Spring, Spring Boot

Prowadzący:
Marcin Dziadoń
dziadonm@gmail.com

Co to jest Framework?

Framework - szkielet do budowy aplikacji. Definiuje on strukturę aplikacji oraz ogólny mechanizm jej działania, a także dostarcza zestaw komponentów i bibliotek ogólnego przeznaczenia.

Programista tworzy aplikację, rozbudowując i dostosowując poszczególne komponenty do wymagań realizowanego projektu, tworząc w ten sposób gotową aplikację.

Zalety

- Efektywność – tworzenie aplikacji z wykorzystaniem frameworków wymaga od programisty mniejszej ilości kodu do napisania.
- Poprawa jakości kodu – ponieważ frameworki są projektowane z myślą o elastyczności, posiadają one dobrą wewnętrzną organizację i logikę, którą narzucają aplikacji.
- Niezawodność – frameworki jako szkielety aplikacji są dobrze zaprojektowane i przetestowane.

Wady

- Złożoność – ze względu na swoją elastyczność oraz wykorzystywanie zaawansowanych koncepcji, opanowanie frameworków nie jest łatwe
- Wydajność – często ceną za elastyczną budowę jest niższa wydajność tworzonego oprogramowania


spring

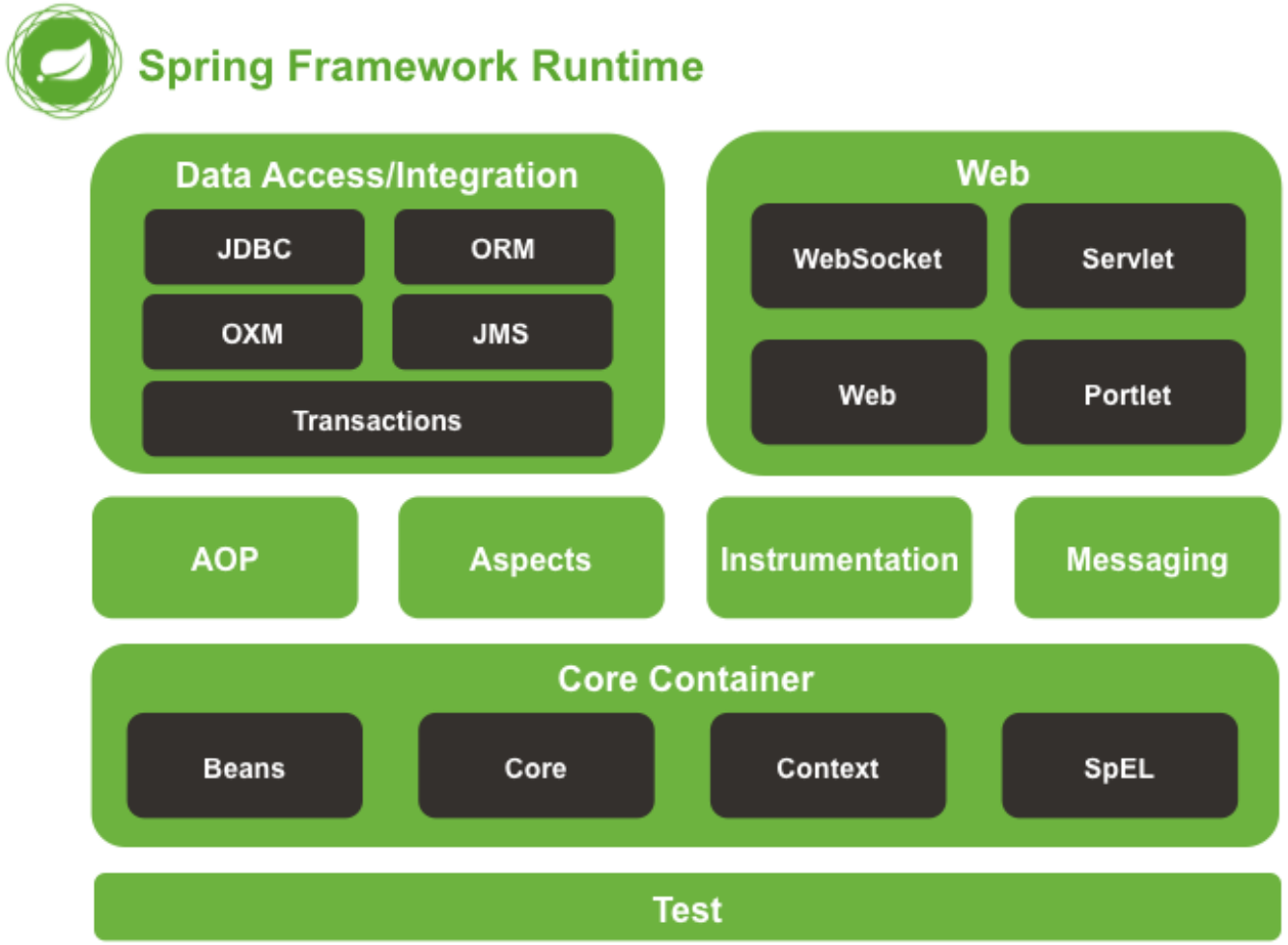
Spring

- Szkielet tworzenia aplikacji w języku Java
- Biblioteka dostępna na zasadach open source
- Alternatywa dla programowania aplikacji z użyciem EJB
- Uproszczenie programowania
- Bardzo rozbudowany moduł komponentów
- Kontener wstrzykiwania zależności
- Zmniejszenie stopnia powiązania klas między sobą

Zalety Springa

- Popularność – wiele ofert pracy wymaga znajomości Springa
- Jakość – przykład dobrze przetestowanego kodu
- Modularność – ponad 20 modułów
- Wykorzystanie najlepszych praktyk programowania
- Przystępny w nauce – bardzo dobra dokumentacja, duża społeczność użytkowników, dużo przykładów

Moduły Springa



IoC Container

- IoC *(inversion of control)* oraz DI *(dependency injection)* to dwa pojęcia określające to samo - przeniesienie odpowiedzialności za tworzenie obiektów z programisty na tzw. ”kontener”
- Programista definiuje tylko jakie obiekty, kiedy i gdzie mają być dostępne
- Fizycznym tworzeniem tych obiektów i dostarczaniem ich we właściwe miejsce zajmuje się kontener
- Konfiguracja Springa to proces informowania kontenera jakie obiekty będzie musiał utworzyć

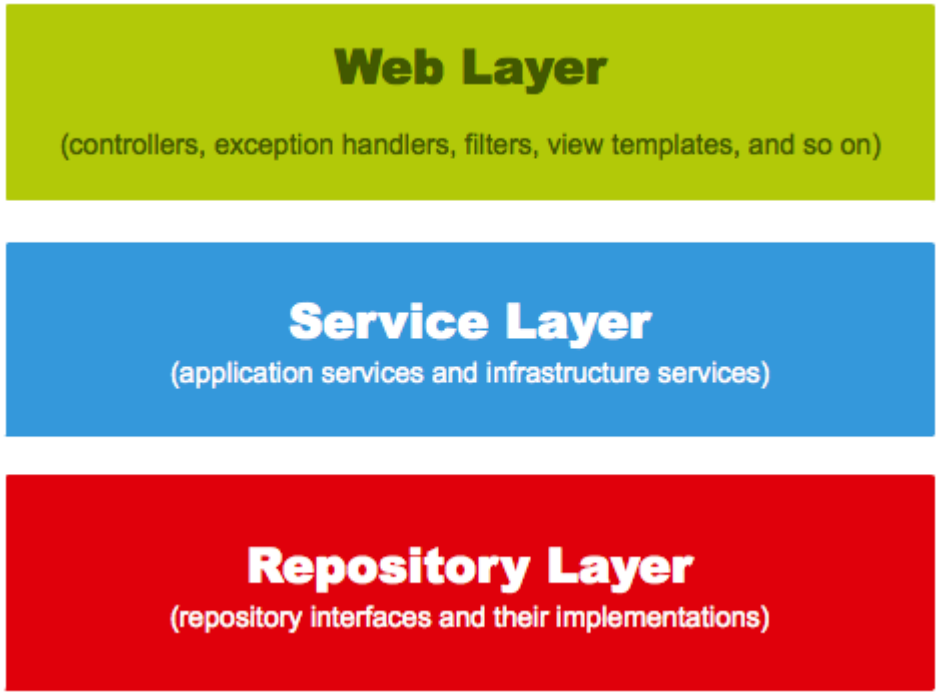
Application Context

- Centralny obiekt, w którym Spring przetrzymuje cała konfigurację - informacje o wszystkich zarządzanych przez siebie obiektach
- Istnieje kilka implementacji – wykorzystujemy odpowiednią w stosunku do sposobu konfiguracji
  - ClasspathXmlApplicationContext
  - AnnotationConfigApplicationContext
  - AnnotationConfigWebApplicationContext

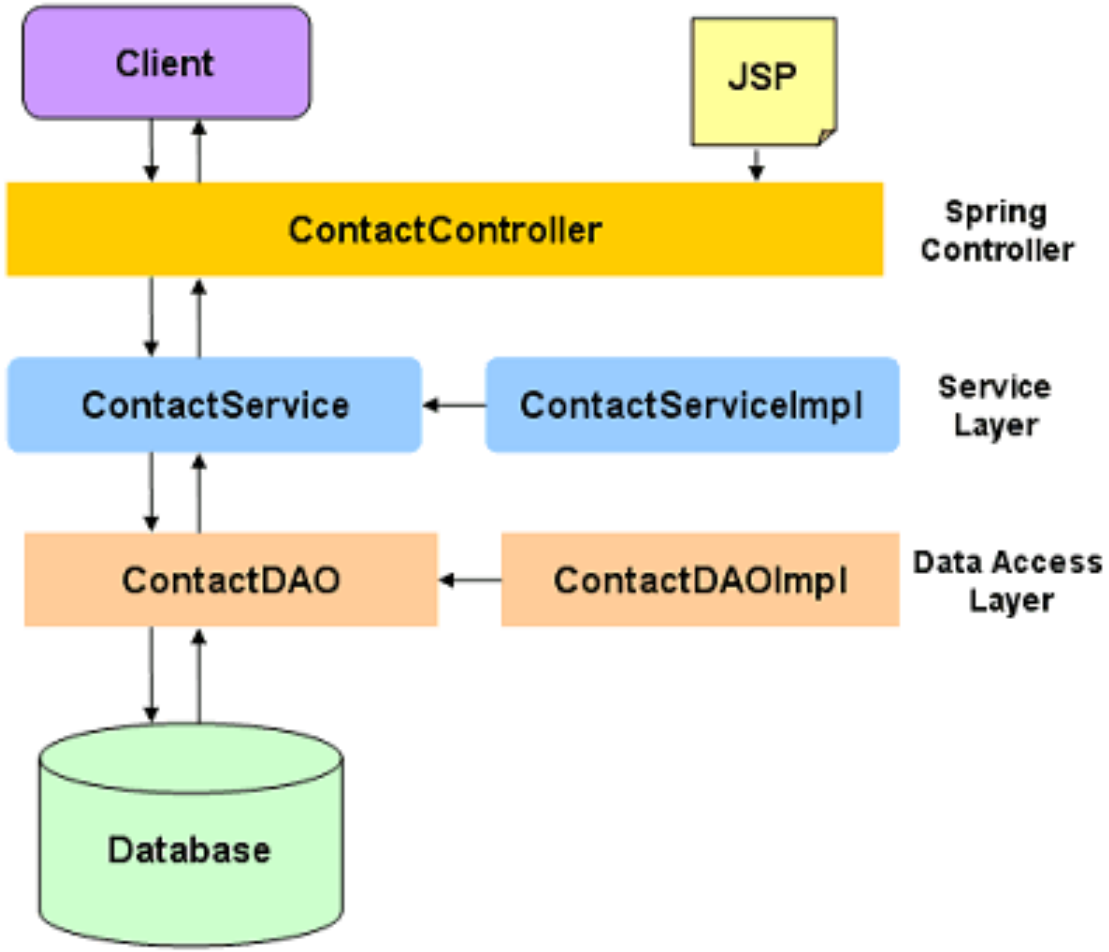
Architektura warstwowa

W świecie programowania obiektowego najbardziej rozpowszechnionym typem architektury aplikacji biznesowych jest architektura warstwowa. Można wyróżnić trzy podstawowe warstwy:

- warstwa prezentacji – odpowiedzialna za przyjmowanie żądań od klientów i zwracanie odpowiedzi
- warstwa usług – realizacja logiki biznesowej, zarządzanie transakcjami i zasobami
- warstwa dostępu do danych



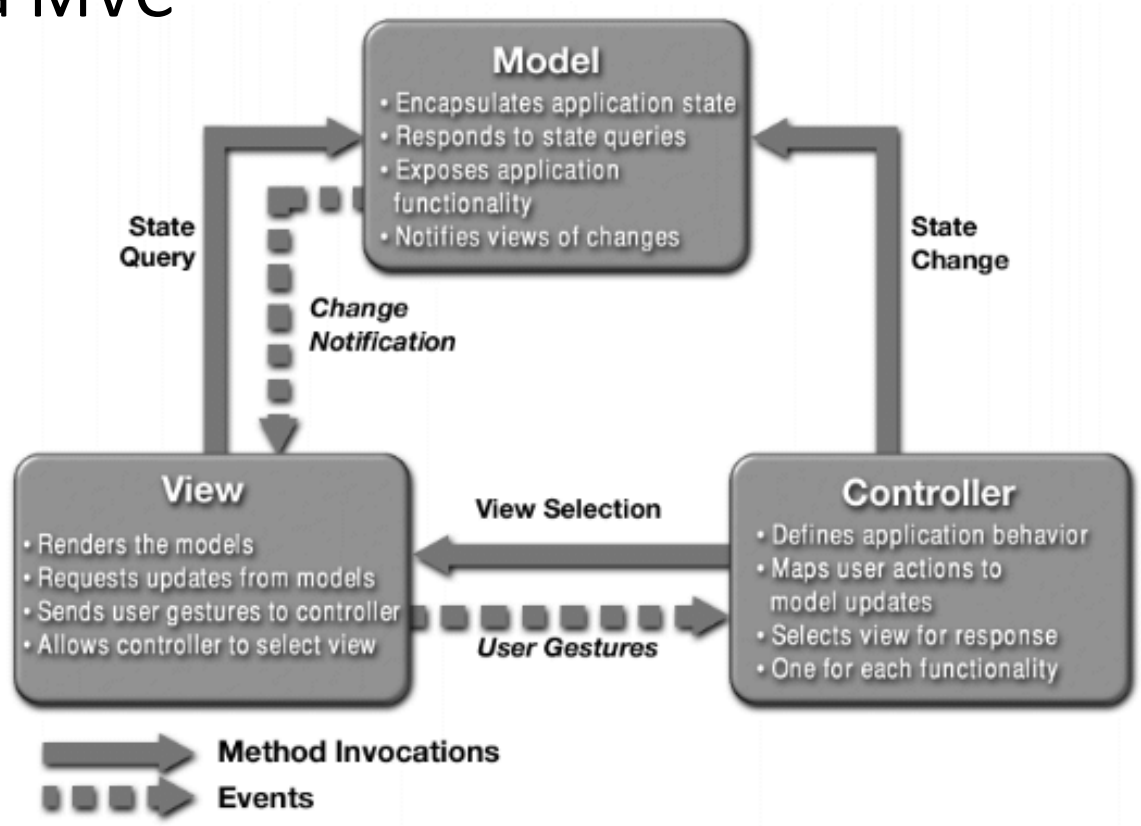
Przykładowa implementacja



Wzorzec MVC

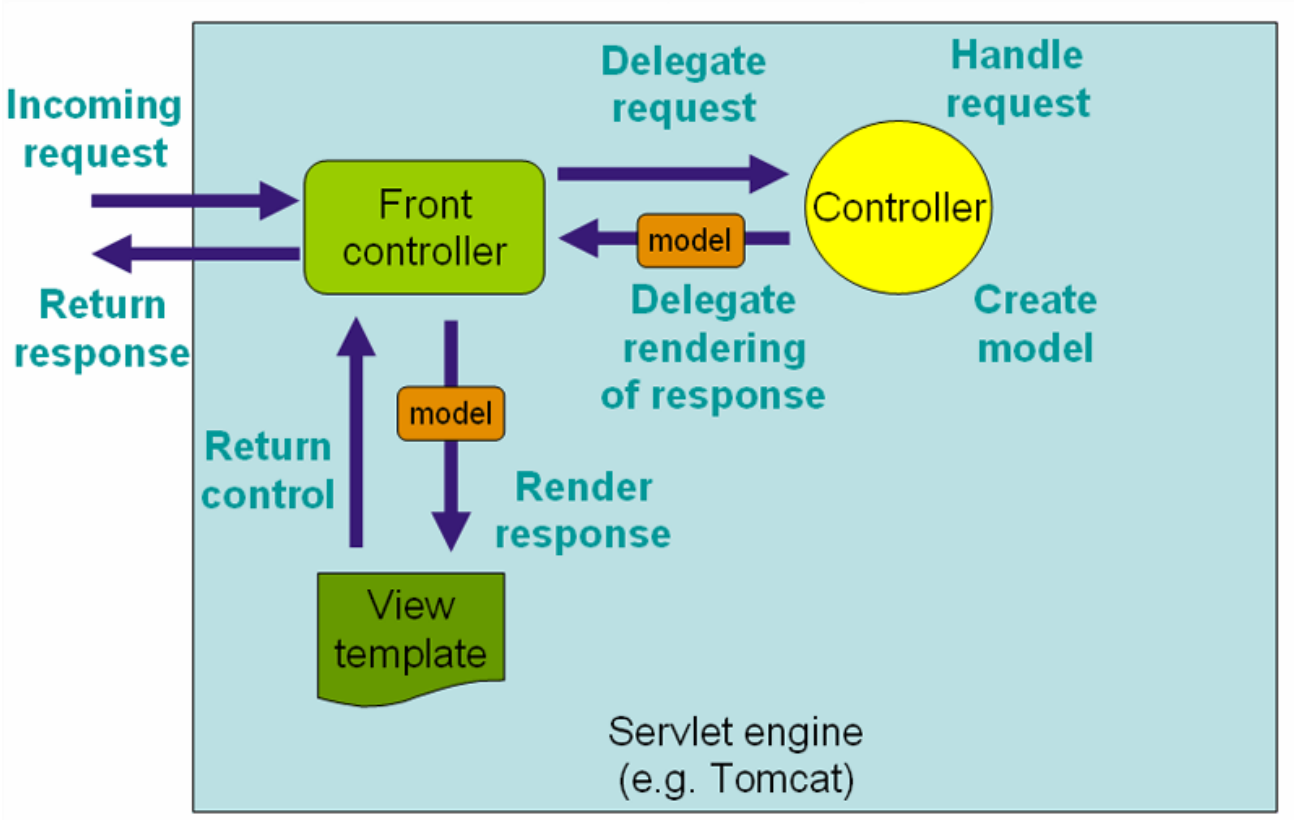
Warstwa prezentacji w systemach warstwowych jest najczęściej zaimplementowana za pomocą wzorca MVC

- Model – reprezentuje logikę aplikacji
- View – odpowiedzialny za prezentację modelu
- Controller – zarządza całym flow: odbiera żądania pobrania/modyfikacji modelu, deleguje do modelu, wybiera view



Cykl życia żądania w Spring MVC

1. **Żądanie**
2. **Serwlet dyspozytora** – deleguje odpowiedzialność za przetwarzanie żądania innym komponentom aplikacji
3. **Odwzorowanie obsługi** – podejmuje decyzję, do którego kontrolera wysłać żądanie
4. **Kontroler** – żądanie wypakowuje swój ładunek i czeka na przetworzenie informacji
5. **Model i logiczna nazwa widoku** – informacje zwrotne dla użytkownika przekazywane są w tzw. modelu
6. **Producent widoków** – przekształcenie logicznej nazwy widoku na konkretną implementację
7. **Widok** – informacja zwrotna dla użytkownika w przyjaznej formie np. html



Podstawowe założenia

- Używamy POJO (plain old java object)
- Framework dostarcza kontener dla obiektów
- Framework zarządza tworzeniem obiektów
- Większość naszego kodu "nie wie" o istnieniu kontenera
- Duża elastyczność i konfigurowalność
- Integracja i unifikacja dostępu do zewnętrznych zasobów i technologii (np. Baz danych)
- Projektowanie i programowanie przez interfejsy

Spring Bean

- Wszystko jest beanem!
- Beany to dowolne obiekty zarządzane przez kontener Springa.
- Beany nie muszą implementować żadnego interfejsu, ani rozszerzać żadnej klasy
- Beany zazwyczaj produkowane są przez fabrykę Springa.
- Beany definiujemy w kontenerze Springa na dwa sposoby w XML lub za pomocą adnotacji.

Konfiguracja XML

- Sposób konfiguracji aplikacji springowej, który był dostępny od początku
- Informacje o wszystkich bean-ach i ich powiązaniach programiści umieszczają w jednym lub kilku plikach XML
- Spring odczytuje dane z plików XML i na tej podstawie tworzy odpowiednie obiekty

```
<bean id="exampleBean" class="com.byteslounge.spring.ExampleBean">
        <property name="injectedBean" ref="injectedBean"/>
</bean>
```

Definiowanie komponentów za pomocą adnotacji

Uwalnia programistów od tworzenia plików XML, wszystkie informacje dla kontenera zawieramy w klasach Javy, głównie za pomocą adnotacji:

- @Component – klasa jest komponentem Springa
- @Controller – klasa definiuje kontroler MVC w Springu
- @Repository – klasa definiuje repozytorium danych
- @Service – klasa definiuje usługę

Spring widząc że klasa ma ustawioną powyższa adnotację wie że z danej klasy musi utworzyć beana i nadać mu id takie jak nazwa klasy tylko małą literą.

Cykl życia

- **Instancjonowanie** – po pierwsze Spring kontener znajduje definicję Beana w pliku XML i instancjonuje Bean.
- **Wypełnianie właściwości** – używając wstrzykiwania zależności, Spring uzupełnia wszystkie właściwości określone w definicji Beana.
- **Ustawianie nazwy Beana** – Jeżeli Bean implementuje interfejs BeanNameAware, Spring ustawia id Beana za pomocą metody setBeanName().
- **Ustawianie fabryki Beana**– Jeśli Bean implementuje interfejs BeanFactoryAware, Spring ustawia beanfactory za pomocą metody setBeanFactory().
- **Wstępna inicjalizacja** – zwany także post-procesowaniem Beana. Jeśli są jakiś BeanPostProcessors związany z Beanem, Spring wywołuje metodę postProcessBeforeInitialization().
- **Inicjalizacja Beana** – Jeżeli bean realizuje interfejs InitializingBean, jego metoda afterPropertySet() jest wywoływana. Jeśli Bean posiada zdeklarowaną metodę inicjującą to ta metoda jest wywoływana
- **Post inicjalizacja** – Jeśli jest jakiś BeanPostProcessors związany z Beanem, jego metoda postProcessAfterInitialization() zostanie wywołana.
- **Gotowy do użycia** – Teraz Bean jest gotowy do użycia przez aplikację.
- **Destroy** – Jeżeli Bean implementuje DisposableBean będzie wywołana metoda destroy()

Zasięg Bean

Domyślnie wszystkie komponenty są singletonami. Bean o zasięgu singleton gwarantuje wyłącznie powstanie jednej instancji określonej definicji komponentu w danym kontekście aplikacji.

Możliwe są też inne zakresy komponentów:

- Singleton – ogranicza zasięg definicji komponentu do jednej instancji w kontenerze
- Prototype – pozwala na utworzenie dowolnej liczby instancji komponentu
- Request – ogranicza zasięg definicji komponentu do żądania http
- Session – ogranicza zasięg definicji komponentu do sesji http
- Global-session - ogranicza zasięg definicji komponentu do globalnej sesji http. Dopuszczalne użycie tylko w kontekście portletów.

Wstrzykiwanie zależności

- Użytkownik nie tworzy i nie łączy obiektów w aplikacji, lecz opisuje zależności między poszczególnymi klasami. Kontener jest w pełni odpowiedzialny za tworzenie obiektów i ich łączenie.
- Do wstrzykiwania zależności służy adnotacja @Autowired

```
public class UserServiceImpl implements UserService {

    private UserDAO userDAO;

    public UserServiceImpl(UserDAO userDAO) {
        this.userDAO = new UserDAO();
    }

    public void update(User user) {
        userDAO.update(user);
    }
}
```

```
@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserDAO userDAO;

    public void update(User user) {
        userDAO.update(user);
    }
}
```


spring
boot

Spring Boot

- Ułatwia tworzenie aplikacji opartych na Spring
- Bierze odpowiedzialność za dołączenie niezbędnych komponentów
- Większość aplikacji opartych na Spring Boot wymaga minimalnej konfiguracji
- Dostarcza skonfigurowaną aplikację ale pozwala na własną konfigurację w zależności od potrzeb
- Zapewnia szeroki zakres funkcji, konfiguracji, które są wspólne dla wielu grup projektów
- Nie wymaga konfiguracji xml

Spring vs Spring Boot