PA165
Introduction to Spring framework

Petr Adámek

Content: Spring Framework

- Introduction into Spring
- Spring IoC
  - XML driven
  - Annotation driven
  - Code driven
- AOP
- Transaction Management

INTRODUCTION INTO SPRING

Spring framework

- http://projects.spring.io/spring-framework/
- Rod Johnson: *Expert One-on-One J2EE Design and Development* (2002)
- Non-invasive framework, very flexible, does not try to reinvent the wheel
- Current version 3.2.4
- Very good documentation (http://docs.spring.io/spring/docs/3.2.4.RELEASE/spring-framework-reference/html/)

SPRING IOC

XML Application Context

- Configured by XML
- Different modules (different namespaces)
- Benefits
  - Pure declarative approach
  - Allows to change system configuration without changing code (very useful for customizations)
- Weaknesses
  - Not transparent
  - Lots of configuration is needed
  - Hard to maintain
  - Problem with refactoring
- Example (property initialization, constructor initialization)

Annotation driven

- Configured by annotations
- Benefits
  - Less configuration needed
  - More clear and transparent
  - No problem with refaktoring
- Weaknesses
  - Less flexibility
- Can be combined with xml configuration (<context:annotation-config />, <context:component-scan />)

Annotation driven

- Proprietary
  - @Component, @Service, @Repository
  - @Autowired, @Required
- JSR-330:
  - @Named
  - @Inject, @Qualifier
- Other
  - @Resource, @PersistenceContext, @PersistenceUnit
- Example

Code driven

- Benefits
  - Almost any code could be evaluated during initialization
  - No problem with refactoring
- Weaknesses
  - Pure imperative approach
  - Configuration is hardcoded into the class
- @Configuration, @Bean
- @ComponentScan, @PropertySource, @Import
- Each @Bean method is evaluated just once!
- Example

SPRING AOP

AOP

- We often need to include some aspects that affects lots of our application parts
  - Transaction management
  - Authorization
  - Validation
  - Logging or auditing
  - Exception handling or translation

AOP

- Template method
  - OOP approach
  - You need new class for each operation
- AOP/Interceptors
  - Incompatible with pure OOP
  - More practical

AOP

- http://docs.spring.io/spring/docs/3.2.4.RELEASE/spring-framework-reference/html/aop.html

TRANSACTION MANAGEMENT

Transactions Management

- PlatformTransactionManager
  - DataSourceTransactionManager
  - JtaTransactionManager
  - JpaTransactionManager
  - HibernateTransactionManager
- Declarative approach (controlled with AOP, @Transactional)
- Imperative approach (API of PlatformTransactionManager)
- <tx:annotation-driven/>
- @EnableTransactionManagement
- Integration with JPA

Questions

?