

# Introduction to Spring Framework



# Agenda



## Java – Introduction to Spring framework

23.10.2018

Tomasz Miler  
Bartosz Niesobski



## Java – Microservices with Spring Boot

06.11.2018

Tomasz Miler  
Bartosz Niesobski



## Technologie Cloudowe w praktyce

20.11.2018

Piotr Mścichowski  
Tomasz Smelcerz



## Wieczorek pod Chmurą

04.12.2018

Mateusz Szostok  
Tomasz Pietrek  
Tomasz Heflik



## Scrum Master – The journey towards mastery

26.11.2018

Michał Drzewiecki  
Mariusz Jasiński



## Scripting vs. Compiled, Dynamic vs. Static – in JVM world

11.12.2018

Karol Grzyb  
Tomasz Miler

# SAP Labs Poland

**Top ecommerce,  
marketing, billing**

**Development:** Go, Java,  
Cloud Native solutions



**> 400 pracowników**

**Najlepszy Pracodawca  
2017 w rankingu AON**

**Jedno z 20 centrów  
SAP's Labs Network**

# Kim jesteśmy?



Tomasz Miler

Developer SAP CX



Bartosz Niesobski

Developer SAP CX

## Czym właściwie 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 do wykonywania określonych zadań.”

~Wikipedia

# Przykładowe frameworki



## IoC – Inversion **of** control

**„Paradygmat polegający na przeniesieniu funkcji sterowania wykonywaniem programu do używanego framework. Framework w odpowiednich momentach wywołuje kod programu stworzony przez programistę w ramach implementacji danej aplikacji”**

~Wikipedia

## DI - Dependency injection

„Wzorzec architektury oprogramowania polegający na usuwaniu bezpośrednich zależności pomiędzy komponentami na rzecz architektury typu plug-in. Polega na przekazywaniu gotowych, utworzonych instancji obiektów udostępniających swoje metody i właściwości obiektem, które z nich korzystają”

~Wikipedia

# Deklarowanie zależności bez IoC

```
public class Foo {  
    private final FooService fooService;  
    private final BarService barService;  
  
    public Foo() {  
        fooService = new FooService();  
        barService = new BarService();  
    }  
}
```

# Deklarowanie zależności z IoC

```
public class Foo {  
    private FooService fooService;  
    private BarService barService;  
  
    public void setFooService(final FooService fooService) {  
        this.fooService = fooService;  
    }  
  
    public void setBarService(final BarService barService) {  
        this.barService = barService;  
    }  
}
```

## Spring bean

**Bean jest definicją obiektu, która jest tworzona, konfigurowana oraz zarządzana przez kontener IoC w Spring framework.**

~We

# Spring bean

```
@Bean
public FooClass getFooClass() {
    return new FooClass();
}

@Component
public class Foo {
    // ...
}

@Controller
public class FooController {
    // ...
}

@Service
public class FooService {
    // ...
}
```

# Spring bean

```
<beans>

    <bean id="xmlBean" class="java.lang.String">
        <constructor-arg value="xmlBean" />
    </bean>

    <alias name="xmlBean2" alias="bean2"/>
    <bean id="xmlBean2" class="xyz.foo.bar.MyFoo">
        <property name="myString" ref="xmlBean"/>
    </bean>

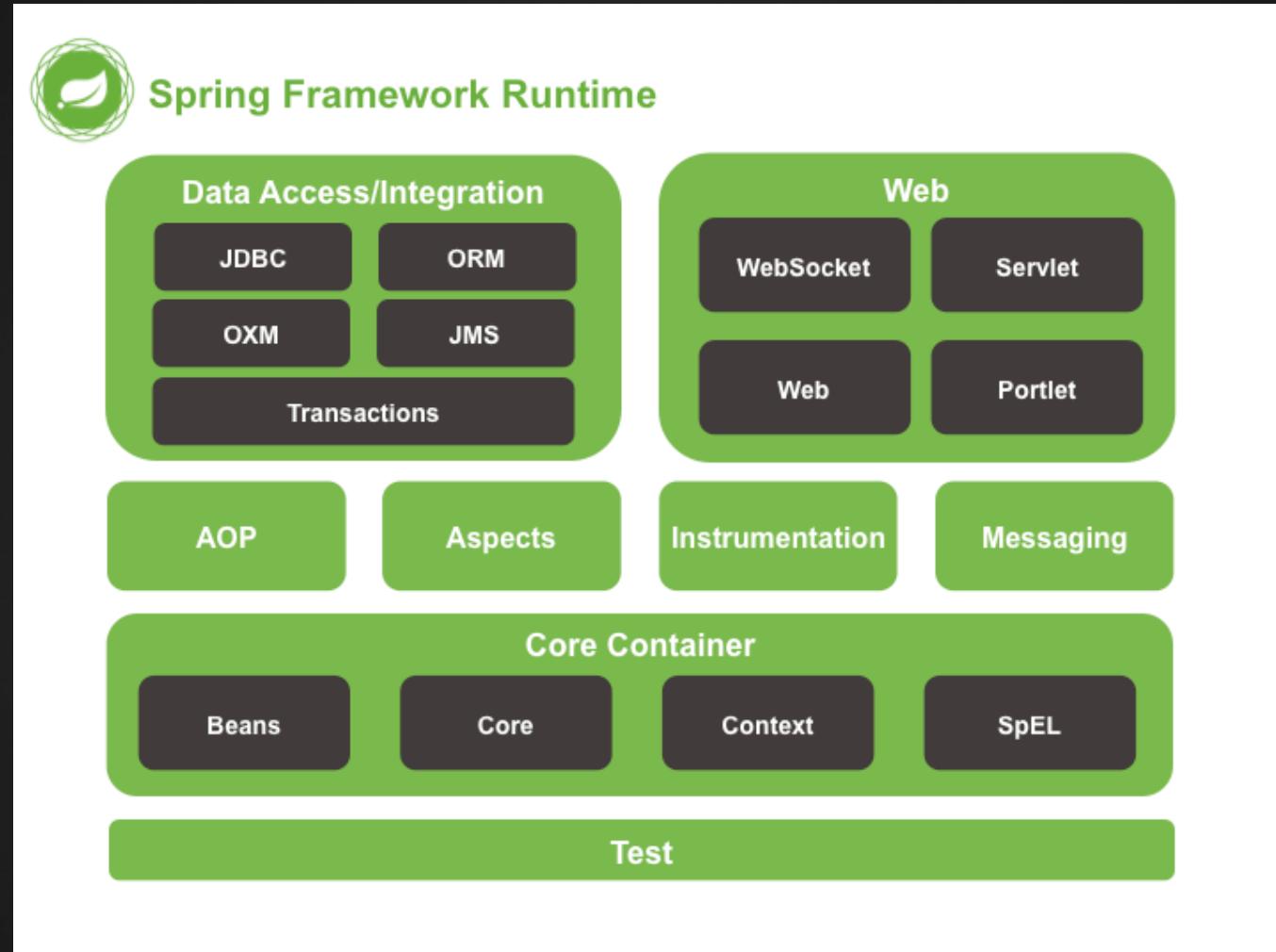
</beans>
```

# Jakie problemy rozwiązuje podejście IoC oraz DI?



- **Rodziela komponenty i warstwy w aplikacji**
- **Komponenty nie są odpowiedzialne za zarządzanie ich zależnościami**
- **Pozwala na szybką zmianę implementacji w różnych środowiskach (dev, test)**
- **Pozwala na testowanie komponentów poprzez mokowanie ich zależności**
- **Dostarcza mechanizm do dzielenia się zasobami w aplikacji**

# Spring framework modules

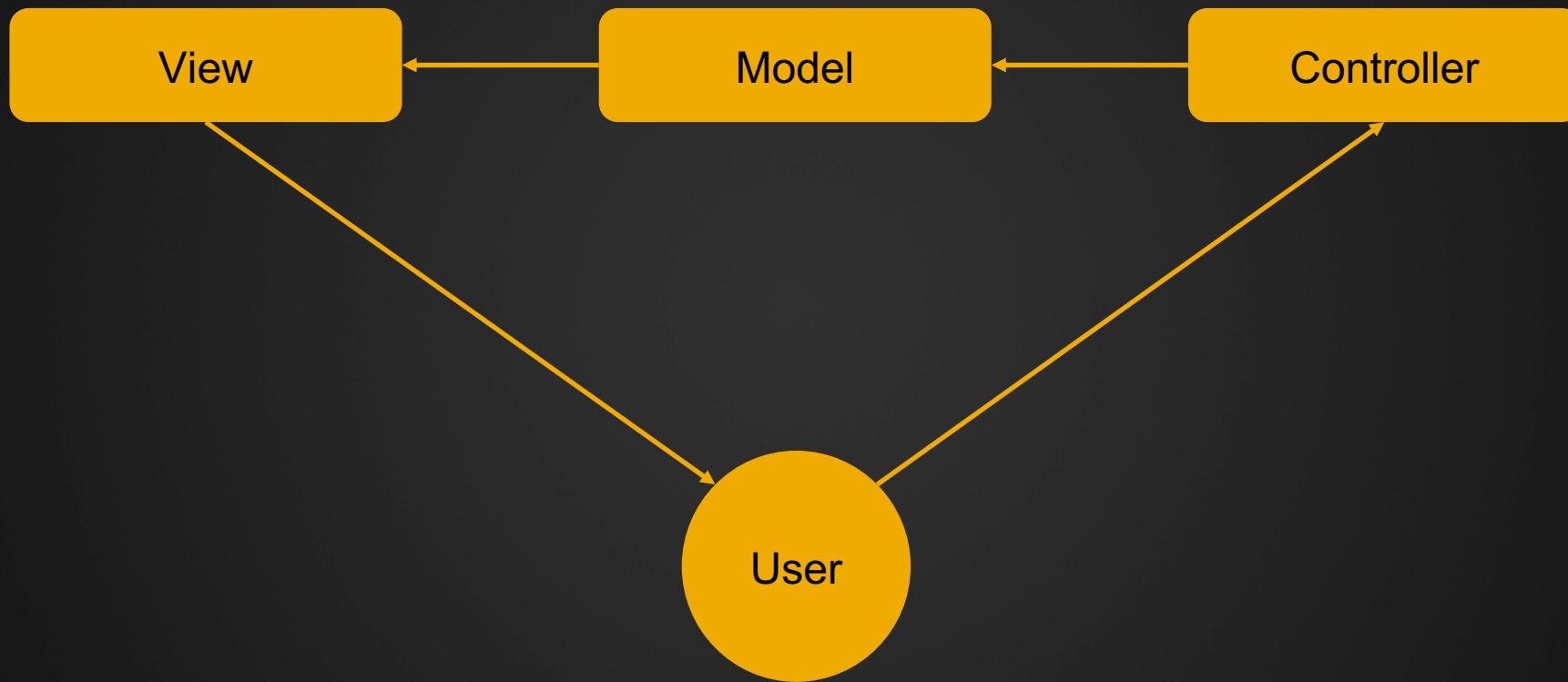


## MVC - Model-**View**-Controller

„Wzorzec architektoniczny służący do organizowania struktury aplikacji posiadających graficzne interfejsy użytkownika”

~Wikipedia

# MVC - Model-View-Controller



# Zalety Spring framework'a

- **Rozszerzalność**
- **Modularność**
- **Elastyczność konfiguracji**
- **Łatwość testowania**
- **Dobra architektura**
- **Brak potrzeby zewnętrznego serwera (np. tomcat)**
- **SpEL – Spring expression language**

# Wady Spring framework'a

- **Złożoność**
- **Wysoki próg wejścia**
- **Pamięciożerny**
- **Ciężko się debuguje**



„Spring Boot ułatwia tworzenie samodzielnych aplikacji, opartych o Spring framework, które można "po prostu uruchomić". (...) dzięki czemu możesz zacząć od minimalnej ingerencji. Większość aplikacji Spring Boot wymaga bardzo niewielkiej konfiguracji Spring.”

~Dokumentacja Spring boot

# Jak zacząć?

<http://start.spring.io>

SPRING INITIALIZR bootstrap your application now

Generate a  with  and Spring Boot

**Project Metadata**

Artifact coordinates

Group

Artifact

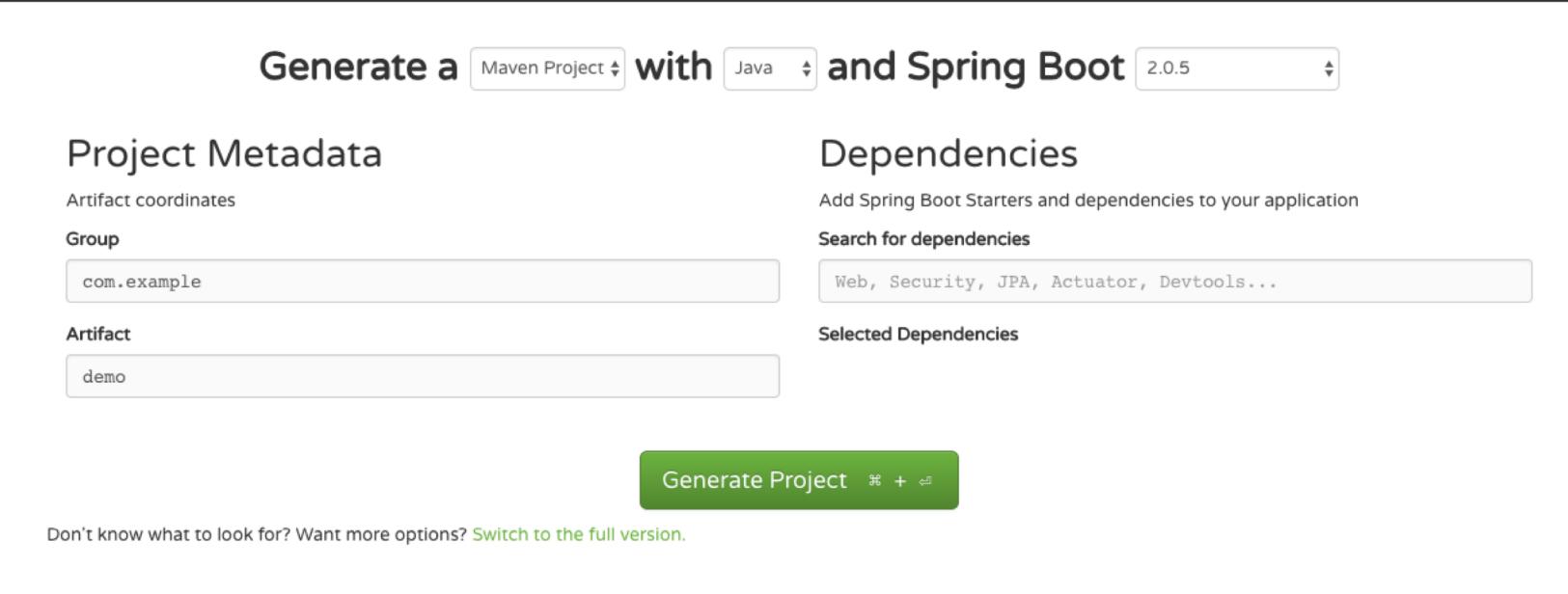
**Dependencies**

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Selected Dependencies

Don't know what to look for? Want more options? [Switch to the full version.](#)



# Demo

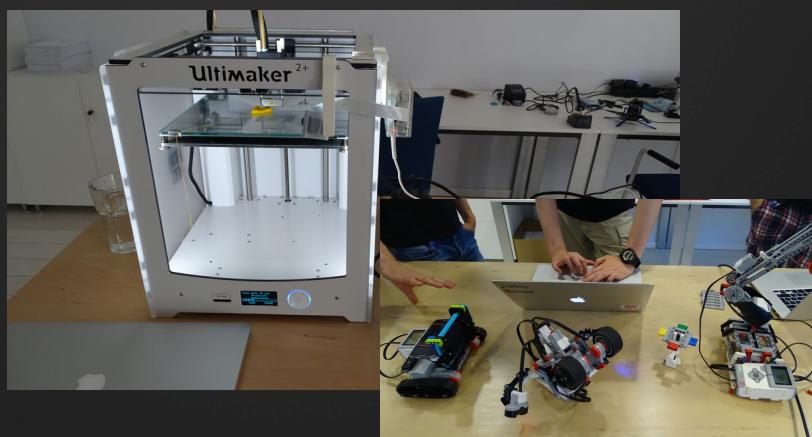
# Kogo szukamy?

## Studentów

- **Students in Development Teams**  
*(Support engineers, Product Development, Open Source)*
- **Hack Team (recruitment starts in April)**  
*(3 months internship, 2 teams realizing dedicated projects)*

## Profesjonalistów

- **Software Developers**  
*(Cloud Native Solutions, Go, Java)*
- **DevOps Engineers**  
*(Kubernetes, Cloud Foundry)*
- **Support Engineers**  
*(Java Spring)*



# Co dalej?

## Opcja 1

- Zapraszamy na zwiedzanie biura



## Opcja 2

- Q&A z naszymi prowadzącymi

# Thank You

