Laboratorium 8 - SOA.
Tematyka: JPA, Hibernate

Praca z bazami danych

Celem laboratorium jest zaznajomienie z technologią pracy z relacyjnymi bazami danych w oparciu o standard JPA.

Zadanie1. Proszę zaznajomić się z dokumentem źródłowym specyfikacji JPA https://docs.oracle.com/javaee/7/tutorial/index.html - rozdział 8 – Persistance

Wprowadzenie

Enterprise JavaBeans zawiera specyfikację dotyczącą trwałości, nazwaną **Java Persistence API (JPA)**. To interfejs przeznaczony do tworzenia, usuwania i odpytywania obiektów Javy nazywanych encjami, z których można korzystać zarówno w kontenerze EJB, jak i w standardowym środowisku Javy SE.
Pojawienie się standardu **Enterprise Java Persistence** bazującego na modelu programistycznym **POJO (Plain Old Java Object)** wypełnia istotną lukę w platformie Javy EE. Poprzednia próba specyfikacja EJB 2.x (tzw. ziarena encyjne) nie powiodła się i nie zyskała uznania w środowsiku deweloperów1) JAVA EE. .
W odróżnieniu od ziaren encyjnych EJB 2.x, interfejs EJB 3.0 Java Persistence API (JPA) to technologia bazująca na POJO i metadanych. Umożliwia zapis danych znajdujących się w obiektach Javy w dowolnej bazie danych. Używane obiekty nie muszą implementować interfejsu, rozszerzać klasy lub dopasowywać się do żadnego konkretnego frameworka.
Inną kluczową cechą JPA jest język zapytań nazywany Java Persistence Query Language (JPQL), który stosuje składnię zapytań zapewniającą działanie systemu niezależnie od stosowanego rodzaju bazy danych. Zapytania JPA składniowo przypominają zapytania SQL, ale działają na poziomie obiektów encji, a nie na poziomie tabel bazy danych.

Korzystanie z JPA

JPA zostało zainspirowane frameworkami ORM więc używa się w nim adnotacji do odwzorowania obiektów na elementy relacyjnych baz danych. Encje JPA są zwykłymi obiektami POJO, więc nie dziedziczą po żadnej innej klasie i nie implementują żadnych dodatkowych interfejsów. Odwzorowania nie wymagają nawet deskryptorów XML. W zasadzie API JPA składa się jedynie z adnotacji i kilku klas oraz interfejsów.

Oznaczenie klasy Company jako encji wyglądałoby następująco.

```
@Entity
public class Company {
        ...
public Company () { }
@Id
String companyName;
...
}
```

Ten krótki fragment kodu przedstawia minimalne wymagania dla klasy, która chce obsługiwać trwałość. Innymi słowy musi:

- być wskazana jako encja przy użyciu adnotacji @javax.persistence.Entity,
- posiadać atrybut identyfikatora wskazany adnotacją @javax.persistence.Id,
- posiadać konstruktor bezargumentowy.

Zadanie2. Zapoznaj się z prezentacjami na temat JPA znajdującymi się z katalogu JPA.

Konfiguracja trwałości

Instalacja sterownika JDBC w JBoss AS 7

W Javie za połączenie z bazami danych odpowiadają sterowniki JDBC, które są stosowane bezpośrednio przez aplikację lub też korzystają z nich biblioteki JPA czy Hibernate. Również sterownik JDBC dla MySQL można pobrać bezpłatnie pod adresem http://dev.mysql.com/downloads/connector/j/.

API dotyczące encji wygląda na proste i intuicyjne, ale jak poinformować serwer, która baza danych odpowiada za przechowanie obiektów encji. Plik *persistence.xml*, który zostanie umieszczony w folderze *src/main/resources/META-INF* projektu, stanowi standardową konfigurację JPA. Dzięki niemu możliwe jest łatwe przełączenie się z jednego dostawcy trwałości do następnego, a tym samym z jednego serwera aplikacji do innego (to naprawdę duży krok w przód w kwestii zgodności różnych serwerów aplikacji).

W pliku persistence.xml musimy określić dostawcę trwałości i używane źródło danych.

```
<persistence version="2.0"
xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
<persistence-unit name="myDatabase">
<jta-data-source>java:jboss/datasources/testdevelopment</jtadata-source>
<class>com.test.model.Seat</class>
<class>com.test.model.SeatType</class>

<properties>
<!-- Właściwości dla Hibernate. -->
 <property name="hibernate.hbm2ddl.auto" value="create-drop" />
 <property name="hibernate.show_sql" value="false" />
</properties>
</persistence-unit>
</persistence>
```

Atrybut name jest elementem niezbędnym, ponieważ posłuży do odnoszenia się do konkretnej jednostki trwałości z poziomu komponentów EJB.

Druga część poprawek dotyczy właściwości specyficznych dla dostawcy, na przykład hibernate. hbm2ddl.auto, co oznacza automatyczne usuwanie i tworzenie tabel bazy danych przy każdym wdrożeniu aplikacji. To duża wygoda, jeśli chce się rozpoczynać pracę z czystą bazą danych po każdym wdrożeniu. Dodatkowo dołączyliśmy atrybut hibernate.show_sql, który, jeśli zostanie włączony, będzie wyświetlał zapytania SQL wysyłane przez Hibernate do serwera bazy danych.

Prace rozpoczynamy od przerobienia następujących tutoriali i przykładów znajdujących się pod poniższymi adresami.

Tutorial prezentujący w jaki sposób tworzyć za pomocą kreatora projekt z JPA w Eclipsie

https://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.jpt.doc.user%2Ftask_create_new_project.htm

Tutoriale cząstkowe dotyczące wybranych elementów implementacji JPA.

http://iprogramujesz.pl/hibernate-dla-poczatkujacych

http://www.objectdb.com/tutorial/jpa/start -> tutaj proszę zwrócić uwagę na porównanie różnych technologii zgodnych ze specyfikacją JPA

http://www.mkyong.com/tutorials/hibernate-tutorials/

Zadanie 2. W sekcji Hibernate znajduje się tutorial opisujący korzystanie z Hibernate. Proszę samodzielnie zrealizować opisane tam przykłady.

Zadanie 3. Oparciu o dowolne rozwiązanie samodzielnie zbudować aplikacje typu katalog książek. Aplikacja umożliwia podgląd, dodawanie, usuwanie i modyfikacje pozycji katalogu. Katalog zawiera następujące pozycje: nazwisko autora, imię, tytuł, numer ISBN, rok wydania, cena.

1. Stwórz bazę danych z tabelą *Books* zawierającą wyżej wymienione pola. Pamiętaj o kluczu głównym i automatycznej jego inkrementacji.
2. Stwórz formularz wyświetlający listę dotychczas wprowadzonych książek umieszczając go w pliku *index.jsp*. Książki powinny być wypisane w tabeli. Ostatnie dwie kolumny tabeli (dla każdego wiersza) powinny być przeznaczone na opcje *Edytuj* oraz *Usuń*. Na dole strony powinien być umieszczony przycisk *Dodaj*.
3. Stwórz formularz *bookEdit.jsp* pozwalający na dodanie nowej oraz edycję wybranej książki. Formularz powinien zawierać takie pola, jakie zdefiniowane są w tabeli w bazie danych.