Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Spring i Spring Boot

Sylwia Oleś Arkadiusz Kasprzak

16 maja 2020

Plan prezentacji

1. Wprowadzenie do Spring Boot
2. Działanie Spring Boot
3. Spring Boot Actuator
4. Struktura projektu i dobre praktyki

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.

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.

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.

Aplikacja w Spring vs. Spring Boot

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

- prosta aplikacja pozwalająca na tworzenie i wyświetlanie rekordów w bazie danych
- wykorzystuje silnik Thymeleaf w warstwie prezentacji
- wykorzystuje bazę danych H2
- znaczna różnica w ilości napisanego przez programistę kodu

Działanie Spring Boot

- Przykład 2: Minimalna aplikacja w Spring Boot
- Dwa główne elementy:
  - adnotacja @SpringBootApplication
  - klasa SpringApplication
- Klasa SpringApplication odpowiada m.in. za uruchomienie aplikacji i stworzenie instancji ApplicationContext.
- W dalszej części skupimy się na adnotacji @SpringBootApplication.

Adnotacja @SpringBootApplication

- Umieszczona zwykle na poziomie głównej klasy w aplikacji
- Równoważna trzem innym adnotacjom:
  - @Configuration
  - @EnableAutoConfiguration
  - @ComponentScan
- Możemy to zobaczyć dzięki opcji `Open Declaration` w Eclipse.
- Te adnotacje są częścią frameworka Spring
- Aby nie wchodzić zbytnio w szczegóły działania samego Springa skupimy się na drugiej i trzeciej.

Adnotacja @ComponentScan

- 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
- daje dużo możliwości, m.in. pozwala zdefiniować, które komponenty powinny zostać pominięte - my skupimy się na przypadku bazowym

Adnotacja @EnableAutoConfiguration

- Wprowadza do działania system automatycznej konfiguracji
- Celem jest dokonanie przez framework automatycznej konfiguracji aplikacji na podstawie zawartości *classpath*
- 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 3: Adnotacje @ComponentScan i @EnableAutoConfiguration

Spring Boot Starters

- 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.

Spring Boot Actuator

- 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 nie jest).
- Endpoint bazowy: `/actuator`
- Rozbudowane narzędzie, tutaj zaprezentowane tylko podstawy.
- Przykład 4: użycie Spring Boot Actuator na prostej aplikacji.

Spring Boot Actuator

Niektóre *endpointy*:

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

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

Struktura projektu i dobre praktyki