НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

Обчислювальної техніки

КУРСОВА РОБОТА
з «ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

на тему: «Система організації вуличних змагань «WorkOUT». Робоче місце гостя »

Студента 2 курсу групи ІО-32
напряму підготовки
6.050102 «Комп'ютерна інженерія»

Попенко Руслан Леонідович

Керівник
Болдак Андрій Олександрович
(прізвище та ініціали)

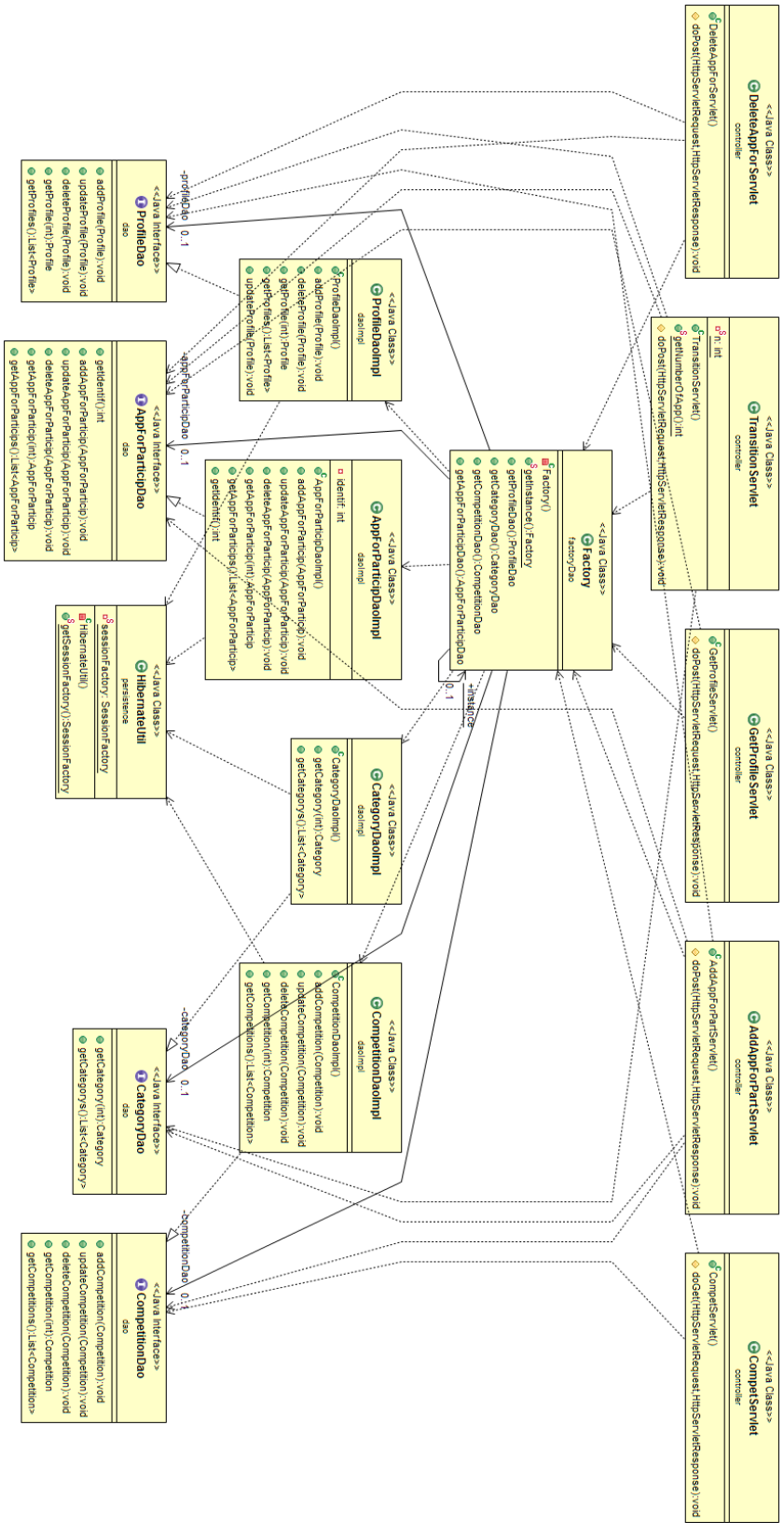
Доцент кафедри ОТ
(посада, вчене звання, науковий ступінь)

Національна шкала_____
Кількість балів: _____
Оцінка: ECTS _____

м. Київ - 2015 рік

ДОДАТОК А

Діаграма класів



ДОДАТОК Б

SQL код для створення таблиць бази даних:

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
-- -----------------------------------------------------
-- Schema compDB
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `compDB` ;
-- -----------------------------------------------------
-- Schema compDB
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `compDB` DEFAULT CHARACTER SET utf8 ;
SHOW WARNINGS;
USE `compDB` ;
-- -----------------------------------------------------
-- Table `users_categ`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `users_categ` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `users_categ` (
  `users_categ_id` INT NOT NULL,
  `category` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`users_categ_id`))
ENGINE = InnoDB;

SHOW WARNINGS;
-- -----------------------------------------------------
-- Table `profile`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `profile` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `profile` (
  `profileID` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `photo` VARCHAR(45) NULL,
  `eMail` VARCHAR(45) NULL,
  `contacts` VARCHAR(45) NOT NULL,
  `usersCategID` INT NOT NULL,
  PRIMARY KEY (`profileID`),
  CONSTRAINT `fk_Profile_Users category`
    FOREIGN KEY (`usersCategID`)
    REFERENCES `users_categ` (`users_categ_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;

SHOW WARNINGS;
CREATE INDEX `fk_Profile_Users category_idx` ON `profile` (`usersCategID` ASC);

SHOW WARNINGS;
-- -----------------------------------------------------
-- Table `competition`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `competition` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `competition` (
  `competition_id` INT NOT NULL,
  `place` VARCHAR(45) NOT NULL,
  `date` DATETIME NOT NULL,
  `description` VARCHAR(45) NULL,
  `rewarding` VARCHAR(45) NULL,
  PRIMARY KEY (`competition_id`))
ENGINE = InnoDB;

SHOW WARNINGS;
-- -----------------------------------------------------
-- Table `appForParticip`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `appForParticip` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `appForParticip` (
  `appForParticipID` INT NOT NULL,
  `date` DATE NOT NULL,
  `result` INT NULL,
  `competitionID` INT NOT NULL,
  `usersCategoryID` INT NOT NULL,
  `profileID` INT NOT NULL,
  PRIMARY KEY (`appForParticipID`),
  CONSTRAINT `fk_Application for participation_Competition1`
    FOREIGN KEY (`competitionID`)
    REFERENCES `competition` (`competition_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Application for participation_Users category1`
    FOREIGN KEY (`usersCategoryID`)
    REFERENCES `users_categ` (`users_categ_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Application for participation_Profile1`
    FOREIGN KEY (`profileID`)
    REFERENCES `profile` (`profileID`)
```

```
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

SHOW WARNINGS;
CREATE INDEX `fk_Application for participation_Competition1_idx` ON `appForParticip`
(`competitionID` ASC);

SHOW WARNINGS;
CREATE INDEX `fk_Application for participation_Users category1_idx` ON `appForParticip`
(`usersCategoryID` ASC);

SHOW WARNINGS;
CREATE INDEX `fk_Application for participation_Profile1_idx` ON `appForParticip` (`profileID` ASC);

SHOW WARNINGS;
```

Б) заповнення таблиць даними

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
-- -----------------------------------------------------
-- Data for table `users_categ`
-- -----------------------------------------------------
START TRANSACTION;
USE `compDB`;
INSERT INTO `users_categ` (`users_categ_id`, `category`) VALUES (1, 'participant');
INSERT INTO `users_categ` (`users_categ_id`, `category`) VALUES (2, 'judge');
INSERT INTO `users_categ` (`users_categ_id`, `category`) VALUES (3, 'guest');
INSERT INTO `users_categ` (`users_categ_id`, `category`) VALUES (4, 'organizer');

COMMIT;
-- -----------------------------------------------------
-- Data for table `profile`
-- -----------------------------------------------------
START TRANSACTION;
USE `compDB`;
INSERT INTO `profile` (`profileID`, `name`, `photo`, `eMail`, `contacts`, `usersCategID`) VALUES (1,
'Pavluchkov Vladislav', NULL, NULL, '0631234567', 4);
INSERT INTO `profile` (`profileID`, `name`, `photo`, `eMail`, `contacts`, `usersCategID`) VALUES (2,
'Zmeul Evgeniy', NULL, NULL, '0671234567', 2);
INSERT INTO `profile` (`profileID`, `name`, `photo`, `eMail`, `contacts`, `usersCategID`) VALUES (3,
'Morozov Max', NULL, NULL, '0961234567', 3);
INSERT INTO `profile` (`profileID`, `name`, `photo`, `eMail`, `contacts`, `usersCategID`) VALUES (4,
'Popenko Ruslan', NULL, NULL, '0981234567', 1);
INSERT INTO `profile` (`profileID`, `name`, `photo`, `eMail`, `contacts`, `usersCategID`) VALUES (5,
'Korchak Myhailo', NULL, NULL, '0501234567', 3);
```

```
COMMIT;
-- -----------------------------------------------------
-- Data for table `competition`
-- -----------------------------------------------------
START TRANSACTION;
USE `compDB`;
INSERT INTO `competition` (`competition_id`, `place`, `date`, `description`, `rewarding`) VALUES (1,
'Stadium Start', '2014-11-15 19:00:00', NULL, NULL);
INSERT INTO `competition` (`competition_id`, `place`, `date`, `description`, `rewarding`) VALUES (2,
'Metro Gidropark bus station', '2014-11-19 19:00:00', NULL, NULL);

COMMIT;
-- -----------------------------------------------------
-- Data for table `appForParticip`
-- -----------------------------------------------------
START TRANSACTION;
USE `compDB`;
INSERT INTO `appForParticip` (`appForParticipID`, `date`, `result`, `competitionID`,
`usersCategoryID`, `profileID`) VALUES (1, '2014-11-15', NULL, 1, 1, 1);
INSERT INTO `appForParticip` (`appForParticipID`, `date`, `result`, `competitionID`,
`usersCategoryID`, `profileID`) VALUES (2, '2014-11-19', NULL, 2, 1, 1);
INSERT INTO `appForParticip` (`appForParticipID`, `date`, `result`, `competitionID`,
`usersCategoryID`, `profileID`) VALUES (3, '2014-11-15', NULL, 1, 2, 2);
INSERT INTO `appForParticip` (`appForParticipID`, `date`, `result`, `competitionID`,
`usersCategoryID`, `profileID`) VALUES (4, '2014-11-19', NULL, 2, 2, 2);
INSERT INTO `appForParticip` (`appForParticipID`, `date`, `result`, `competitionID`,
`usersCategoryID`, `profileID`) VALUES (5, '2014-11-15', NULL, 1, 3, 3);
INSERT INTO `appForParticip` (`appForParticipID`, `date`, `result`, `competitionID`,
`usersCategoryID`, `profileID`) VALUES (6, '2014-11-19', NULL, 2, 3, 3);
INSERT INTO `appForParticip` (`appForParticipID`, `date`, `result`, `competitionID`,
`usersCategoryID`, `profileID`) VALUES (7, '2014-11-15', NULL, 1, 4, 4);
INSERT INTO `appForParticip` (`appForParticipID`, `date`, `result`, `competitionID`,
`usersCategoryID`, `profileID`) VALUES (8, '2014-11-19', NULL, 2, 4, 4);

COMMIT;
-- -----------------------------------------------------
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

ДОДАТОК В

HibernateUtil.java

```
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
/**
 * Клас для взаємодії з конфіг файлами і створення обєкту
SessionFactory,котрий відповідає за створення hibernate-сесії
 * @author Руслан Попенко
 * @version 1.0
 * @since 2015-04-14
 */
public class HibernateUtil {
   /**
    * об'єкт SessionFactory,котрий відповідає за створення
hibernate-сесії
    */
   private static SessionFactory sessionFactory;


   private HibernateUtil () {

   }

   /**
    * Створює нову сесію із hibernate.cfg.xml
    */
   static {
      try {
         sessionFactory = new
Configuration().configure().buildSessionFactory();
      } catch (Throwable e){
         throw new ExceptionInInitializerError(e);
      }

   }

   /**
    * Повертає обєкт SessionFactory
    * @return обєкт SessionFactory
    */
   public static SessionFactory getSessionFactory() {
      return sessionFactory;
   }
}
```

AppForParticipDaoImpl.java

```
package daoImpl;
import java.sql.SQLException;
```

```
import java.util.List;

import org.hibernate.Session;

import entity.AppForParticip;

import persistence.HibernateUtil;
import dao.AppForParticipDao;
public class AppForParticipDaoImpl implements AppForParticipDao{
   private int identif;

   @Override
   public void addAppForParticip(AppForParticip appForParticip)
throws SQLException {
      Session session = null;
      try {
          session =
HibernateUtil.getSessionFactory().openSession();
          session.beginTransaction();
          session.save(appForParticip);
          identif=appForParticip.getAppForParticipID();
          session.getTransaction().commit();

      } catch (Exception e){
         e.printStackTrace();
      } finally {
         if ((session!= null) && (session.isOpen()))
            session.close();
      }

   }

   @Override
   public void updateAppForParticip(AppForParticip
appForParticip) throws SQLException {
      Session session = null;
      try {
          session =
HibernateUtil.getSessionFactory().openSession();
          session.beginTransaction();
          session.update(appForParticip);
          session.getTransaction().commit();

      } catch (Exception e){
         e.printStackTrace();
      } finally {
```

```
         if ((session!= null) && (session.isOpen()))
            session.close();
      }

   }

   @Override
   public void deleteAppForParticip(AppForParticip
appForParticip) throws SQLException {
      Session session = null;
      try {
          session =
HibernateUtil.getSessionFactory().openSession();
          session.beginTransaction();
          session.delete(appForParticip);
          session.getTransaction().commit();

      } catch (Exception e){
         e.printStackTrace();
      } finally {
         if ((session!= null) && (session.isOpen()))
            session.close();
      }

   }

   @Override
   public AppForParticip getAppForParticip(int id) throws
SQLException {
      AppForParticip result=null;

      Session session = null;
      try {
          session =
HibernateUtil.getSessionFactory().openSession();
          result = (AppForParticip)
session.get(AppForParticip.class, id);

      } catch (Exception e){
         e.printStackTrace();
      } finally {
         if ((session!= null) && (session.isOpen()))
            session.close();
      }
      return result;
   }
```

```
   @Override
   public List<AppForParticip> getAppForParticips() throws
SQLException {
      List<AppForParticip> appForParticips = null;
      Session session = null;
      try {
          session =
HibernateUtil.getSessionFactory().openSession();
          appForParticips =
session.createCriteria(AppForParticip.class).list();

      } catch (Exception e){
         e.printStackTrace();
      } finally {
         if ((session!= null) && (session.isOpen()))
            session.close();
      }
      return appForParticips;
   }
   @Override
   public int getIdentif () {
      return identif;
   }

}
```

AddAppForPartServlet.java

```
package controller;

import dao.AppForParticipDao;
import dao.CategoryDao;
import dao.CompetitionDao;
import dao.ProfileDao;
import entity.AppForParticip;
import entity.Profile;
import factoryDao.Factory;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
```

```
import java.io.PrintWriter;
import java.sql.SQLException;
import java.sql.Timestamp;

/**
 * Сервлет для взаэмодії зі сторінками /registr.jsp,
/registred.jsp
 * @author Руслан Попенко
 * @version 1.0
 * @since 2015-04-14
 */
@WebServlet("/AppForPartServlet")
public class AddAppForPartServlet extends HttpServlet {

    /**
     * Отримаємо дані про профіль з /registr.jsp,
     * заносимо профіль і заявку в базу даних,
     * і пересилаємо результат реєстрації на /registred.jsp
     * @param request запит
     * @param response відповідь
     * @throws ServletException необхідне виключення
     * @throws IOException необхідне виключення
     */
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        Factory factory=Factory.getInstance();
        ProfileDao profileDao = factory.getProfileDao();
        CategoryDao categoryDao=factory.getCategoryDao();
        CompetitionDao
competitionDao=factory.getCompetitionDao();
        AppForParticipDao
appForParticipDao=factory.getAppForParticipDao();

        String name=request.getParameter("name");
        String photo="photo";
        String eMail=request.getParameter("eMail");
        String contact=request.getParameter("contact");
        Profile profile=new Profile();
        try {
            profile.setCategory(categoryDao.getCategory(3));
        } catch (SQLException e) {
            e.printStackTrace();
        }
        profile.setName(name);
        profile.setPhoto(photo);
```

```
        profile.seteMail(eMail);
        profile.setContacts(contact);

        try {
            profileDao.addProfile(profile);
        } catch (SQLException e) {
            e.printStackTrace();
        }


        String number=request.getParameter("num");
        int num=Integer.parseInt(number);



        AppForParticip appForParticip=new AppForParticip();

        try {

appForParticip.setCategory(categoryDao.getCategory(3));
        } catch (SQLException e) {
            e.printStackTrace();
        }

        appForParticip.setProfile(profile);
        try {

appForParticip.setCompetition(competitionDao.getCompetition(num)
);
        } catch (SQLException e) {
            e.printStackTrace();
        }


        java.util.Date date= new java.util.Date();
        Timestamp current=new Timestamp(date.getTime());
        appForParticip.setDate(current);
        appForParticip.setResult(false);

        try {
            appForParticipDao.addAppForParticip(appForParticip);
        } catch (SQLException e) {
            e.printStackTrace();
        }

        request.setAttribute("id",
```

```
""+appForParticipDao.getIdentif());

request.getRequestDispatcher("/registred.jsp").forward(request,
response);


    }


}
```

CompetServlet.java

```
package controller;

import dao.CompetitionDao;
import entity.Competition;
import factoryDao.Factory;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

/**
 * Сервлет для взаэмодії зі сторінкою /competition.jsp
 * @author Руслан Попенко
 * @version 1.0
 * @since 2015-04-14
 */
@WebServlet("/CompetServlet")
public class CompetServlet extends HttpServlet {

    /**
     * Повертає з бази даних таблицю Змагання
     * пересилає її на /competition.jsp
     * @param request запит
     * @param response відповідь
     * @throws ServletException необхідне виключення
     * @throws IOException необхідне виключення
```

```
     */
    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        Factory factory = Factory.getInstance();
        CompetitionDao competitionDao =
factory.getCompetitionDao();

        List<Competition> compList = null;
        try {
            compList = competitionDao.getCompetitions();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        request.setAttribute("ListOfComp", compList);

request.getRequestDispatcher("/competition.jsp").forward(request
, response);
    }

}
```

DeleteAppForServlet.java

```
package controller;

import dao.AppForParticipDao;
import dao.ProfileDao;
import entity.AppForParticip;
import entity.Profile;
import factoryDao.Factory;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.SQLException;

/**
 * Сервлет для взаэмодії зі сторінками /index.jsp, /profile.jsp
```

```
 * @author Руслан Попенко
 * @version 1.0
 * @since 2015-04-14
 */
@WebServlet("/DeleteAppForServlet")
public class DeleteAppForServlet extends HttpServlet {

    /**
     * Видаляє профіль за бази даних, отриманий з /profile.jsp
     * і напрявляє на /index.jsp
     * @param request запит
     * @param response відповідь
     * @throws ServletException необхідне виключення
     * @throws IOException необхідне виключення
     */
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        Factory factory=Factory.getInstance();
        AppForParticipDao
appForParticipDao=factory.getAppForParticipDao();
        ProfileDao profileDao = factory.getProfileDao();
        String str=request.getParameter("deleted").toString();
        AppForParticip appForParticip=new AppForParticip();

        try {

appForParticip=appForParticipDao.getAppForParticip(Integer.parse
Int(str));
        } catch (SQLException e) {
            e.printStackTrace();
        }
        Profile profile=appForParticip.getProfile();
        try {

appForParticipDao.deleteAppForParticip(appForParticip);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        try {
            profileDao.deleteProfile(profile);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
```

```
            out.println("<script type=\"text/javascript\">");
            out.println("alert('Deleted');");
            out.println("location='index.jsp';");
            out.println("</script>");

        }

}
```

GetProfileServlet.java

```
package controller;

import dao.AppForParticipDao;
import dao.CategoryDao;
import dao.CompetitionDao;
import dao.ProfileDao;
import entity.AppForParticip;
import entity.Profile;
import factoryDao.Factory;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;
import java.sql.Timestamp;

/**
 * Сервлет для взаэмодії зі сторінкою /profile.jsp
 * @author Руслан Попенко
 * @version 1.0
 * @since 2015-04-14
 */
@WebServlet("/GetProfileServlet")
public class GetProfileServlet extends HttpServlet {
    /**
     * Виймає профіль з бази даних і відправляє на /profile.jsp
     * @param request запит
     * @param response відповідь
     * @throws ServletException необхідне виключення
```

```
     * @throws IOException необхідне виключення
     */
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        String usN =
request.getParameter("userName").toString();
        request.setAttribute("usN", usN);
        Factory factory=Factory.getInstance();

        AppForParticipDao
appForParticipDao=factory.getAppForParticipDao();

        AppForParticip ap=new AppForParticip();

        int n=Integer.parseInt(usN);
        try {
            ap=appForParticipDao.getAppForParticip(n);
        } catch (SQLException e) {

request.getRequestDispatcher("/error.jsp").forward(request,
response);
        }
        if (ap==null){

request.getRequestDispatcher("/error.jsp").forward(request,
response);
        }
        Timestamp date=ap.getDate();

        request.setAttribute("TimeOfApp",
date.toLocaleString());

        if(ap.getResult()) {
            request.setAttribute("result",
"\u0417\u0430\u0442\u0432\u0435\u0440\u0434\u0436\u0435\u043d\u0
43e");
        } else {
            request.setAttribute("result", "\u041d\u0435
\u0437\u0430\u0442\u0432\u0435\u0440\u0434\u0436\u0435\u043d\u04
3e");
        }

        String compId=ap.getCompetition().getId()+"";
        String
compDate=ap.getCompetition().getDate().toLocaleString();
```

```
        String compPlace=ap.getCompetition().getPlace();

        request.setAttribute("compId", compId);
        request.setAttribute("compDate", compDate);
        request.setAttribute("compPlace", compPlace);

        Profile profile=ap.getProfile();
        String name=profile.getName();
        String photo=profile.getPhoto();
        String eMail=profile.geteMail();
        String contacts=profile.getContacts();

        request.setAttribute("name", name);
        request.setAttribute("photo", photo);
        request.setAttribute("eMail", eMail);
        request.setAttribute("contacts", contacts);

        String category=ap.getCategory().getCateg();
        if (category.equals("guest")){
            request.setAttribute("category", category);

request.getRequestDispatcher("/profile.jsp").forward(request,
response);

        } else {

request.getRequestDispatcher("/error.jsp").forward(request,
response);
        }


    }
}
```

TransitionServlet.java

```
package controller;

import dao.AppForParticipDao;
import dao.CategoryDao;
import dao.ProfileDao;
import entity.AppForParticip;
import entity.Category;
```

```
import entity.Profile;
import factoryDao.Factory;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.SQLException;

/**
 * Сервлет для взаэмодії зі сторінкою /profile.jsp
 * @author Руслан Попенко
 * @version 1.0
 * @since 2015-04-14
 */
@WebServlet("/TransitionServlet")
public class TransitionServlet extends HttpServlet {
    /**
     * Поле ідентиіфікатор заявки
     */
    private static int n;

    /**
     * Геттер для ідентиіфікатора заявки
     * @return ідентиіфікатор заявки
     */
    public static int getNumberOfApp (){
        return n;
    }

    /**
     * Здійснює перехід з гостя до іншої категорії
     * @param request запит
     * @param response відповідь
     * @throws ServletException необхідне виключення
     * @throws IOException необхідне виключення
     */
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        Factory factory=Factory.getInstance();
        AppForParticipDao
```

```
appForParticipDao=factory.getAppForParticipDao();
        ProfileDao profileDao = factory.getProfileDao();
        CategoryDao categoryDao=factory.getCategoryDao();
        String
str=request.getParameter("numberOfApp").toString();
        n=Integer.parseInt(str);
        AppForParticip appForParticip=new AppForParticip();
        Category category=new Category();

        try {

appForParticip=appForParticipDao.getAppForParticip(Integer.parse
Int(str));
        } catch (SQLException e) {
            e.printStackTrace();
        }
        Profile profile=appForParticip.getProfile();

        String selectedValue=request.getParameter("catOptions");
        if(selectedValue.equals("1"))
        {
            try {
                category=categoryDao.getCategory(1);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            appForParticip.setCategory(category);
            profile.setCategory(category);
            try {

appForParticipDao.updateAppForParticip(appForParticip);
                profileDao.updateProfile(profile);
            } catch (SQLException e) {
                e.printStackTrace();
            }

        }
        else if(selectedValue.equals("2"))
        {
            try {
                category=categoryDao.getCategory(2);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            appForParticip.setCategory(category);
            profile.setCategory(category);
```

```
            try {

appForParticipDao.updateAppForParticip(appForParticip);
                profileDao.updateProfile(profile);
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        else if(selectedValue.equals("4"))
        {
            try {
                category=categoryDao.getCategory(4);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            appForParticip.setCategory(category);
            profile.setCategory(category);
            try {

appForParticipDao.updateAppForParticip(appForParticip);
                profileDao.updateProfile(profile);
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }


        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        out.println("<script type=\"text/javascript\">");
        out.println("alert('Category changed');");
        out.println("location='index.jsp';");
        out.println("</script>");

    }

}
```

Factory.java

```
package factoryDao;

import dao.AppForParticipDao;
import dao.CategoryDao;
import dao.CompetitionDao;
import dao.ProfileDao;
import daoImpl.AppForParticipDaoImpl;
```

```
import daoImpl.CategoryDaoImpl;
import daoImpl.CompetitionDaoImpl;
import daoImpl.ProfileDaoImpl;


public class Factory {
   public static Factory instance = new Factory();
   private ProfileDao profileDao;
   private CategoryDao categoryDao;
   private CompetitionDao competitionDao;
   private AppForParticipDao appForParticipDao;

   private Factory (){

   }

   public static Factory getInstance(){
      return Factory.instance;
   }

   public ProfileDao getProfileDao (){
      if (profileDao == null) {
         profileDao = new ProfileDaoImpl();
      }
      return profileDao;
   }

   public CategoryDao getCategoryDao (){
      if (categoryDao == null) {
         categoryDao = new CategoryDaoImpl();
      }
      return categoryDao;
   }

   public CompetitionDao getCompetitionDao (){
      if (competitionDao == null) {
         competitionDao = new CompetitionDaoImpl();
      }
      return competitionDao;
   }
   public AppForParticipDao getAppForParticipDao (){
      if (appForParticipDao == null) {
         appForParticipDao = new AppForParticipDaoImpl();
      }
      return appForParticipDao;   }
```