JAVA SPRING CURS1: INTRODUCERE

INVERSION OF CONTROL

DPENDENCY INJECTION

Spring overview

- JED (java enterprise development) simplificat (mai simplu de utilizat decât J2EE sau Java EE)
- Lightweight development POJOs vs EJB
- DI/IoC
- AOP aspect oriented programming (exemplu aspect transaction management)
- Reduce "*boilerplate code"* cod refolosit de mai multe ori

ISTORIC

- 2004 Spring 1.0 J2EE (beans)
- 2006 Spring 2.0
- 2009 Spring 3.0
- 2013 Spring 4.0
- 2016 Spring 4.3
- 2017 (septembrie) -
  - **Spring 5.0**

SPRING 5.0

- 2004 Versiuni Java acceptate: minim JAVA 8
- 2006 Spring MVC folosește Servlet API 4.0
- 2009 new feature: Spring WebFlux: reactive programming

SPRING FRAMEWORK

SPRING CORE CONTAINER

- Beans
- Core
  - “bean factory”Crează bean-uri, manageriază dependențele dintre bean-uri
  - Citește fișiere de configurare/adnotări pentru a seta proprietăți și dependețe
- SpEL Spring Expression Language
- Context containerul unde sunt memorate bean-urile

Infrastructure
AOP aspect oriented programming

- Adaugă funcționalități obiectelor
- Servicii pentru logging, securitate, tranzacții
- Instrumentation: JMX Java management extension monitorizează remote aplicația

Integration
Data acces layer

- Asigură comunicarea cu baza de date
- JDBC helper classes
- ORM object to relation mapping: Hibernate și JPA
- JMS Java message service helper classes: asigură trimiterea de mesaje asincrone

Web layer

- Spring MVC: controllers, views
- WebSocket
- Servlet
- Portlet

Test layer

- Suport pentru TDD, test-driven-development
- Unit tests/integration tests
- Suport pentru mocking și teste out-of-container

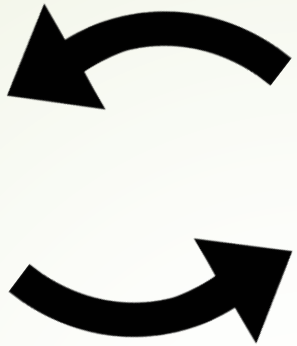
SPRING PROJECTS

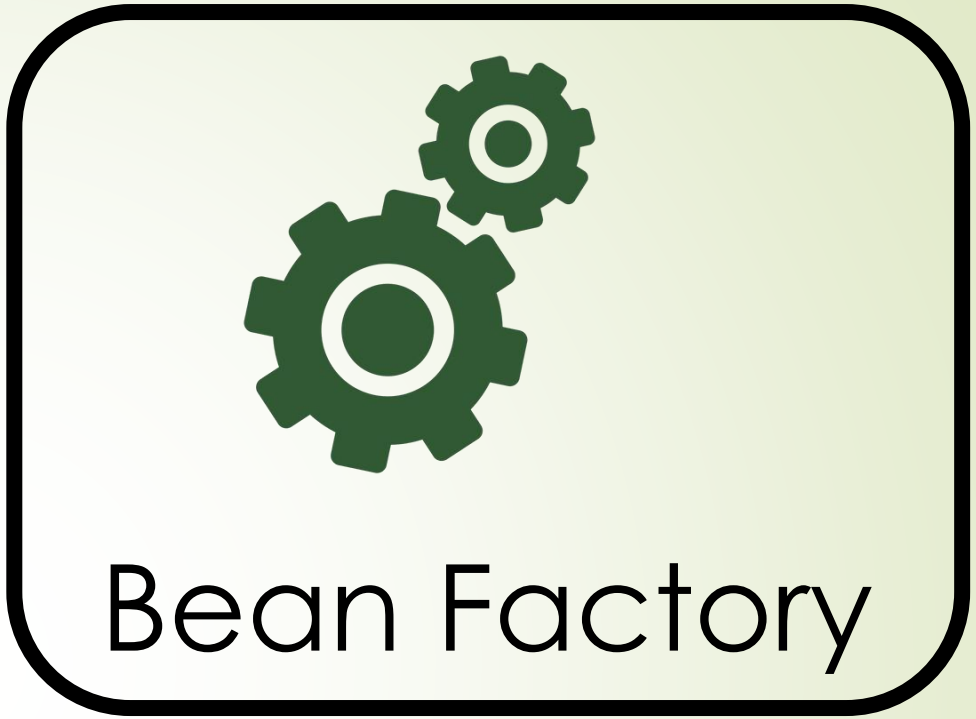
Spring projects overview

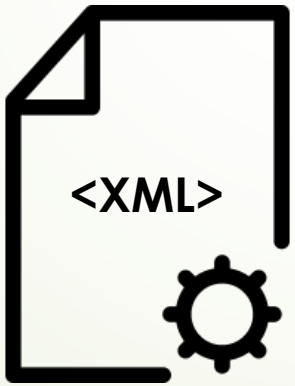
- Module adiționale, add-ons
- Spring Cloud,
- Spring Data,
- **Spring Security**
- Spring Web Flow
- **Spring Web Services (REST**/SOAP)
- Spring LDAP
- Spring Social
- **Spring Boot**

IoC INVERSION OF CONTROL

App

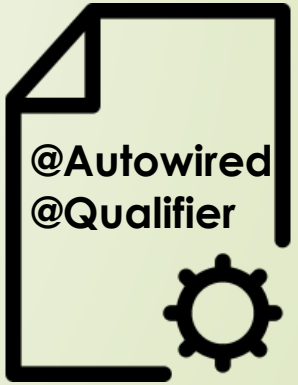






Configuration

Annotations



- Configurare Beans
  - **Fișiere xml**,
  - adnotări,
  - cod Java

  Beans = Obiecte Java instanțiate, “asamblate” si administrate de Spring IoC Container.

```
<bean id="mySubscription"
      class="com.apbdoo.lab1.BooksSubscription">
</bean>
```

https://docs.spring.io/spring/docs/current/spring-framework-reference/core.html#beans-introduction

- Crearea unui container Spring sau ***BeanFactory***, instanțierea interfeței **ApplicationContext** care oferă metode pentru crearea și accesarea/utilizarea componentelor aplicației.
- Interfața ApplicationContext este o specializare a intefeței ***BeanFactory.***
- Exemple de implementări:
  - ClassPathXmlApplicationContext
  - AnnotationConfigApplicationContext
  - GenericApplicationContext
  - GenericWebApplicationContext

```
ClassPathXmlApplicationContext context =
        new ClassPathXmlApplicationContext("applicationContext.xml");
```

- Folosirea Bean-urilor administrate de container.

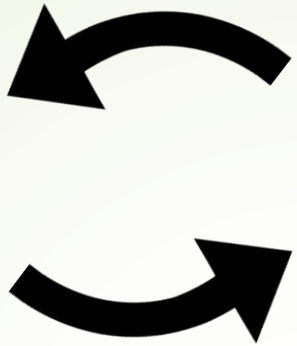
```
// retrieve bean from spring container
Subscription theSubscription = context.getBean("mySubscription",  Subscription.class);

// call methods on the bean
System.out.println(theSubscription.getPrice() + "   " + theSubscription.getDescription());

// close the context
context.close();
```

DEPENDENCY INJECTION

App



DI Bean

Constructor/
Setter /
Annotations

- Dependență = **helper class**
- DI delegarea provizionării dependențelor unei entități externe
- Există Constructor DI și Setter DI

- **Constructor DI**

```
DiscountCalculator discountCalculator;
public MoviesSubscription(DiscountCalculator discountCalculator)
{
    this.discountCalculator = discountCalculator;
}
```

- **Constructor DI**

```
DiscountCalculator discountCalculator;
public MoviesSubscription(DiscountCalculator discountCalculator)
{
    this.discountCalculator = discountCalculator;
}

<bean id="mySubscription"
    class="com.apbdoo.lab1.MoviesSubscription">
    <constructor-arg ref="myDiscountCalculator"/>
</bean>
```

- **Setter DI**

```
DiscountCalculator discountCalculator;
public void setDiscountCalculator(DiscountCalculator discountCalculator)
{
        this.discountCalculator = discountCalculator;
}

<bean id="myBooksSubscription"
          class="com.apbdoo.lab1.BooksSubscription">
        <property name="discountCalculator" ref="myDiscountCalculator"/>
        <property name="description" value="${foo.description}"/>
</bean>
```

- **Annotation DI**

```
<context:component-scan base-package="com.apbdoo.lab1"/>

 @Autowired
 @Qualifier("testServiceImpl")
      public void setTestService(TestService testService) {
            this.testService = testService;
 }
```

BEANS LIFECYCLE

- Cât timp este păstrat obiectul în context? Cum sunt share-*uite* instanțele?

- Cât instanțe sunt create?
  - Default **singleton**: este creată o singură instanță pentru fiecare bean

```
<bean id="mySportSubscription"
      class="com.apbdoo.lab1.SportSubscription"
      scope = "singleton">
          …
</bean>
```

  - Alte valori posibile pentru scope: **prototype, request, session, global-session**

START CONTAINER

INSTANTIERE BEAN

SETARE PROPRIETATI DI

CUSTOM INIT METHOD

SHUTDOWN CONTAINER

CUSTOM DESTROY METHOD

- Configurare custom init/destroy method

```
<bean id="mySportSubscription"
      class="com.apbdoo.lab1.SportSubscription"
      init-method = "initSportSubscrition"
      destroy-method = "destroySportSubscrition" >
          …
</bean>
```

- Pentru bean-uri cu scope = prototype nu se apelează destroy method!

ANNOTATION CONFIGURATIONS

- Adnotările sunt etichete adăugate claselor Java
- Adaugă meta-date despre clase
- Simplifică configurațiile xml
- Sunt procesate la run-time sau la compilare
  - Exemplu @Override -- la compilare

IoC with addnotations

- Se activează în configurația xml **component scanning**

<context:component-scan base-package="com.apbdoo.lab2“ />

</beans>

- Se adnotează clasele cu **@Component** și opțional cu **@Scope**

- Se încarcă din Containerul IoC *bean*-urile i.e clasele adnotate @Component.

Defaul Bean Id

Auto wiring

- Se scanează contextul pentru a se găsi o instanță de tipul necesar a fi injectat în bean

Constructor DI auto wiring

```
@Autowired
public BooksSubscription(DiscountCalculator   discountCalculator) {
      this.discountCalculator = discountCalculator;
}
```

- Începând cu versiunea 4.3 adnotarea constructorilor @Autowired nu mai este obligatorie.

Setter DI auto wiring

```
@Autowired
public void setDiscountCalculator(DiscountCalculator discountCalculator) {
        this.discountCalculator = discountCalculator;
}
```

- Metoda @Autowired poate avea orice denumire. Nu este obligatoriu să se respecte convenția setAtribute

Field DI auto wiring. De ce nu?

@Autowired

discountCalculator

- Field DI: Este mai ușor de observat dacă este încălcat principiul S (single responsibility principle) din **SOLID**
- Field DI: Clasele devin inutilizabile în afara containerului Spring.

Field DI auto wiring. De ce nu?

- Field DI: Este mai ușor de observat dacă este încălcat principiul S (single responsibility principle) din **SOLID**
- Field DI: Clasele devin inutilizabile în afara containerului Spring.
- Field DI: Nu se poate folosi field DI pentru câmpuri final.
- SetterDI: pentru câmpuri opționale și care pot fi modifiate
- ConstructorDI: pentru câmpuri obligatorii

@Qualifier @Primary

- Dacă sunt găsite pentru o dependență mai multe implementări se obține eroarea org.springframework.beans.factory.UnsatisfiedDependencyException
- Pentru a se evita această eroare se va adnota una dintre aceste implementări @Primary
- Alternativ, se poate specifica cu adnotarea @Qualifier care dintre implementări va fi utilizată.

@Qualifier @Primary

- @Primary specifică implementarea care va fi injectată default.

```
@Component
@Primary
public class FixDiscount implements DiscountCalculator
```

@Qualifier @Primary

- @Qualifier specifică id-ul instanței (bean) care va fi injectată.

```
@Autowired
@Qualifier("discountCalculatorImpl")
public void setDiscountCalculator(DiscountCalculator
discountCalculator)
```

@Qualifier @Primary

- @Qualifier specifică id-ul instanței (bean) care va fi injectată.

```
@Autowired
@Qualifier("discountCalculatorImpl")
public void setDiscountCalculator(DiscountCalculator
discountCalculator)
```

JAVA CODE CONFIGURATIONS

- @Configuration specifică o clasă care va fi folosită pentru configurare
- @ComponentScan specifică pachetele care vor fi scanate pentru regăsirea componentelor

```
@Configuration
@ComponentScan("com.apbdoo.lab2")
public class SubscriptionConfig{  … }
```

- AnnotationConfigApplicationContext este o implementare ApplicationContext care utilizează pentru configurare o clasă Java în locul unui fișier xml.

```
@Configuration
@ComponentScan("com.apbdoo.lab2")
public class SubscriptionConfig{  … }
```