



**AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE**

Spring i Spring Boot

Sylwia Oleś Arkadiusz Kasprzak

10 maja 2020



- 1 Wprowadzenie do Spring Boot**
- 2 Działanie Spring Boot**
- 3 Spring Boot Actuator**
- 4 Struktura projektu i dobre praktyki**
- 5 Testowanie za pomocą Spring Boot**
- 6 Konfiguracja w Spring Boot**
- 7 Integracja ze Spring Security**



AGH Po co używać Spring Boot?

- Używanie frameworka Spring często niesie za sobą konieczność długiej i powtarzalnej konfiguracji używanych w projekcie zależności.
- Konieczne jest tworzenie sporej liczby plików `.xml` i/lub klas, często na zasadzie kopiowania gotowych rozwiązań.
- Takie podejście daje z jednej strony dużą elastyczność, ale jeśli chcemy po prostu zrobić coś w sposób standardowych to dodaje nam sporo pracy.
- Spring Boot jest odpowiedzią na ten problem.



AGH Co to właściwie jest Spring Boot?

- Zbiór predefiniowanych konfiguracji pozwalających w prosty sposób korzystać z domyślnych rozwiązań - może to być np. dodanie do aplikacji obsługi jakiejś bazy danych.
- Bardzo prosta zasada działania: na etapie **startowania aplikacji** Spring Boot skanuje *classpath* (lokalizacja, w której znajdują się pliki *.class* i pakiety) i na podstawie jego zawartości **konfiguruje te komponenty**, które są nam potrzebne.
- W dalszej części prezentacji pokazane zostaną niektóre szczegóły działania tego mechanizmu.



AGH Co to właściwie jest Spring Boot?

Ponadto Spring Boot:

- dostarcza narzędzi do monitorowania stanu aplikacji.
- dostarcza narzędzia wzbogacające możliwość pisania testów.
- dostarcza wbudowany serwer Tomcat (i możliwość zamiany na Jetty czy Undertow).

Spring Boot nie jest więc jedynie narzędziem do szybkiego generowania projektów.



AGH Aplikacja w Spring vs. Spring Boot

Przykład 1: Porównanie aplikacji napisanej w Spring z analogiczną w Spring Boot

Tutaj jakies madre porownanie jednego z drugim na jakims mocno wymownym przykladzie. Moze byc jakis z baza danych. Pokazać jak np. zmienic port i co to jest plik properties.

Albo to (Thymeleaf) bo w sumie w temacie: https://www.baeldung.com/spring-vs-spring-boot?fbclid=IwAR1Fc4nibRyxSLk00jIHQSUBS0xtLhQPyXlv1H7kMzAWD1R1sQ0_hxyJqGg



- Przykład 2: Minimalna aplikacja w Spring Boot
- Dwa główne elementy:
 - adnotacja `@SpringBootApplication`
 - klasa `SpringApplication`
- Zrozumienie ich działania pozwala w łatwy sposób kontrolować to, co dzieje się podczas konfiguracji aplikacji.



- Umieszczona zwykle na poziomie głównej klasy w aplikacji
- Równoważna trzem innym adnotacjom z ich domyślnymi atrybutami:
 - @Configuration
 - @EnableAutoConfiguration
 - @ComponentScan
- **DODAC DO NOTATEK: pokazac ze tak faktycznie jest - go to declaration**
- Te adnotacje są częścią frameworka Spring
- Aby nie wchodzić zbyt wiele w szczegóły działania samego Springa skupimy się na drugiej i trzeciej.



- używana razem z adnotacją @Configuration
- odpowiada za skanowanie w celu poszukiwania w projekcie komponentów (*Spring Bean*)
- użycie jej bez atrybutów oznacza: znajdź komponenty w tym pakiecie oraz wszystkich pod-pakietach
- komponenty znalezione w ten sposób mogą być następnie m.in. wstrzykiwane za pomocą adnotacji @Autowired
- adnotacja daje sporo możliwości - pozwala m.in. podać pakiet bazowy czy definiować, które komponenty powinny zostać pominięte
- ponieważ jednak @SpringBootApplication używa jej w sposób domyślny, nie będziemy się na tym skupiać
- Przykład 3: Przykład użycia @ComponentScan



go to declaration na jakiejś autokonfiguracji żeby pokazać że to tam faktycznie jest

- Wprowadza do działania system automatycznej konfiguracji
- Celem jest dokonanie przez framework automatycznej konfiguracji aplikacji na podstawie zawartości *classpath* - czyli w skrócie na podstawie naszych komponentów i zależności z pliku POM
- Mechanizm jest nieinwazyjny - automatyczna konfiguracja jest wdrażana tylko wtedy, gdy spełnione są odpowiednie warunki: dodaliśmy odpowiednie zależności i nie nadpisaliśmy konfiguracji sami
- Takie podejście sprawia, że chcąc zrobić coś niestandardowo nie musimy „walczyć” z frameworkiem.
- Przykład 4: Przykład użycia @EnableAutoConfiguration



AGH

Klasa SpringApplication





- Zbiór wygodnych deskryptorów zależności
- Przykład dla Maven:

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-web</artifactId>  
</dependency>
```

- Każda wersja Spring Boot wspiera konkretne wersje zależności tak, by nie pojawiały się żadne konflikty.



- Narzędzie pozwalające monitorować i zarządzać aplikacją m.in. za pomocą specjalnego zestawu *endpointów* HTTP.
- Możemy monitorować np.: stan aplikacji (ang. *health*), listę komponentów wchodzących w skład aplikacji czy listę *endpointów* aplikacji.
- W pliku *application.properties* możemy łatwo konfigurować dostępność tych *endpointów* - nie wszystkie dostępne są domyślnie (od wersji 2.0 większość jest domyślnie wyłączona).
- Rozbudowane narzędzie, tutaj zaprezentowane tylko podstawy.
- Przykład 5: użycie Spring Boot Actuator na prostej aplikacji CRUD.



Niektóre *endpointy*:

- `health` - podsumowanie stanu naszej aplikacji
- `shutdown` - wyłączenie aplikacji
- `info` - ogólne informacje
- `bean` - lista komponentów Spring-owych
- `logfile` - logi aplikacji
- `metrics` - szczegółowe metryki aplikacji
- `mappings` - mapowania ścieżek (`@RequestMapping`)
- `httptrace` - ostatnie zapytania HTTP

Istnieje możliwość tworzenia własnych *endpointów* jak również rozszerzania funkcjonalności tych domyślnych.



AGH

Struktura projektu i dobre praktyki



AGH

Testowanie za pomocą Spring Boot



AGH

Plik .properties

