Warsztaty III
szkoła programowania

v.1.0.0

Coders Lab
SZKOŁA PROGRAMOWANIA

Cel warsztatów

- Celem warsztatów jest wykonanie projektu **szkoły programowania** z wykorzystaniem nowo poznanych technologii (servlety oraz JSP)**.**
- Projekt ma wykorzystywać warstwę dostępu do danych wykonaną podczas poprzedniego warsztatu.

- Projekt powinien oprogramowywać opisane funkcjonalności.
- Projekt może zostać rozwinięty o dowolny zestaw własnych funkcjonalności.

Np.:
- Wiadomości wewnętrzne
- Oceny i opinie
- Umiejętności użytkowników
- Działy zadań

Funkcjonalności aplikacji

Aplikacja ma implementować następujące funkcjonalności.

Zarządzanie tzw. danymi słownikowymi:

- Wyświetlanie listy danych
- Dodawanie danych
- Usuwanie danych
- Edycja danych

Za słownik w naszej aplikacji uważamy:

Grupy użytkowników
Możliwość zarządzania grupami użytkowników.

Funkcjonalności aplikacji

Zarządzanie zadaniami
Aplikacja ma umożliwiać:

- Możliwość przeglądania, dodawania, usuwania, edycji zadań.

Zarządzanie rozwiązaniami zadań
Aplikacja ma umożliwiać:

- Możliwość przeglądania, dodawania, usuwania, edycji rozwiązania do zadania.

Funkcjonalności aplikacji

Zarządzanie użytkownikami
Aplikacja ma umożliwiać zarządzanie użytkownikami

- Możliwość przeglądania, dodawania, usuwania, edycji użytkownika.

Strony, które musi mieć aplikacja

Strona główna aplikacji
Ma mieć możliwość przejścia do poszczególnych elementów aplikacji (nawigację w postaci linków).

Wyświetlać 5 ostatnio dodanych rozwiązań zadań z możliwością przejścia do konkretnego rozwiązania.

Strona grup
Strona ma wyświetlać listę wszystkich grup z możliwością przejścia do użytkowników danej grupy.

Strona użytkowników danej grupy
Ta strona ma wyświetlać listę wszystkich użytkowników należących do danej grupy z możliwością przejścia do strony danego użytkownika.

Strona użytkownika
Strona ma wyświetlać wszystkie dane jakie posiadamy na temat użytkownika.

Wszystkie dodane przez niego rozwiązania zadań w kolejności od najnowszych.

Strony, które musi mieć aplikacja

Strona rozwiązań zadania

Strona musi wyświetlać treść zadania oraz wszystkie jego rozwiązania wraz danymi użytkownika, który je dodał.

Panel administracyjny

Ma on być dostępny po wejściu na podstronę /panel.

Strona główna panelu ma mieć linki prowadzące do stron, na których zostaną wyświetlone listy elementów:

- Link do listy zadań,
- Link do listy grup użytkowników,
- Link do listy użytkowników.

Prototyp

Jeżeli nie wiesz od czego zacząć, zapoznaj się z prototypem znajdującym się pod adresem.

https://app.moqups.com/arek-cl/CLpMMElO6t/view/page/ad64222d5

Prototyp nie zawiera kompletnego zestawu funkcjonalności – jest tylko wzorem, który ma Cię wspomóc w rozpoczęciu prac.

Nie musisz zwracać zbytniej uwagi na kwestie wizualnego interfejsu aplikacji.

Przygotowanie

Przygotowanie

Zadanie: Przygotowanie

- Załóż repozytorium na GitHubie
- Podepnij swoje nowe projekty do repozytorium i zobacz, czy działa
  (np. przez dodanie pliku readme na GitHubie i ściągnięciu go na oba komputery).

Zadanie: Połączenie do bazy danych

- Do stworzenia połączenia wykorzystamy kontener servletów Tomcat, oraz odpowiednio skonfigurowany zasób DataSource.

W tym celu należy stworzyć plik context.xml w lokalizacji META-INF projektu.

Szczegółowe informacje o możliwościach konfiguracyjnych możemy znaleźć tutaj: http://tomcat.apache.org/tomcat-8.0-doc/jdbc-pool.html

Przygotowanie

Zadanie: Połączenie do bazy danych

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
    <Resource name="jdbc/school"
    auth="Container"
    type="javax.sql.DataSource"
    username="root"
    password="root"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/school"
    maxActive="100"
    maxIdle="30"
    maxWait="10000" />
</Context>
```

Nazwa zasobu – będziemy z niej korzystali.

Dane autoryzacyjne do bazy danych.

Adres url bazy danych.

Przygotowanie

Zadanie: Połączenie do bazy danych

Dobrą praktyką jest wydzielenia kodu odpowiedzialnego za tworzenie połączenia.

Przykład takiej klasy korzystającej z zasobu który wcześniej zdefiniowaliśmy mamy poniżej.

Pozwoli nam to skupić się bezpośrednio na funkcjonalności.

Przygotowanie

Zadanie: Połączenie do bazy danych

```
public class DbUtil {
    private static DataSource ds;
    public static Connection getConn() throws SQLException {
        return getInstance().getConnection();
    }
    private static DataSource getInstance() {
        if(ds == null) {
            try {
              Context ctx = new InitialContext();
              ds = (DataSource)ctx.lookup("java:comp/env/jdbc/school");
            } catch (NamingException e) {
                e.printStackTrace();
            }
        }
        return ds;
    }
}
```

Zasób określony wcześniej.

Przygotowanie

Zadanie: Połączenie do bazy danych

Aby utworzyć połączenie skorzystamy ze statycznej metody **getConn()** w następujący sposób:

```
Connection c = DbUtil.getConn();
```

Co zastępuje znane nam z zajęć o MySQL:

```
Connection conn =
DriverManager.getConnection("jdbc:mysql:
//localhost:3306/school?useSSL=false",
"root","coderslab");
```

Pozostały kod związany z połączeniem pozostanie bez zmian.

Zadania

Zadanie 1

Zadanie: Warstwa bazy dancych

Stwórz pakiet **pl.coderslab.model**

Skopuj do niego klasy dostępu do danych z poprzedniego warsztatu. Jeżeli to konieczne popraw **package.**

Zmodyfikuj metody tak, aby korzystały z klasy **DbUtil** – czyli nowego sposobu tworzenia połącznia do bazy danych.

Zadanie 2

Ćwiczenia

- Podczas ćwiczeń stworzysz stronę główną aplikacji wyświetlającą 5 ostatnio dodanych rozwiązań.
- Utwórz pakiet **pl.coderslab.controller** – w którym będziemy umieszczać klasy naszych kontrolerów - servletów.
- W klasie modelu **Solution** dopisz przeciążoną metodę **loadAll(int)** - która przyjmie dodatkowy parametr typu **int** – parametr ten posłuży nam do ograniczenia ilości zwracanych wierszy.
- W tym celu posłużymy się w naszym zapytaniu SQL słowem kluczowym **LIMIT.**
- Dodaj klauzulę SQL **ORDER BY –** tak by pobierać najnowsze rozwiązania.

Zadanie 2

Ćwiczenia

- Utwórz servlet z mapowaniem dla strony startowej @WebServlet("/").
- W metodzie **doGet** utwórz obiekt klasy **Solution.**
- Utwórz w deskryptorze wdrożenia parametr dla całej aplikacji o nazwie **number-solutions** oraz wartości 5.

Zadanie 2

Ćwiczenia

Utworzony parametr wykorzystamy do określenia ilości wyświetlanych rozwiązań na stronie głównej.

Dzięki temu będziemy mogli w łatwy sposób modyfikować ich ilość w zależności od potrzeb.

Określanie parametrów w kodzie.

Wszelkiego rodzaju elementy które mogą ulec zmianie ze względu na wymogi biznesowe np. ilość wpisów na stronie głównej, o ile jest to możliwe zawsze lepiej umieszczać w konfiguracji.

Tak by ich zmiana nie wymagała modyfikacji naszych plików z kodem.

Zadanie 2

Ćwiczenia

- Utwórz pliki **header.jsp**, **footer.jsp** oraz plik **index.jsp**, który będzie je załączał – w ten sposób stworzymy szablon naszej aplikacji.
- W pliku **header.jsp** umieścimy linki nawigacyjne naszej aplikacji.
- W pliku **footer.jsp** umieścimy stopkę informacyjną.
- W metodzie **doGet** pierwszego servletu pobierz utworzony wcześniej parametr określający ilość wyświetlanych rozwiązań na stronie startowej.
- Następnie wywołaj metodę **loadAll(int)** na obiekcie klasy **Solution** wykorzystując uprzednio pobraną wartość parametru.
- Przekaż pobraną listę do widoku **index.jsp** a następnie wyświetl szczegóły rozwiązań w wierszach tabeli html.

Zadanie 3

Szczegóły rozwiązania

- Dodaj na stronie głównej w tabeli html z rozwiązaniami zadań odnośnik url do servletu wyświetlającego szczegóły rozwiązania.
- Pamiętaj o przekazaniu identyfikatora.
- Utwórz servlet który pobierze z **GET** parametr – na jego podstawie przy pomocy metody **loadById** pobierz obiekt rozwiązania zadania i przekaż go do widoku.
- Utwórz widok i wyświetl w nim szczegóły rozwiązania zadania.

Zadanie 4

Grupy

- Dodaj w pliku **header.jsp** link nawigacyjny do servletu wyświetlającego listę wszystkich grup.
- Utwórz servlet który przy pomocy metody **loadAll** pobierze wszystkie grupy i przekaż go do widoku.
- Utwórz widok a następnie w tabeli html umieść listę grup oraz linki zawierające **id** grupy.
- Linki mają prowadzić do listy użytkowników danej grupy.

Zadanie 5

Użytkownicy danej grupy

- Utwórz servlet który pobierze z **GET** parametr – na jego podstawie przy pomocy metody **loadAllByGrupId** pobierz listę wszystkich użytkowników danej grupy i przekaż ją do widoku.

- Utwórz widok a następnie w tabeli html umieść listę wszystkich użytkowników grupy oraz linki zawierające **id** użytkownika.

- Linki mają prowadzić do szczegółów danego użytkownika.

Zadanie 6

Szczegóły użytkownika

- Utwórz servlet który pobierze z GET parametr – na jego podstawie przy pomocy metody **loadById** klasy **User** pobierz obiekt i przekaż go do widoku.
- Pobierz listę wszystkich rozwiązań dodanych przez użytkownika i przekaż ją do widoku. Jeżeli nie posiadasz odpowiedniej metody – dopisz ją.
- Utwórz widok i wyświetl na nim wszystkie szczegóły użytkownika.
- Wyświetl listę wszystkich rozwiązań dodanych przez użytkownika.

Zadanie 7

Panel administratora

- Utwórz servlet dostępny pod adresem **/panel.**

- Utwórz widok i wyświetl linki prowadzące do list poszczególnych elementów:

  - Listy zadań,
  - Listy grup użytkowników,
  - Listy użytkowników.

Zadanie 7

Panel administratora - grupy

- Utwórz servlet służący do zarządzania grupami użytkowników (wyświetlanie listy, dodanie/edycja).
- W metodzie **doGet** servletu pobierz listę grup za pomocą metody **loadAll** klasy **Group** a następnie przekaż ją do widoku.
- W metodzie **doPost** dodaj obsługę parametrów pobranych metodą POST z formularza.
- Utwórz widok a następnie w tabeli html umieść listę wszystkich grupy oraz link prowadzący do edycji grupy.
- Utwórz widok z formularzem umożliwiającym dodawanie/edycję grupy.

Zadanie 7

Panel administratora - użytkownicy

- Utwórz servlet służący do zarządzania użytkownikami (wyświetlanie listy, dodanie/edycja).
- W metodzie **doGet** servletu pobierz listę grup za pomocą metody **loadAll** klasy **Users** a następnie przekaż ją do widoku.
- W metodzie **doPost** dodaj obsługę parametrów pobranych metodą POST z formularza.
- Utwórz widok a następnie w tabeli html umieść listę wszystkich użytkowników oraz link prowadzący do edycji użytkownika.
- Utwórz widok z formularzem umożliwiającym dodawanie/edycję użytkownika.

Zadanie 8

Panel administratora - zadania

- Utwórz servlet służący do zarządzania zadaniami (wyświetlanie listy, dodanie/edycja).
- W metodzie **doGet** servletu pobierz listę grup za pomocą metody **loadAll** klasy **Exercise** a następnie przekaż ją do widoku.
- W metodzie **doPost** dodaj obsługę parametrów pobranych metodą POST z formularza.
- Utwórz widok a następnie w tabeli html umieść listę wszystkich użytkowników oraz link prowadzący do edycji zadania.
- Utwórz widok z formularzem umożliwiającym dodawanie/edycję zadania.

Zadanie dodatkowe

Zadanie dodatkowe

- Zmodyfikuj wykorzystywaną warstwę dostępu dodanych opartą o wzorzec **Active Record** tak by implementowały **DAO**.

- Zadanie polega na rozdzieleniu klas dostępu do danych na dwie klasy pierwsza zawierać ma jedynie atrybuty klasy oraz getery i setery do atrybutów. Druga pozostałe metody.

Przykładowo klasę User rozdzielimy na dwie klasy User oraz UserDao.

Klasy Dao powinny implementować przynajmniej zestaw operacji CRUD.

Skrót ten oznacza:
Create
Read
Update
Delete

https://en.wikipedia.org/wiki/Data_access_object

Zadanie dodatkowe

Zadanie dodatkowe

Stwórz pakiet pl.coderslab.dao a w nim klasy dostępu do danych (Data Access Object).

- Jest to wzorzec podobny do znanego już nam ActiveRecords z tą różnicą że wszystkie metody związane z operacjami na obiekcie są wydzielone do oddzielnej klasy np. UserDao.
- Zmodyfikuj aplikację tak by korzystała z nowych klas.

Wszystkie nasze metody dostępu do danych powinny się znajdować w nowo utworzonych klasach.

Są to np. **loadAll, loadUserById, loadAllByUserId, loadAllByExerciseId.**

W zależności od przeznaczenia metody powinny się znajdować w odpowiednio nazwanych klasach.

Np. UserDao – metody korzystające z tabeli users.