Spring Core

Herhaling

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Wat is Spring

- Open Source java Framework
- POJO's → herbruikbare componenten
- Gebruik voor Multitierapplicaties
  - Data laag → communicatie met database
  - Service laag → Business logica
  - Presentatie laag → UI
  - Domain objects → pojo's → Data objecten

Voordelen Spring

Lightweight: geen server nodig

DI → Loose coupling

AOP → Cross Cutting Concern

→ Func. niet tot kerntaak!

…

DI en IOC

Beans krijgen hun afhankelijkheden van buitenaf toebedeeld.

- **Property/field injection**
- **Setter injection**
- **Constructor injection**

Focus on Business

```
public Car getById(String id) {
    Connection con = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;

    try {
        String sql = "select * from CAR where ID = ?";
        con = DriverManager.getConnection("localhost:
3306/cars");
        stmt = con.prepareStatement(sql);
        stmt.setString(1, id);
        rs = stmt.executeQuery();
        if(rs.next()) {
            Car car = new Car();
            car.setMake(rs.getString(1));
            return car;
        }
        else {
            return null;
        }
    } catch (SQLException e) { e.printStackTrace();}
    finally {
        try {
            if(rs != null && !rs.isClosed()) {
                rs.close();
            }
        } catch (Exception e) {}
    }
    return null;
}
```

```
public Car findCar(String id) {
    return getEntityManager().find(Car.class, id);
}
```

Configuratie

- XML → applicationContext.xml
  - Configuratiebestand
  - Seperation of concerns
- Annotaties
  - Component scanner
  - Stereotype annotaties (Com, Ser, Con, Rep)
- Java
  - AppConfig.java → Configuratie klasse
  - @Bean methods

Spring Core

Nieuwigheden

Scopes

Singleton: stateless beans

Prototype: stateful bean

Lazy: Singleton bean die pas aangemaakt wordt tijdens gebruik.

Stateful vs Stateless

Stateful: informatie opslaan over wat er is gebeurd of veranderd sinds opstart.

Stateless: Elke keer zelfde response op eenzelfde request. Geen info bijhouden.

→ HTTP is stateless

Spring Expression Language

SpEL

(Vergelijkbaar met Expression Language in JSP)

#{expression}

SpEL: voorbeelden

```
@Value("#{'Joske'}")
private String name;

@Value("#{5}")
private int number;

@Value("#{5 < 4}")
private boolean myBoolean;
```

SpEL: voorbeelden

```
@Value("#{new java.io.File('hello.txt')}")
private File file

@Value("#{systemProperties['user.home']}")
private String home;

@Value("#{systemEnvironment['JAVA_HOME']}")
private String java;
```

Event Listener

Op de hoogte brengen van obj. (beans) tijdens relevante toestandsveranderingen.

BV:

User logt uit

User wisselt van taal

…

Event Listener

Eventklasse → Afleiden van ApplicationEvent

@EventListener

Implementeren van de methode in de specifieke beans.

ApplicationEventPublisher → publishEvent()

ApplicationContext → publishEvent()

Event Listeners: Spring

Event	*Omschrijving*
`ContextClosedEvent`	Wordt verstuurd zodra de `ApplicationContext` gesloten wordt met `close()`.
`ContextRefreshedEvent`	Wordt verstuurd zodra de `ApplicationContext` ververst wordt met `refresh()`.
`ContextStartedEvent`	Wordt verstuurd zodra de `ApplicationContext` gestart wordt met `start()`.
`ContextStoppedEvent`	Wordt verstuurd zodra de `ApplicationContext` gestopt wordt met `stop()`.
`RequestHandleEvent`	Wordt verstuurd door de `WebApplicationContext` zodra een verzoek afgehandeld wordt.

Internationalization

Ondersteuning van talen en lokale instellingen

Locale object → Locale.getDefault() of new Locale(“nl”, “BE”)

Taal: ISO-639 standaard (nl, fr, en)

Land: ISO-3166 standaard (BE, FR, US)

Internationalization

VM-argument:

-Duser.language=nl → Wijzigen van de taal

Resources →

- messages.properties → default
- messages_nl.properties → Nederlands
- messages_fr.properties → Frans
- …

Internationalization

AppConfig:

```
@Bean
public MessageSource messageSource(){
    ResourceBundleMessageSource rbms =
new ResourceBundleMessageSource();
    rbms.setBasename(“messages”);
    return rbms;
}
```

Internationalization

Opvragen:

In App:

context.getMessage(“hello”, null, Locale.getDefault());

In Bean:

messageSource.getMessage(“hello”, null, Locale.getDefault());

Aspect Oriented Programming

Cross Cutting Concern

→Bepaalde taken uitvoeren die niet tot de kerntaak van de bean behoort.

Bepaalde aspecten van objecten:

BV: Trainer → kerntaak is lesgeven

Andere aspecten: rapporteren aan bazin, cursussen bestellen …

AOP: begrippen

Advice: extra taak toevoegen

JoinPoint: mogelijke plaats voor toevoeging advice

Pointcut: Concrete plaats van uitvoering

Aspect: toepassing van advice op bep. Pointcut

AOP: begrippen

Target: Doelobject

Weaving: aspecten dienen verweven/ verbonden te worden met het programma.

- Compile time (compilatie)
- Classload time (inladen van de klasse)
- Runtime (tijdens uitvoering)

Spring AOP

Aspecten toevoegen aan publieke methoden van een bean.

Enkel van toepassing op objecten die in de Spring Container beheerd worden.

Gebruikt bepaalde technieken van AspectJ

Spring AOP: proxies

Geen ref. naar echt object maar naar proxy-object

= Op basis van interface

Geen interface → CGLIB (Code Generation Library)

Spring AOP: pointcuts

Enkel bij publieke methoden van bean:

- Before
- After
- After-throwing
- After-returning

Spring AOP: toepassen

pom.xml → aspectjweaver toevoegen

Appconfig.java → @EnableAspectJAutoProxy

MyAspect.java→ @Component @Aspect

Pointcut expression: voorbeeld

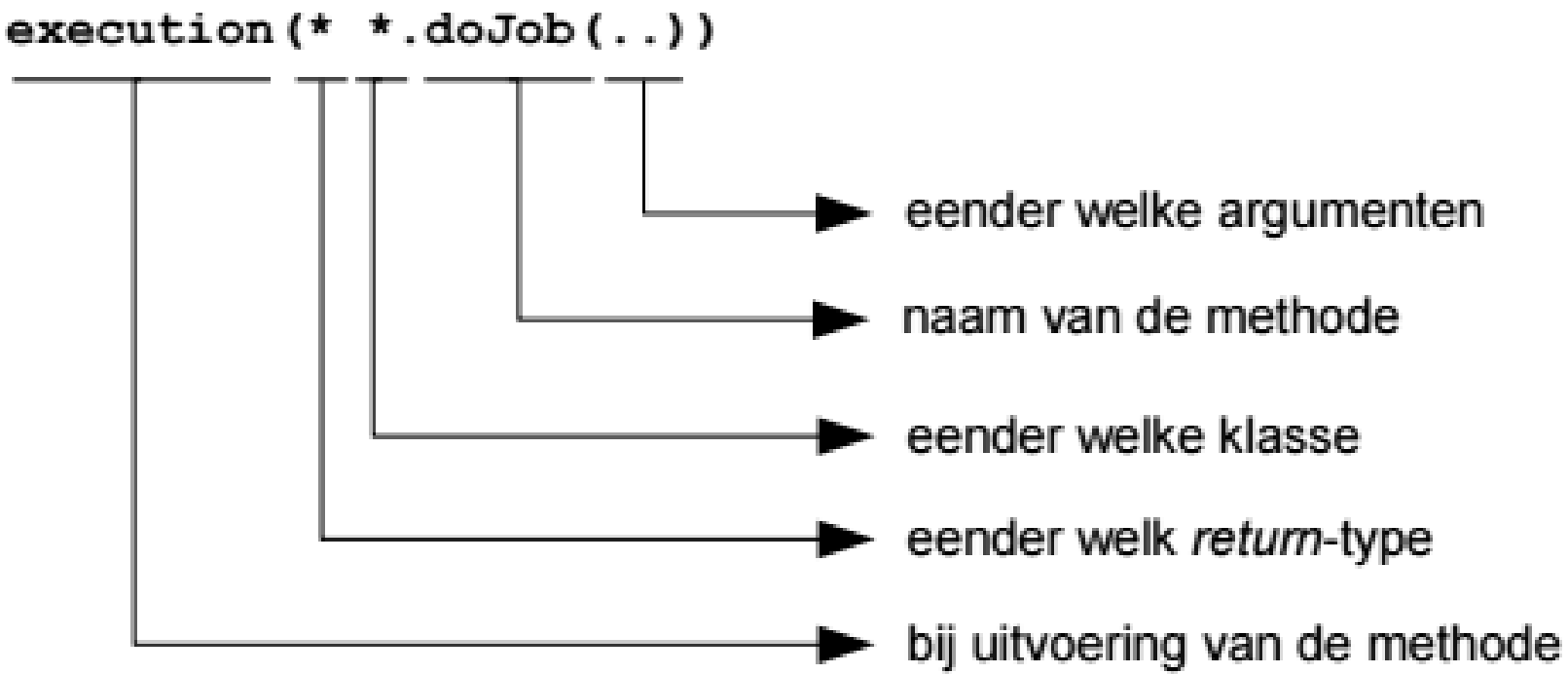
```
@Before("execution(* *.find*(..) )")
public void loggingAspect() {
    System.out.println("Retrieving data!");
}

@After("execution(* *.CustomerService.findAll(..))")
public void doSomethingAfter(JoinPoint joinPoint) {
    System.out.println("Do Something after findAll(): " +
    joinPoint.getSignature().getName());
}

@After("execution(void be.oak3.service.*.set*(..))")
public void logginAspectAfter() {
    System.out.println("Called setter!");
}
```

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Pointcut expressions



Afbeelding 22: Elementen van een pointcut expression

Testing in Spring

- **Unittesten**: jUnit → de kerntaak van elke klasse uittesten. Stubs/ Mocks gebruiken indien afhankelijkheden nodig zijn.

- **Integratietesten**: Samenwerking testen tussen meerdere componenten (klassen)

Testing: benodigdheden

Dependecies:

- jUnit
- Spring test: beans beschikbaar in test

Testing: annotaties

@Runwith(Springrunner.class)

→Eigen test runner van Spring

@SpringBootTest(classes = App.class)

→ Naam van de SpringBootApplicatie die uitgevoerd moet worden

@WebAppConfiguration

→ Specifieke voor testen van webapps.

Testing: annotaties

@TestPropertySource

→Specifiek propertybestand te gebruiken bij testen

@ActiveProfiles

→ Actieve profielen tijdens testen