

Program szkolenia:

Java Persistence API - zagadnienia zaawansowane

Informacje ogólne

Nazwa:	Java Persistence API - zagadnienia zaawansowane
Kod:	JPA-pro
Kategoria:	Java Enterprise Edition
Grupa docelowa:	Architekci, Projektanci, programiści
Czas trwania:	2 dni
Forma:	50% wykłady / 50% warsztaty

Wydajność jest częstym problemem systemów opartych o ORM dlatego program szkolenia dobrano pod kątem optymalizacji wykorzystania JPA.

Szkolenie przedstawia typowe błędy programistyczne wpływające na drastyczny spadek wydajności, sposoby ich wykrywania oraz zapobiegania.

Materiał uzupełniono o specyficzne zagadnienia Hibernate, takie jak cache oraz strategie mapowania encji.

Zalety szkolenia:

- » Zwracamy szczególną uwagę na wydajność
- » Prezentujemy typowe jak i mniej typowe pułapki
- » Wskazujemy rozwiązania każdego z omawianych problemów

Program szkolenia:

1. Modelowanie encji z wykorzystaniem technik Obiektowych

1.1. Elementy Domain Driven Design

1.2. Mapowanie powiązań, sterowania kierunkiem właściciela

1.3. Mapowania dziedziczenia - klasy bazowe dla Agregatów

1.4. Wprowadzania do modelu ValueObjects zapewniających immutability i zwiększających siłę wyrazu modelu (adnotacja Embeded)

1.5. Enkapsulacja modelu

1.6. Określanie wyraźnej granicy grafu obiektu - określanie jednostki zmiany

2. Architektura dostępu do danych w aplikacjach warstwowych

2.1. Abstrakcja źródeł danych

2.2. Wzorce DAO i Repository

2.3. Wpływ na testability

2.4. Wpływ na przenośność i skalowalność systemu

2.5. Base DAO/Repository

2.6. Umieszczenie dostępu do danych w architekturze aplikacji

2.7. Projekt architektury pod kątem testowalności

3. Wydajność dostępu do danych

3.1. Pułapki wydajności JPA

3.1.1. „n+1 Select problem” - wykrywanie i zapobieganie

3.2. Pułapki Lazy Loadingu oraz zbyt chciwego pobierania danych

3.2.1. Nadmierne pobieranie danych

3.2.2. Racjonalne wykorzystanie Lazy Loadingu

3.3. Optymalne mapowanie encji

3.3.1. Nadmiarowość pobierania danych

3.3.2. Problemy z leniwym ładowaniem pól

3.4. Pobieranie konkretnych atrybutów

4. JPA czy native SQL – dobór odpowiedniego narzędzia do konkretnego problemu

4.1. Podejście pragmatyczne: refaktoryzacja z JPA na SQL przy pomocy DAO i wstrzykiwania zależności

4.2. Dostęp do danych w kontekście architektury Command-query Responsibility Segregation

4.2.1. Dedykowany model do odczytu - pobieranie danych odpowiednich do prezentacji

4.2.2. Uaktualnianie modelu do odczytu

5. Hibernate cache – niezastąpione rozwiązanie.

5.1. Idea działania i konfiguracja

5.2. Cache pierwszego poziomu

5.3. Cache drugiego poziomu

5.4. Cache zapytań - najlepsze praktyki

5.5. Mapowanie encji zorientowanie na cacheowanie

6. Entity Manager - tryb rozszerzony

6.1. Naturalna granica jednostki pracy

6.2. Transakcje aplikacyjne

6.3. Model konwersacji

6.4. Manualne opróżnianie kontekstu

7. Testowanie

7.1. Racjonalizacja architektury w kontekście piramidy testów

7.2. Testowanie operacji na JPA

7.3. Unikanie mockowania źródeł danych w testach jednostkowych

7.4. Preferowanie testów end-to-end gdy pobieramy dane

7.5. Zwiększanie pokrycia poprzez testy jednostkowe

8. Transakcje w Java EE

8.1. Deklaratywne (zarządzane przez kontener)

8.2. Koncepcja transakcji sterowanych wyjątkami

8.3. Zarządzane przez Bean

8.4. Zarządzane przez klienta – praktyczne wykorzystanie w przypadku łączenia z operacjami nie tylko bazodanowymi

8.5. Zagadnienie propagacji transakcji - Praktyczne przykłady na każdy z typów

8.6. Zagadnienie izolacji transakcji – wpływ na wydajność

8.7. Transakcje rozproszone w środowisku JEE

8.7.1. Konfiguracja

8.7.2. Zasada działania „double commit”