

Program szkolenia:

Spring i Spring Boot - wprowadzenie i zagadnienia zaawansowane

Informacje:

Nazwa:	Spring i Spring Boot - wprowadzenie i zagadnienia zaawansowane
Kod:	Spring -core
Kategoria:	Spring Framework
Odbiorcy:	developerzy, architekci
Czas trwania:	3 dni
Forma:	50% wykłady / 50% warsztaty

Koncepcja szkolenia została oparta o zestaw praktycznych problemów jakie stają przed developerem – prezentujemy sprawdzone i najlepsze ich rozwiązania w Spring.

Szkolenie swoim zakresem wykracza daleko poza powszechnie dostępne materiały dydaktyczne. Szkolenie oprócz prezentacji technologii Spring zostało wzbogacone o aspekty: doboru architektury aplikacji, konfiguracji narzędzi developerskich, praktyk projektowych odpowiednich do produktywności pracy z Spring, istotnych zaawansowanych praktycznych aspektów JPA/Hibernate.

Wiedza zdobyta podczas szkolenia pozwoli na pełne wykorzystanie frameworka, zwiększenie rozszerzalności systemów oraz racjonalizację procesu testowania - zwiększając jakość z jednoczesną redukcją kosztów utrzymania testów.

Program szkolenia stanowi bazę do której możemy dodawać dowolnie wybrane zagadnienia ze szkolenia [Spring-moduły](#).

Sprawdź naszą implementację przykładowego projektu DDD+CqRS opartego o Spring Framework: [Sample Leaven](#).

Zalety szkolenia:

- Dobór architektury aplikacji
- Wzorce i pułapki
- Nacisk na testability - architektura wspierająca testy
- Elementy Domain Driven Design

Szczegółowy program:

1. Warstwowa architektura systemów opartych o Spring

1.1. Wzorce

1.1.1. Wzorce architektoniczne

1.1.2. Wzorce projektowe stosowane w aplikacjach webowych

1.1.3. Wzorce porządkowania logiki biznesowej

1.2. Architektura dla podejścia Domain Driven Design i CqRS(na życzenie)

1.3. Architektura zorientowana na testy

1.3.1. Testy jednostkowe w warstwie domenowej

1.3.2. testy end2end w warstwie serwisów (API)

1.4. Spring boot - architektura rozwiązania

2. Kontener

2.1. Konfiguracja

2.2. Techniki uruchamiania

2.3. Zasada działania – Inversion of Control (w szczególności Dependency Injection)

2.4. Dodatkowe moduły

3. Komponenty

3.1. Deklaracja - Java config, adnotacje i XML (dobór właściwego podejścia w zależności od problemu)

3.2. Cykl życia

3.3. Zależności

3.4. Zasięg komponentów

3.4.1. Pułapki Scoped Proxy

3.4.2. Wstrzykiwanie krócej żyjących komponentów do dłużej żyjących

3.4.3. Praktyczne zastosowanie: model zalogowanych użytkowników

3.5. Trzy sposoby definiowania - dobór do problemu, łączenie wszystkich technik

3.5.1. Adnotacje - dla rdzennych obiektów

3.5.2. XML - dla logiki, która jest specyficzna dla wdrożenia

3.5.3. Java - gdy potrzeba dynamiki w runtime

3.6. Pre/post – procesory

4. Paradygmat Inversion of Control - wsparcie frameworka dla 3 podejść

4.1. Dependency Injection

4.1.1. Wykorzystanie do zmniejszenia poziomu zależności

4.2. Praktyczne przykłady modularyzacji aplikacji biznesowych - wstrzykiwanie strategii (policy) biznesowych

4.3. Zdarzenia

4.3.1. Zdarzeniowe architektury otwarte na rozbudowę

4.3.2. Asynchroniczne przetwarzanie zdarzeń jako technika zwiększająca skalowalność

4.3.3. Praktyczne przykłady generowania zdarzeń biznesowych z warstwy logiki

4.4. Techniki Aspect Oriented Programming

4.4.1. Podstawy teoretyczne

4.4.2. Praktyczne przykłady wykorzystania w aplikacjach biznesowych

4.4.3. Zastosowanie w Spring

4.4.4. Zastosowanie do budowy własnych frameletów

5. Funkcjonalności kontenera

5.1. Zarządzanie zasobami

5.2. Język wyrażeń

5.3. Async

5.4. Quartz

6. Warstw dostępu do danych

6.1. Transakcje (konfiguracja, poziomy izolacji, warstwa abstrakcji Spring)

6.2. Integracja z JPA/Hibernate (szczegółowy program w module JPA/Hibernate)

6.3. Integracja z JDBC

6.4. Spring Data

6.4.1. Automatyzacja powtarzalnego kodu

6.4.2. Standaryzacja dostępu do danych (paginacja, transakcje)

6.4.3. Najlepsze praktyki

7. MVC

7.1. Dispatcher Servlet

7.2. Implementowanie kontrolerów

7.3. Widoki

7.4. Konfiguracja

8. Testowanie

8.1. Spring - wsparcie dla testów

8.2. Testowanie jednostkowe - techniki mockowania

8.3. Testowanie integracyjne - wsparcie kontenera

8.4. Profile testowe

8.5. Wydzielanie deskryptorów na potrzeby testów

8.6. Behavior Driven Development

8.6.1. Idea BDD

8.6.1.1. User Story - dobre praktyki

8.6.1.2. Wykonywalne scenariusze akceptacyjne

8.7. Specification by Example

8.8. Spring i Cucumber

9. Spring boot

9.1. budowa

9.2. konfiguracja

9.3. najlepsze praktyki