/initializr>.

Spring Initializr to narzędzie udostępniające interfejs API do szybkiego generowania projektów opartych na platformie Spring. Możesz użyć jego domyślnej instancji o podanym wyżej adresie lub sklonować jego repozytorium w serwisie GitHub i utworzyć własną instancję.

Narzędzie Spring Initializr ma kilka opcji konfiguracyjnych, za pomocą których definiuje się różne aspekty tworzonego projektu, między innymi:

- listę zależności,
- wersję języka Java,
- wersję języka Kotlin,
- wersję platformy Spring Boot.

## Utworzenie projektu

Otwórz stronę narzędzia Spring Initializr: <https://start.spring.io>. Pojawi się kreator projektów, jak na poniższym rysunku:

The screenshot shows the Spring Initializr web interface in a browser window. The address bar displays `https://start.spring.io`. The page header reads "SPRING INITIALIZR bootstrap your application now". The main configuration area includes three dropdown menus: "Generate a" (Maven Project), "with" (Java), and "and Spring Boot" (2.0.6). Below these are two sections: "Project Metadata" and "Dependencies".

**Project Metadata:**

- Artifact coordinates: Group: `com.example`, Artifact: `demo`

**Dependencies:**

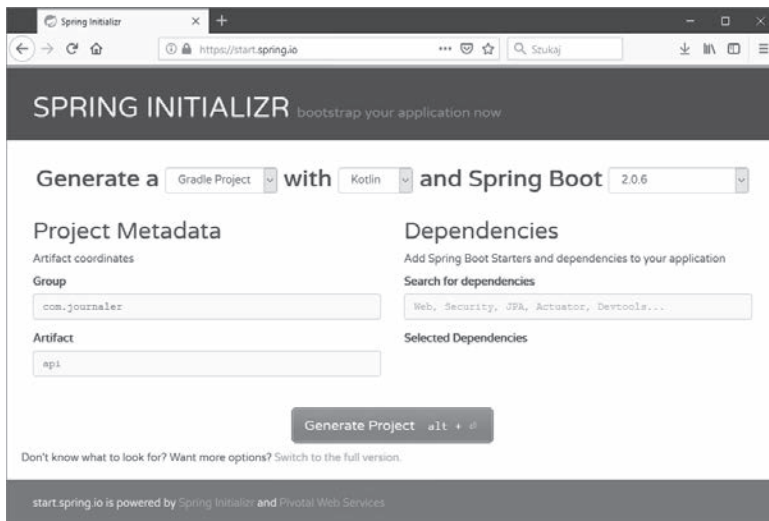
- Add Spring Boot Starters and dependencies to your application
- Search for dependencies: `Web, Security, JPA, Actuator, Devtools...`
- Selected Dependencies: (empty)

A "Generate Project" button is located at the bottom of the form. A footer note states: "start.spring.io is powered by Spring Initializr and Pivotal Web Services".

Wybierz następujące ustawienia projektu:

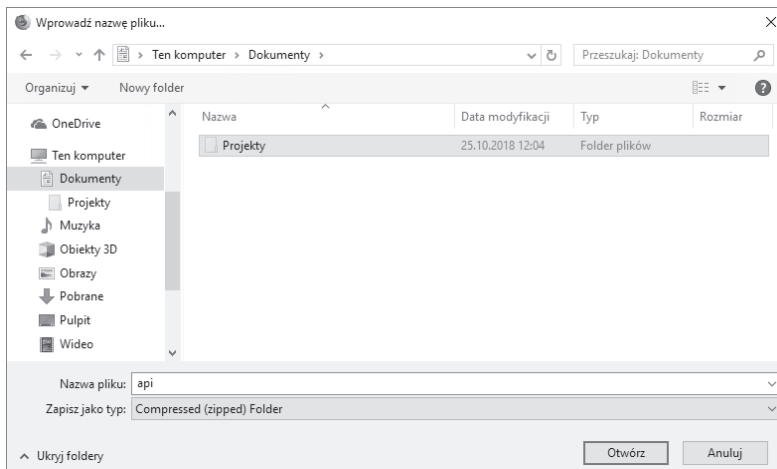
- Wybierz *Generate a Gradle Project with Kotlin and Spring Boot 2.0.6* (wygeneruj projekt Gradle z językiem Kotlin i platformą Spring Boot 2.0.6).
- W polu *Group* wpisz *com.journaler*.
- W polu *Artifact* wpisz *api*.

Poniżej przedstawiony jest widok ustawień:

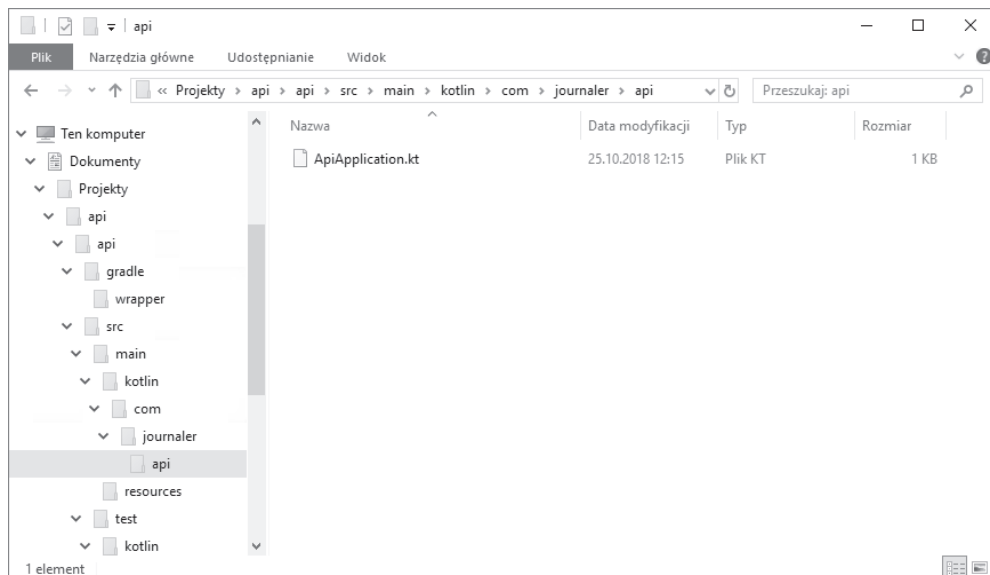


Pierwsza konfiguracja będzie prosta, ale później w miarę swoich postępów będziesz ją rozszerzał o dodatkowe zależności.

Kliknij przycisk *Generate Project* (wygeneruj projekt). Pojawi się okno, jak na poniższym rysunku, z pytaniem o folder, w którym ma być zapisany plik z projektem:



Zapisz plik *api.zip* w preferowanym folderze, a następnie rozpakuj go. Poniższy rysunek przedstawia zawartość projektu:



Jest to standardowy projekt aplikacji w języku Kotlin. Przyjrzyjmy się dokładniej jego najważniejszym częściom. Otwórz plik konfiguracyjny *build.gradle*. Jego zawartość powinna być podobna do tej niżej przedstawionej:

```
buildscript {
    ext {
        kotlinVersion = '1.2.51'
        springBootVersion = '2.0.6.RELEASE'
    }
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plugin:${springBootVersion}")
        classpath("org.jetbrains.kotlin:kotlin-gradle-plugin:${kotlinVersion}")
        classpath("org.jetbrains.kotlin:kotlin-allopen:${kotlinVersion}")
    }
}

apply plugin: 'kotlin'
apply plugin: 'kotlin-spring'
apply plugin: 'eclipse'
apply plugin: 'org.springframework.boot'
apply plugin: 'io.spring.dependency-management'
```

```

group = 'com.journaler'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = 1.8
compileKotlin {
    kotlinOptions {
        freeCompilerArgs = ["-Xjsr305=strict"]
        jvmTarget = "1.8"
    }
}
compileTestKotlin {
    kotlinOptions {
        freeCompilerArgs = ["-Xjsr305=strict"]
        jvmTarget = "1.8"
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation('org.springframework.boot:spring-boot-starter')
    implementation("org.jetbrains.kotlin:kotlin-stdlib-jdk8")
    implementation("org.jetbrains.kotlin:kotlin-reflect")
    testImplementation('org.springframework.boot:spring-boot-starter-test')
}

```

Kolejne sekcje skryptu, począwszy od samego początku, zawierają następujące ustawienia:

- oznaczenia wersji języka Kotlin i platformy Spring Boot,
- nazwy i ścieżki repozytoriów niezbędnych do skompilowania kodu,
- wtyczki niezbędne do skompilowania i uruchomienia kodu,
- nazwę grupy projektu (`com.journaler`),
- oznaczenie wersji projektu (`0.0.1-SNAPSHOT`),
- wymóg kompatybilności z wersją języka Java 1.8,
- nazwy wykorzystywanych repozytoriów,
- zależności wykorzystywane przez język Kotlin i platformę Spring.

Inną ważną kwestią jest konfiguracja narzędzia Git. Otwórz plik `.gitignore`, zawierający ustawienia wyłączające niepotrzebną kontrolę wersji w przypadku niektórych plików, i przyjrzyj się jego zawartości:

```

.gradle
/build/
!gradle/wrapper/gradle-wrapper.jar

### STS ###
.appt_generated
.classpath
.factorypath

```



```

.project
.settings
.springBeans
.sts4-cache

### IntelliJ IDEA ###
.idea
*.iws
*.iml
*.ipr
/out/

### NetBeans ###
/nbproject/private/
/nbbuild/
/dist/
/nbdist/
/.nb-gradle/

```

Otwórz plik *application.properties*, który znajduje się w folderze *src\main.resources*. Plik ten zawiera ustawienia aplikacji specyficzne dla wykorzystywanego środowiska. W tej chwili jest on pusty.

I wreszcie: przejrzyj kod! Projekt został utworzony w języku Kotlin, więc przejdź do folderu *src\main\kotlin\com\journaler\api* i otwórz plik *ApiApplication.kt*:

```

package com.journaler.api

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication

@SpringBootApplication
class ApiApplication

fun main(args: Array<String>) {
    runApplication<ApiApplication>(*args)
}

```

Kod nie jest długi, więc łatwo go zrozumieć. Zdefiniowana jest w nim klasa aplikacyjna *ApiApplication* opatrzona adnotacją *@SpringBootApplication*. Metoda *main()* uruchamia aplikację wykorzystującą platformę Spring. To wszystko!

Zanim skompilujesz i uruchomisz kod, musisz sprawdzić jedną rzecz. Przejdź do folderu *src\test\kotlin\com\journaler\api* i otwórz plik *ApiApplicationTest.kt*, który zawiera następujący test:

```

package com.journaler.api

import org.junit.Test
import org.junit.runner.RunWith
import org.springframework.boot.test.context.SpringBootTest
import org.springframework.test.context.junit4.SpringRunner

```

```
@RunWith(SpringRunner::class)
@SpringBootTest
class ApiApplicationTests {

    @Test
    fun contextLoads() {
    }

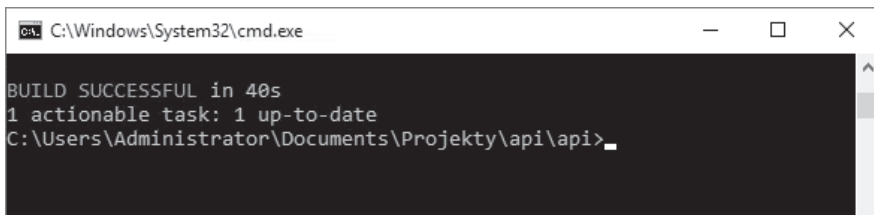
}
```

Powyższy test nie robi nic. Więcej informacji o testowaniu aplikacji opartych na platformie Spring znajdziesz w następnych rozdziałach książki. Na razie ogólnie zapoznaj się z kodem testu.

Teraz skompiluj i uruchom aplikację. W tym celu otwórz wiersz poleceń, przejdź do głównego folderu projektu i wpisz następujące polecenie:

```
C:\Users\Administrator\Documents\Projekty\api\api>gradlew clean
```

Narzędzie Gradle pobierze wymagane zależności:

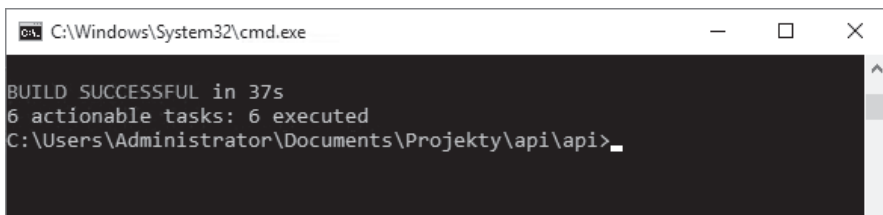


```
C:\Windows\System32\cmd.exe
BUILD SUCCESSFUL in 40s
1 actionable task: 1 up-to-date
C:\Users\Administrator\Documents\Projekty\api\api>
```

Powyższy komunikat oznacza, że zależności zostały pomyślnie pobrane. Aby skompilować kod, wpisz następujące polecenie:

```
C:\Users\Administrator\Documents\Projekty\api\api>gradlew build
```

Jeżeli kompilacja zakończy się pomyślnie, pojawi się taki komunikat:



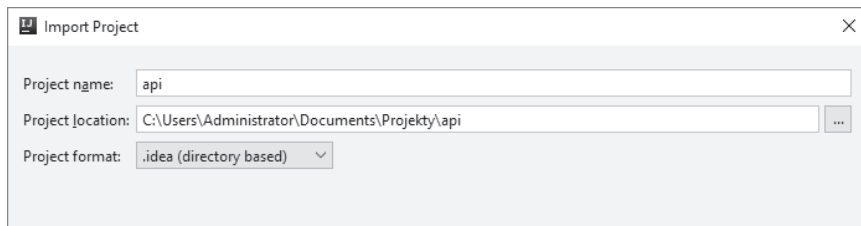
```
C:\Windows\System32\cmd.exe
BUILD SUCCESSFUL in 37s
6 actionable tasks: 6 executed
C:\Users\Administrator\Documents\Projekty\api\api>
```

Aby uruchomić aplikację, przejdź do folderu *build/libs* i wpisz polecenie:

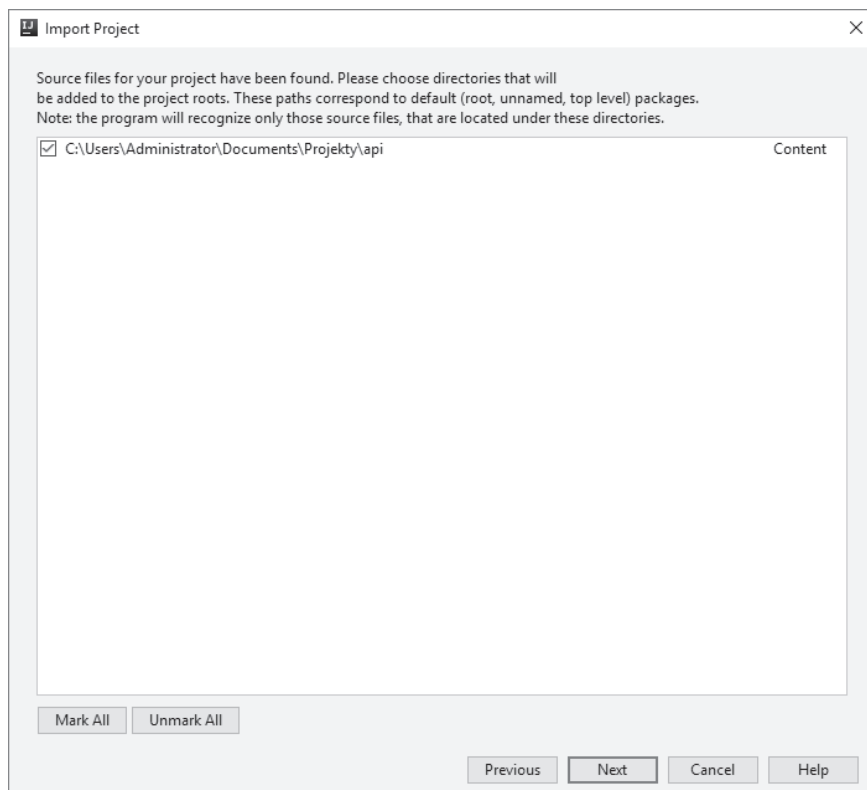
```
C:\Users\Administrator\Documents\Projekty\api\api\build\
↳libs>java -jar ./api-0.0.1-SNAPSHOT
```



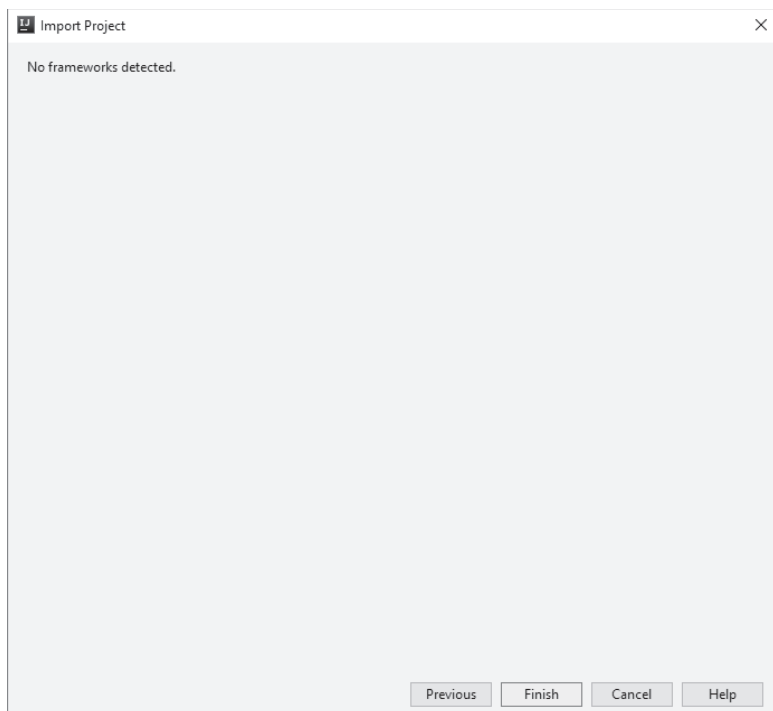
Kliknij przycisk *Next*. Pojawi się kolejne okno z polem *Project name* zawierającym nazwę projektu:



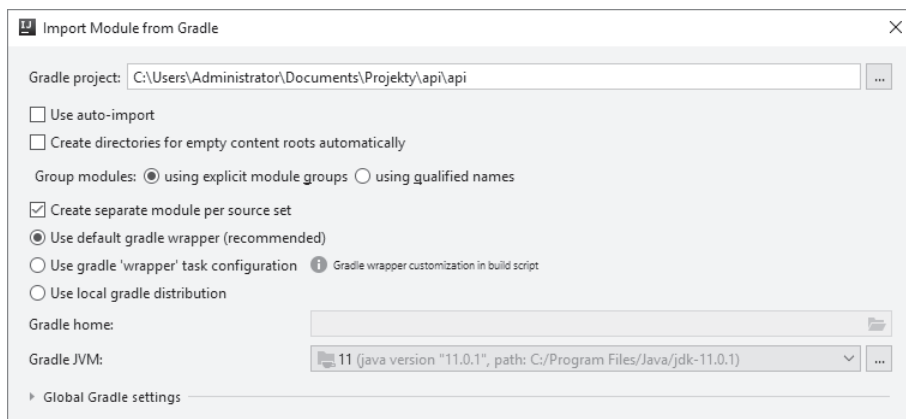
Jeżeli nazwy projektu i folderu Ci odpowiadają, kliknij przycisk *Next*. W następnym oknie kliknij przycisk *Mark All* (zaznacz wszystko):



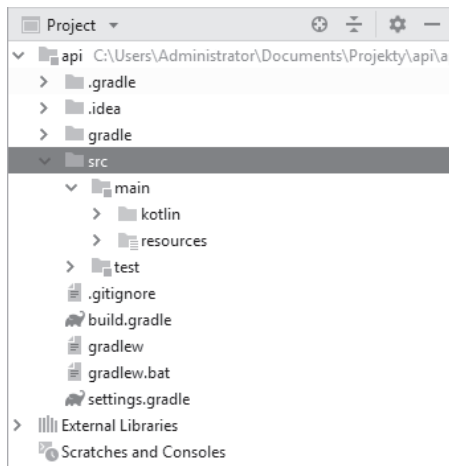
W ostatnim oknie kliknij przycisk *Finish*:



Następnie kliknij polecenie menu *File/New/Module from Existing Sources* (plik/nowoty/moduł z istniejących źródeł) i zaimportuj moduł Gradle. W oknie z ustawieniami importu kliknij przycisk *OK*:



Import modułu zajmie chwilę. Rysunek na następnej stronie przedstawia strukturę projektu aplikacji gotowej do uruchomienia:



Jeżeli środowisko programistyczne nie rozpozna automatycznie konfiguracji projektu, rozwiń sekcję *api/src/main/kotlin/com.journaler.api*, kliknij prawym przyciskiem myszy plik *ApiApplication.kt* i wybierz polecenie *Run 'com.journaler.api.ApiApplication.kt'*. Aplikacja powinna wyświetlić takie same komunikaty jak plik JAR, który uruchomiłeś wcześniej.

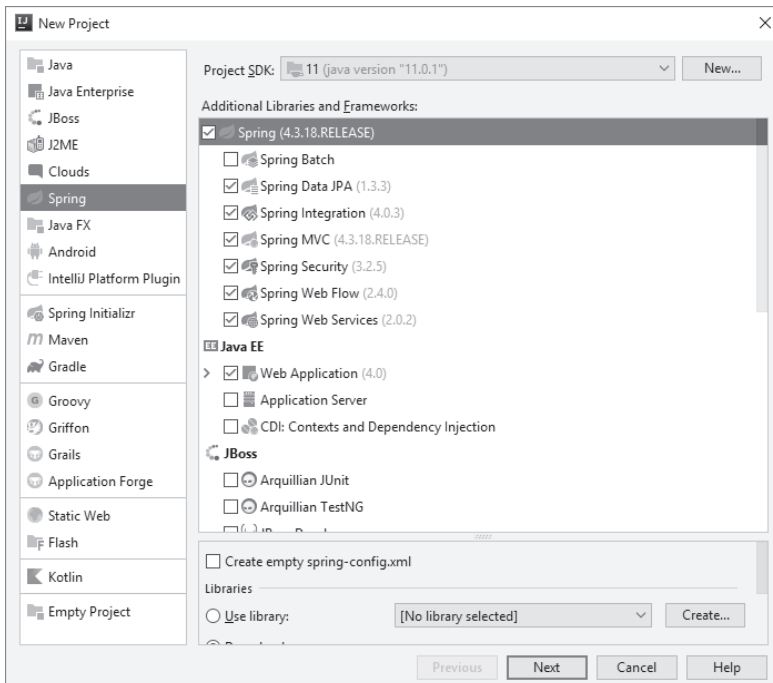
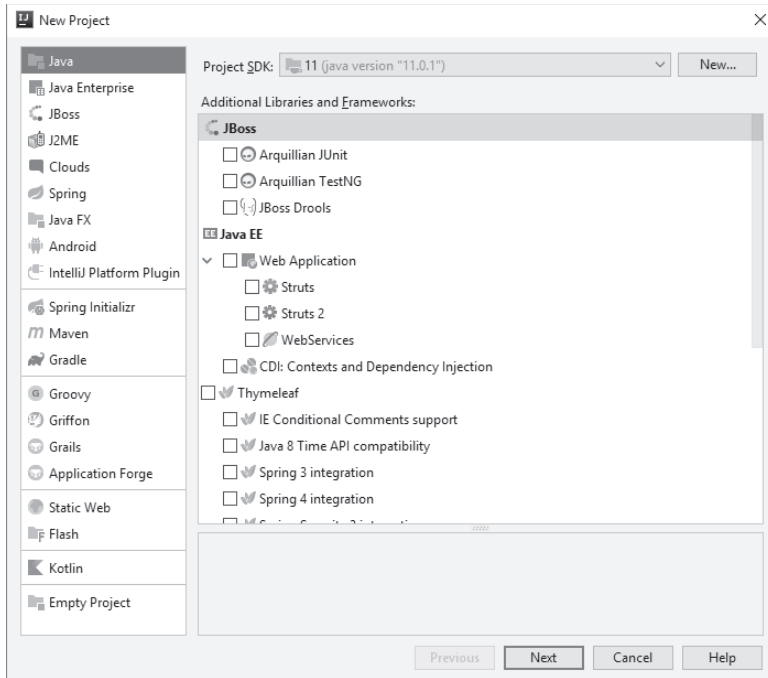
W pliku *.gitignore* wpisz poniższe wiersze (których narzędzie Spring Initializr nie utworzyło) wyłączające kontrolę wersji w przypadku plików narzędzia Gradle:

```
gradlew
gradlew.bat
gradle/*
```

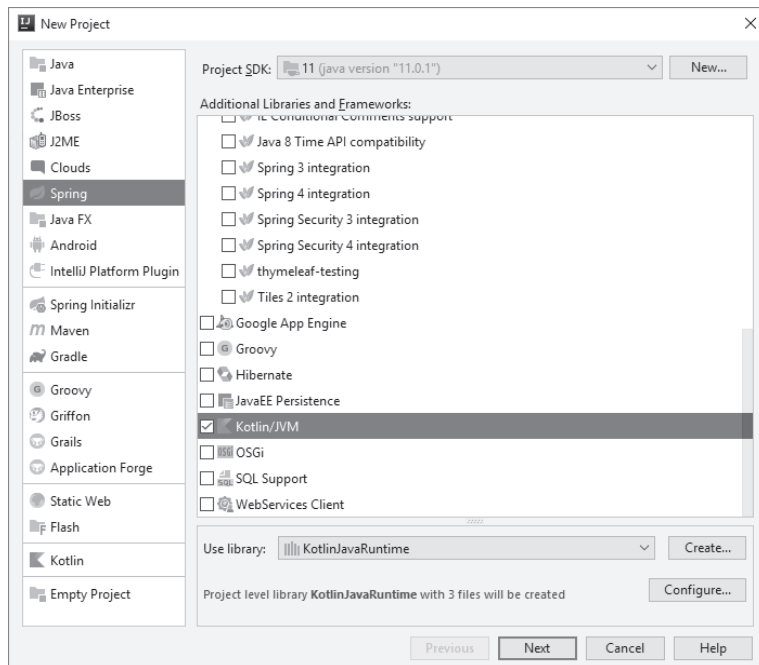
## Tworzenie projektu w środowisku IntelliJ IDEA

Na zakończenie rozdziału dowiesz się, jak inicjuje się projekty w środowisku IntelliJ IDEA. Efekt będzie taki sam jak ten uzyskany za pomocą narzędzia Spring Initializr. Uruchom środowisko IntelliJ IDEA i kliknij opcję *Create New Project* (utwórz nowy projekt). Pojawi się okno *New Project* (zobacz pierwszy rysunek na następnej stronie).

Kliknij sekcję *Spring* i zaznacz opcje jak na drugim rysunku na następnej stronie.



Ponadto zaznacz opcję *Kotlin/JVM*, jak niżej:



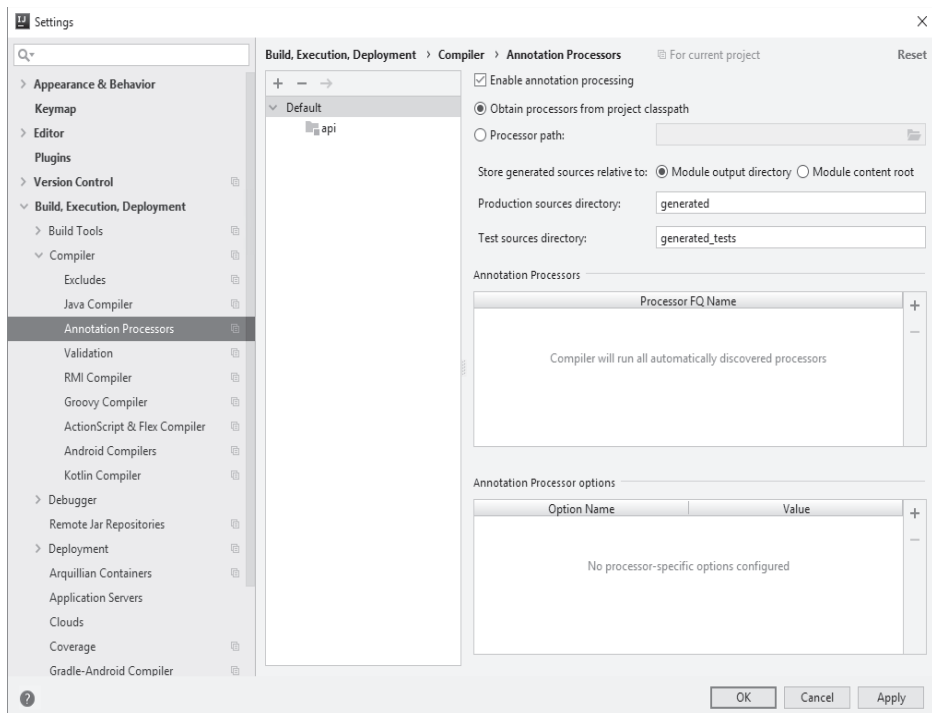
Kliknij przycisk *Next* i dokończ tworzenie projektu zgodnie ze wskazówkami kreatora. Następnie kliknij polecenie menu *File/Settings*. W oknie, które się pojawi, rozwiń po lewej stronie sekcję *Build, Execution, Deployment/Compiler/Annotation Processing* (kompilacja, uruchomienie, wdrożenie/kompilator/przetwarzanie adnotacji) i zaznacz opcję *Enable annotation processing* (włącz przetwarzanie adnotacji) — zobacz rysunek na następnej stronie.

Jak widać na rysunku na następnej stronie, w środowisku brak jest ustawień platformy Spring. Dlatego będziesz tworzył projekty przy użyciu narzędzia Spring Initializr — zawiera ono wszystkie niezbędne opcje.

## Podsumowanie

Ten rozdział był wprowadzeniem do platformy Spring i jej funkcjonalności. Utworzyłeś swój pierwszy projekt, pomyślnie go skompilowałeś i uruchomiłeś. W następnym rozdziale będziesz kontynuował rozpoczętą podróż i zajmiesz się bardziej konkretnymi zadaniami: utworzysz swoją pierwszą usługę REST, która będzie się składać z kontrolera, klasy danych i komponentów usługowych.







# Skorowidz

## A

adnotacja  
  @Component, 73  
  @Controller, 74  
  @Repository, 74  
  @RunWith, 191  
  @Service, 74  
AOP, aspect oriented programming, 36  
API, application programming interface, 14  
  modyfikacja interfejsu, 143  
architektura  
  mikrouslugowa, 132  
  SOA, 131  
autoryzowanie użytkowników, 123  
AWS, Amazon Web Services, 214

## B

baza danych MySQL, 79  
bezpieczeństwo, 111  
biblioteka  
  Actuator, 71  
  Spring Data, 80  
brama, 134, 140

## C

chmura AWS, 214  
Created on, 15  
CRUD, create, read, update, delete, 14, 89

## D

DAO, Data Access Object, 74  
definiowanie obiektów DTO, 119  
dobre praktyki, 173  
dodanie  
  danych, 95  
  komponentu Service, 73  
  zależności, 88  
DTO, data transfer objects, 100, 119  
dzielenie kodu, 14

## E

Email, 15  
Enabled, 15

## F

First, 15  
funkcjonalności, 34

## I

ID, 15  
identyfikator UUID, 15  
implementacja  
  klas reprezentujących role, 112  
  operacji CRUD, 89  
informacje  
  o ostatnich zapytaniach HTTP, 71  
  o stanie aplikacji, 71  
  o środowisku, 71

instalacja  
 bazy MySQL, 80–84  
 narzędzia Git, 17  
 pakietu JDK, 19  
 platformy Spring 5, 24  
 pliku TAR, 85  
 programu Postman, 29  
 środowiska programistycznego, 21  
 IntelliJ IDEA, 21  
 tworzenie projektu, 48  
 uruchamianie testów, 189  
 interfejs  
 API, 14, 143  
 REST API, 160  
 Spring Data JPA, 79, 80  
 UserDetails, 113  
 IoC, Inversion of Control, 35

**J**

jednostki, 15  
 język Kotlin, 38  
 JPA, Java Persistence API, 79

**K**

klasa  
 danych, 57  
 kontrolera, 55  
 kod  
 implementujący operacje, 78  
 obsługujący zapytania, 78  
 kompilacja kodu źródłowego, 18  
 komponent Service, 73  
 konfiguracja  
 narzędzia Gradle, 28  
 narzędzia Maven, 28  
 XML, 173  
 kontener, 36  
 kontroler, 37

**L**

Last name, 15  
 Location, 15

**M**

Message, 15  
 metoda  
 getAuthorities(), 113  
 getPassword(), 113

getUsername(), 113  
 isAccountNonExpired(), 114  
 isAccountNonLocked(), 114  
 isCredentialsNonExpired(), 113  
 isEnabled(), 113  
 metodyki testowania, 178  
 mikrousługi, 132  
 model, 36  
 moduły Spring Data, 80  
 modyfikacja  
 danych, 95  
 interfejsu API, 143  
 mutowalne zmienne, 172  
 MVC, Model-View- -Controller, 36  
 MySQL, 80  
 instalacja bazy, 81–84

**N**

narzędzie  
 Git, 17, 18  
 Gradle, 28  
 Maven, 28  
 Spring Initializr, 39  
 Note, 15

**O**

obiekty DTO, 100, 119  
 odczytywanie danych, 97  
 odwrócenie sterowania, 35  
 opcje wdrożeniowe, 208  
 operacje CRUD, 14, 89

**P**

pakiet JDK, 19  
 Password, 15  
 pierwsza usługa, 53  
 planowanie pracy, 16  
 platforma  
 Spring, 33  
 Spring 5, 24  
 Spring Cloud, 131  
 Spring Security, 111  
 pliki, 180  
 POJO, Plain Old Java Object, 37  
 poprawność danych, 173  
 praktyki programistyczne, 171  
 problem wielowłokowości, 174

program Postman, 29  
 programowanie aspektowe, 36  
 projekt Reactor, 159

**R**

Reactor, 159  
   komponenty, 160  
   stosowanie, 159  
 repozytorium Git, 30  
 role użytkowników, 112  
 Roles, 15

**S**

schemat bazy danych, 86  
 serwer  
   Java EE, 210  
   konfiguracji, 134, 135  
   Tomcat, 208  
   wykrywania usług, 134  
 SOA, service-oriented architecture, 131  
 Spring, 33  
   Cloud, 131  
     mikrousługi, 133  
     zabezpieczanie usług, 148  
   Initializr, 39  
   Security, 111, 129

**Ś**

środowisko  
   programistyczne IntelliJ IDEA, 21  
   robocze, 16

**T**

testowanie, 173, 177  
   aplikacji Spring REST, 191  
   w IntelliJ IDEA, 189  
   w języku Kotlin, 183  
 Title, 15  
 TODO, 15  
 transakcje, 37  
 tworzenie  
   klasy danych, 57  
   klasy kontrolera, 55  
   projektu, 38  
   projektu w środowisku IntelliJ IDEA, 48  
   repozytorium Git, 30

schematu bazy danych, 86  
 zapytań SQL, 106

**U**

Updated on, 15  
 uruchamianie  
   aplikacji, 65  
   testów, 189  
   zestawu testów, 198  
 User, 15  
 usługi Spring Cloud, 148  
 usuwanie danych, 98  
 UUID, Universally Unique Identifier, 15  
 uwierzytelnianie użytkowników, 123

**W**

walidator, 174  
 wdrażanie aplikacji, 201  
   na serwerze Java EE, 210  
   na serwerze Tomcat, 208  
   opcje wdrożeniowe, 208  
   w chmurze AWS, 214  
 weryfikacja danych, 174  
 widoczność, 173  
 widok, 36  
 wielowątkowość, 172, 174  
 wskaźniki, 71  
 wstrzykiwanie zależności, 34, 171  
 wykrywanie serwerów, 137  
 wzorzec MVC, 36

**X**

XML, 173

**Z**

zabezpieczanie  
   aplikacji, 111  
   usług Spring Cloud, 148  
 zakres testów, 173  
 zależności, 54, 88  
 zapytania, 58, 78  
   nazwane SQL, 106, 108  
 zarządzanie transakcjami, 37  
 zestaw testów, 198  
 zmienne mutowalne, 172



# PROGRAM PARTNERSKI

— GRUPY HELION —

1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

GRUPA  
**Helion** 

# Kotlin: tak świetny jak Java, ale o wiele łatwiejszy i przyjemniejszy!

Kotlin jest interoperacyjnym, obiektowym i coraz popularniejszym językiem programowania. Charakteryzuje się statycznie typowanymi zmiennymi, czytelną składnią i znakomitą reaktywnością. Z kolei Spring jest wyjątkowo cenioną platformą do budowania stron WWW. Służy do tworzenia aplikacji internetowych udostępniających interfejs REST i inne usługi. Spring znakomicie współpracuje z Kotlinem, co pozwala programiście na wyższy poziom skutecznego programowania i efektywne tworzenie funkcjonalnych, stabilnych, skalowalnych i reaktywnych aplikacji. Przy tym wszystkim obydwa te produkty w całości są otwartym oprogramowaniem!

Ta książka jest przewodnikiem dla osób, które chcą szybko poznać platformę Spring i zacząć posługiwać się językiem Kotlin w stopniu pozwalającym na tworzenie aplikacji internetowych. Zawiera przystępne wprowadzenie do pracy na platformie Spring i jej konfiguracji dla potrzeb Kotlin; omawia też zasady projektowania aplikacji za pomocą tych narzędzi. Sporo uwagi autor poświęca budowaniu systemu mikrousług udostępniających interfejs REST. Opisuje również techniki posługiwania się tak pożytecznymi narzędziami jak Spring Data, Spring Security czy biblioteka JUnit, a także sposoby wdrażania aplikacji w chmurze AWS.

## W książce między innymi:

- przygotowanie platformy Spring do pracy z Kotlinem
- korzystanie z bazy danych MySQL
- usługi REST i programowanie reaktywne
- zabezpieczanie aplikacji za pomocą Spring Security
- dobre praktyki programistyczne, testowanie i wdrażanie aplikacji

**Miloš Vasić** jest serbskim programistą, autorem książek o programowaniu i entuzjastą otwartego oprogramowania. Ukończył studia na uniwersytecie w Singidunum, ze specjalizacją w dziedzinie grafiki komputerowej i programowania dla Androida. Stara się dzielić swój czas między pisanie kolejnej książki a pracę nad nowym — oczywiście otwartym — projektem.

|  |  |  |
|--|--|--|
| <br><b>Helion</b>  | <i>Sprawdź nasze szkolenia!</i><br>SZKOLENIA<br><br>AKADEMIA IT & BUSINESS<br>WWW.SZKOLENIA.HELION.PL | <b>KOD KORZYŚCI</b><br>Sięgnij po więcej! <br><br>ISBN 978-83-283-5183-7<br><br>9 788328 351837 |
|  <a href="http://helion.pl">helion.pl</a>  |  |  |
|  <b>HELION SA</b><br>ul. Kościuszki 1c<br>44-100 Gliwice<br>tel.: 32 230 98 63<br>helion@helion.pl |  |  |
| <b>INFORMATYKA W NAJLEPSZYM WYDANIU</b>  |  | Cena: 49,00 zł   |