JPA - podsumowanie

➔ **bibliotek Hibernate**

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.3.1.Final</version>
</dependency>
```

➔ **logger** - przyda się do logowania komunikatów:

```
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.2.3</version>
</dependency>
```

➔ **META-INF/persistence.xml** - konfiguracja Persistence Unit

➔ **encje** - model danych, czyli klasy + mapowanie (XML lub adnotacje)
http://www.techferry.com/articles/hibernate-jpa-annotations.html

➔ ***EntityManagerFactory*** - stworzony na podstawie konfiguracji z **persistence.xml**

➔ ***EntityManager* + klasy DAO -** kod JPA do obsługi komunikacji z bazą danych

Zadanie nr 10:

1. Stwórz nowy moduł Maven o nazwie: **zoo-keeper-jpa**

2. Na podstawie bazy danych **zoo-keeper** przygotowanej w **zadaniu nr 4** przygotuj kod w module **zoo-keeper-jpa** do pracy z bazą danych zgodnie z rozpiską podaną na początku (konfiguracja, encje, DAO itp)

3. Wykonaj ćwiczenia opisane w **zadaniu nr 4** tym razem używając kodu JPA

4. **(dla chętnych)** Stwórz nową tabelkę **animal_keeper**, z polami: *id, name* , w której będą znajdować się dane opiekunów zwierząt w zoo. Dodaj relację opiekun - zwierzę - przyjmujemy że jedno zwierzę może mieć wielu opiekunów i jeden opiekun może zajmować się wieloma zwierzętami. Do klasy ***AnimalsDao*** dodaj metodę która wyciągnie opiekunów dla podanego zwierzęcia oraz zestawienie wszystkich zwierząt i ilość ich opiekunów (jako lista osobnych obiektów z polami: *animalName, animalKeepersNumber*)

5. **(dla chętnych)** Przejdź do modułu **library-manager** i spróbuj napisać klasy: ***JpaBooksDao, JpaOrdersDao, JpaUsersDao*** implementujące odpowiednie interfejsy. Użyj nowych klas w ***DaoFactory*** i sprawdź czy aplikacja działa.