ZAAWANSOWANE TECHNIKI WWW (WFAIS.IF-C112)
(zajęcia 28.01.2016 r.)

Wprowadzenie do ORM (Mapowanie obiektowo-relacyjne)

Mapowanie obiektowo-relacyjne (ORM) ma na celu usystematyzowanie i ułatwienie komunikacji pomiędzy aplikacją, a bazą danych, w zasadzie elimunując konieczność stosowania języka zapytań np. SQL.

W środowisku uruchomieniowym Node.js, pakietem służącym jako interfejs ORM jest Sequelize (http://docs.sequelizejs.com/). Pakiet tej jest bogatym zbiorem metod umożliwiających:

- stworzenie modelu danych,
- nawiązanie połączenia z bazą danych,
- wykonywanie zapytań i agregację danych.

a.) Instalcja w wybranym projekcie Express.js oraz nawiązanie połączenia:

```
npm install sequelize --save
```

Dostęp i nawiązanie połączenia z bazą danych:

```
var Sequelize = require('sequelize');
var sequelize = new
Sequelize('nazwa_bazy_danych','nazwa_uzytkownika','haslo', {
                              host: 'localhost', // nazwa hosta
                              port: 3306 // numer portu
                              } );
```

b.) Stworzenie modelu danych:

Model danyc ma odwozorowywać fizyczną strukturę bazy danych czyli relacje połączone związakim. Pierwszym krokiem w tworzeniu modelu jest zadeklarowanie wszystkich relacji używając metody .define():

```
Projekt = sequelize.define('Projekt',{
          tytul: Sequelize.STRING,
          opis:  Sequelize.TEXT,
          data:  Sequelize.DATE
    });
```

gdzie pierwszy parametr metody oznacza nazwę relacji (tabeli), a drugi stanowi obiekt własności relacji z zadeklarowanym typem (patrz strona: http://docs.sequelizejs.com/en/latest/docs/models-definition/)

Zadeklarujmy drugi obiekt:

```
Zadanie = sequelize.define('Zadanie',{
```

```
        tytul: Sequelize.STRING,
        stat:  Sequelize.STRING
        });
```

Następnie musimy zadeklarować związki pomiędzy relacjami uzywając: metod: .belongsTo(), hasMany():

```
    this.Zadanie.belongsTo(Projekt);
    this.Projekt.hasMany(Zadanie);
```

c.) Nawiązanie połączenia i z bazą i jej synchronizacja:

Dzięki ORM nie musimy “ręcznie” implementować struktury bazy danych, ponieważ Sequelize przy pierwszym połączeniu sam zrobi to za nas używając informacji z modelu:

```
     sequelize.sync();
```

d.) Wykonywanie zapytań i ich wyświeltanie.

Do odpytywania bazy danych o dane ich dodawanie i usuwanie jest wyknywane za pomocą zestawu funkcji. Przykładowo gdy chcemy odpytać bazę danych o wszystkie dosßepne projekty przechowywane w bazie (zgodnie z stworoznym modelem) musimy wywołać metodę: .findAll():

```
Projekt.findAll()
   .then(function(projekty){
      res.render('index', { title: 'Przykład ORM', data: projekty });
   })
   .error(0);
```

W wyniku działania tej metody zostanie zwrócony obiekt projekty, który jest tablicą [0...n] (w zależności od liczby rezultatów), który zawiera dane w obiekcie “dataValues” o polach zgodnych ze stworonym modelem. Zatem obsługa zwróconego obiektu z bazy danych za powyższego zapytania i przekazanie go do widoku musi zostać w widoku obsłużone w następujący sposób:

```
div
   each item in data
    span #{item.dataValues.tytul} |  #{item.dataValues.opis} |
#{item.dataValues.data}
    br
```

ZADANIE:

Proszę przygotować aplikacje w środowisku Express, w której będzie model danych złożony z dwóch relacji (tak jak w przykładzie powyżej) i będzie obsługiwany za pomocą mapowania obiektowo-relacyjnego ORM za pomocą pakietu Sequelize. Następnie proszę przygotować funkcje kontrolera wyświetlające dane z tak obsługiwanej bazy danych