SPRING I SPRING BOOT

Spring to popularny framework aplikacji webowych i enterprise dla języka Java. Framework ten zapewnia bogaty zestaw narzędzi do tworzenia aplikacji o różnym stopniu skomplikowania. Wśród głównych cech Spring można wymienić wstrzykiwanie zależności, aspektową orientację programowania, obsługę transakcji, konfigurację i zarządzanie ziarnami (beanami) oraz wiele innych.

Spring Boot jest z kolei narzędziem, które umożliwia szybkie tworzenie aplikacji opartych o Spring, dzięki uproszczeniu konfiguracji i automatycznemu dostarczaniu wielu funkcjonalności, takich jak serwowanie statycznych zasobów, tworzenie RESTful Web Services, obsługa baz danych itp. Spring Boot wprowadza również wbudowany serwer aplikacji (np. Tomcat, Jetty), co umożliwia uruchamianie aplikacji w prosty sposób.

RÓŻNICE:

Spring dostarcza szereg narzędzi i modułów do różnych zadań, takich jak integracja baz danych, bezpieczeństwo i testowanie. Z kolei Spring Boot dostarcza zestaw standardowych i automatycznych featerów/funkcjonalności, dzięki którym dużo łatwiej zacząć pracę ze Spring Bootem niż ze Springiem

Spring wymaga znacznej ilości konfiguracji, aby rozpocząć projekt, podczas gdy Spring Boot w znacznym stopniu automatyzuje konfigurację, pozwalając programistom na szybkie rozpoczęcie pracy.

Spring Boot zawiera wbudowany serwer WWW, co ułatwia tworzenie i wdrażanie aplikacji internetowych bez konieczności stosowania zewnętrznego serwera WWW.

Spring Boot zapewnia szereg funkcji ułatwiających tworzenie mikroserwisów, takich jak wykrywanie usług, równoważenie obciążenia i scentralizowana konfiguracja.

Podsumowując, Spring i Spring Boot to narzędzia, które zapewniają wiele funkcjonalności dla tworzenia aplikacji w języku Java. Spring to bardziej rozbudowany framework, który wymaga bardziej skomplikowanej konfiguracji, natomiast Spring Boot pozwala na szybkie i proste tworzenie aplikacji bez potrzeby ręcznej konfiguracji.

KONTENER IoC:

Odwrócenie sterowania (ang. Inversion of Control, IoC) – paradygmat polegający na przeniesieniu funkcji sterowania wykonywaniem programu do używanego frameworku. Framework w odpowiednich momentach wywołuje kod programu stworzony przez programistę w ramach implementacji danej aplikacji. Odbiega to od popularnej metody programowania, gdzie programista tworzy kod aplikacji, który

steruje jej zachowaniem. Następnie używa we własnym modelu sterowania bibliotek dostarczonych przez framework.

Kontener IoC pozwala na przechowywanie obiektów i zarządzanie nimi w całym cyklu życia aplikacji. Podejście to jest wykorzystywane przez wiele frameworków – nie tylko w Springu. Największe korzyści płynące z jego stosowania to:

- nie musisz ręcznie tworzyć obiektów, kontener zrobi to za Ciebie,
- umożliwia wstrzykiwanie zależności – czyli dostarczanie obiektu dokładnie tam gdzie tego potrzebujesz.

POJO (Plain Old Java Object) – obiekty, będących zwyczajnymi obiektami Java, nie zaś obiektami specjalnymi, w szczególności Enterprise JavaBeans.

DEPENDENCY INJECTION

Dependency injection (wstrzykiwanie zależności) jest wzorcem projektowym używanym w programowaniu obiektowym w celu uzyskania luźnego wiązania pomiędzy komponentami aplikacji.

Luźne wiązanie oznacza, że komponenty są niezależne od siebie i omgą być wymieniane lub modyfikowane bez wpływu na funkcjonowanie aplikacji jako całości.

Aby skorzystać z dependency injection w Spring Boot'cie zazwyczaj definiuje się komponenty aplikacji jako beany zarzadzane przez Spring. Robi się to z wykorzystaniem adnotacji takich jak np. @Component, @Service, czy @Repository. Za tworzenie instacji tak oznaczonych klas i zarządzanie nimi jest odpowiedzialny kontener IoC.

MODUŁY SPRINGA

- Spring Boot Starter - ten starter jest podstawową zależnością Spring Boot i zawiera wiele innych starterów, które są potrzebne do tworzenia podstawowej aplikacji.
- Spring Boot Starter Web - ta zależność zawiera wszystko, co jest potrzebne do tworzenia aplikacji webowej w Spring Boot. Zawiera ona między innymi moduł Spring MVC, Apache Tomcat oraz wiele innych zależności.
- Spring Boot Starter Data JPA - ta zależność zawiera wszystko, co jest potrzebne do zarządzania persystencją danych przy użyciu technologii JPA (Java Persistence API).
- Spring Boot Starter Security - ta zależność zawiera wszystko, co jest

potrzebne do zapewnienia bezpieczeństwa aplikacji, w tym moduł Spring Security.
- Spring Boot Starter Test - ta zależność zawiera wiele narzędzi do testowania aplikacji w Spring Boot, w tym moduł JUnit, Mockito i wiele innych.
- Spring Boot Actuator - ta zależność zawiera moduł Spring Boot Actuator, który umożliwia monitorowanie i zarządzanie aplikacją w czasie rzeczywistym.

Oprócz wymienionych powyżej zależności, istnieje wiele innych, takich jak Spring Boot Starter Batch, Spring Boot Starter Cloud, Spring Boot Starter Integration, które umożliwiają korzystanie z dodatkowych funkcjonalności w Spring Boot.

INTEGRACJA Z INNYMI TECHNOLOGIAMI

- Bazy danych - Spring Boot oferuje integrację z wieloma bazami danych, takimi jak MySQL, PostgreSQL, Oracle, MongoDB i wiele innych. Można to osiągnąć poprzez odpowiednie konfiguracje i dodanie odpowiednich zależności.
- Technologie frontendowe - Spring Boot można zintegrować z wieloma technologiami frontendowymi, takimi jak Angular, React czy Vue.js. Można to osiągnąć poprzez odpowiednie konfiguracje, dodanie odpowiednich zależności oraz umieszczenie plików z kodem frontendowym w odpowiednim katalogu.
- Integracja z usługami chmurowymi - Spring Boot umożliwia łatwą integrację z usługami chmurowymi, takimi jak Amazon Web Services, Google Cloud czy Microsoft Azure. Dzięki temu można łatwo tworzyć i deployować aplikacje w chmurze.
- Mikroserwisy - Spring Boot jest często wykorzystywany do tworzenia mikroserwisów, a do zarządzania nimi można wykorzystać Spring Cloud. Spring Cloud oferuje narzędzia do rejestracji, konfiguracji, monitorowania i wiele innych funkcjonalności dla mikroserwisów.
- RESTful API - Spring Boot umożliwia tworzenie RESTful API, które mogą być wykorzystywane przez aplikacje frontendowe lub przez inne serwisy. Do tego celu można wykorzystać moduł Spring MVC, który oferuje obsługę żądań HTTP.
- Cache - Spring Boot oferuje integrację z różnymi mechanizmami cache, takimi jak Redis czy Ehcache (open-Sourcowe oparte o baze oprogramowanie cashowe. Zawiera magazyny pamięci i dysków, odbiorniki, moduły ładujące pamięć podręczną, interfejsy API RESTful i SOAP oraz inne bardzo przydatne funkcje.). Można to osiągnąć poprzez odpowiednie konfiguracje oraz dodanie odpowiednich zależności.
- Integracja z innymi frameworkami i bibliotekami - Spring Boot umożliwia integrację z wieloma innymi frameworkami i bibliotekami, takimi jak Hibernate, Thymeleaf, Jackson, czy Apache Camel. Dzięki temu można korzystać z różnych narzędzi i rozwiązań dostępnych w ekosystemie Javy.

ADNOTACJE

Adnotacje (annotations) są rodzajem metadanych, które dostarczają informacji o klasach, metodach polach, czy argumentach. W Javie adnotacje rozpoczynają się od symbolu @, po którym następuje nazwa adnotacji. Adnotacje mogą mieć wartości i parametry.

Adnotacje mogą być używane do calów takich jak:

- Dostarczanie dodatkowych informacji do kopilatora lub runtime'a
- Dostarczanie dodatkowych danych lub konfiguracji frameworków i bibliotek
- Automatyczne generowanie kodu lub dokumentacji
- Wymuszanie ograniczeń związanych z dostępem, walidacją i bezpieczeństwem

Podstawowe i najczęściej używane adnotacje w Spring Boot'cie:

- @SpringBootApplication - w połączeniu z kilkoma innymi adnotacjami służy do oznaczania głównej klasy aplikacji. Używa się jej w połączeniu z innymi adnotacjami, takimi jak @Configuration, @EnableAutoConfiguration, @ComponentScan
- @Controller - oznacza klasę jako kontroler obsługujący żądania HTTP. Podobna do adnotacji @Component, ale używa się jej konkretnie do klas kontrolerów.
- @Autowired - automatycznie wiąże beany ze sobą. Służy do wsrzykiwania zależności. W pokazanym przykładzie @Autowired wstrzykuje MyRepository do konstruktora MyService.