

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**ЛАБОРАТОРНА РОБОТА № 12**

**з кредитного модуля «Проектування корпоративних інформаційних систем» на тему:**

«Створення DAO і DAOFactory в проектах з JPA»

Виконав студент групи КП-61с

Литвин Вадим Олександрович

Київ 2017

**Мета роботи**: Розробка та імплементація інтерфейсів DAO. Використання шаблону проектування DAOFactory.

Тема «ІС хлібозаводу».

Вихідний код проекту: <https://github.com/VadyaVL/CIS-Bakery>

**Хід роботи**

Для виконання поточної лабораторної роботи використовуємо базу даних з попередньої, одинадцятої, лабораторної роботи. В структуру проекту було додано незначні зміни. В пакет ua.bakery.logic було додано новий клас Lab12.class в якому відбувалася реалізація поставленого в поточній лабораторній роботі завдання. В деяких класах було перевизначено метод toString(). Також додано пакети для DAO шаблону.

|  |
| --- |
|  |

Рис. 1 Структура jdbc та JPA проекту «ІС хлібозаводу»

Використовувана СУБД в даній роботі – MySQL.

На рис. 1 відображено структуру проекту даної лабораторної роботи.

Демонстрація роботи програми (вивід M-to-M прайс-листа):

Місто: Рокитне- доставка

Хліб Пшеничний (ФОРМОВИЙ) 0,5 1.0

Хліб Пшеничний (ФОРМОВИЙ) 0,5 різ. 2.0

Хліб Пшеничний (ФОРМОВИЙ) 0,9 3.0

Хліб Пшеничний (ФОРМОВИЙ) 0,8 6.0

Хліб арнаут КиІвський 0,6 5.0

Хліб арнаут 0,6 1.0

Місто: Села

Хліб Пшеничний (ФОРМОВИЙ) 0,5 47.0

Хліб Пшеничний (ФОРМОВИЙ) 0,5 різ. 8.0

Хліб Пшеничний (ФОРМОВИЙ) 0,9 9.0

Хліб Пшеничний (ФОРМОВИЙ) 0,8 5.0

Хліб арнаут КиІвський 0,6 4.0

Хліб арнаут 0,6 6.0

Місто: Біла Церква-День

Хліб Пшеничний (ФОРМОВИЙ) 0,5 2.0

Хліб Пшеничний (ФОРМОВИЙ) 0,5 різ. 1.0

Хліб Пшеничний (ФОРМОВИЙ) 0,9 4.0

Хліб Пшеничний (ФОРМОВИЙ) 0,8 5.0

Хліб арнаут КиІвський 0,6 6.0

Хліб арнаут 0,6 4.0

Місто: Рокитне з хлібом Б.Ц.

Хліб Пшеничний (ФОРМОВИЙ) 0,5 7.0

Хліб Пшеничний (ФОРМОВИЙ) 0,5 різ. 8.0

Хліб Пшеничний (ФОРМОВИЙ) 0,9 9.0

Хліб Пшеничний (ФОРМОВИЙ) 0,8 6.0

Хліб арнаут КиІвський 0,6 5.0

Хліб арнаут 0,6 2.0

Місто: Кагарлик ринок

Хліб Пшеничний (ФОРМОВИЙ) 0,5 1.0

Хліб Пшеничний (ФОРМОВИЙ) 0,5 різ. 3.0

Хліб Пшеничний (ФОРМОВИЙ) 0,9 6.0

Хліб Пшеничний (ФОРМОВИЙ) 0,8 5.0

Хліб арнаут КиІвський 0,6 4.0

Хліб арнаут 0,6 7.0

Місто: Магазини

Хліб Пшеничний (ФОРМОВИЙ) 0,5 8.0

Хліб Пшеничний (ФОРМОВИЙ) 0,5 різ. 9.0

Хліб Пшеничний (ФОРМОВИЙ) 0,9 5.0

Хліб Пшеничний (ФОРМОВИЙ) 0,8 1.0

Хліб арнаут КиІвський 0,6 2.0

Хліб арнаут 0,6 3.0

Місто: КиІв

Дані відсутні

Місто: ПТУ

Дані відсутні

Місто: Фастов - Кагарлик- Б.Ц. Ніч

Дані відсутні

Нижче наведемо лістинг деяких файлів з проекту. Повний код лабораторної роботи можна побачити за посилання вказаним на початку даного документу.

|  |
| --- |
| Лістинг 1. Клас Lab12.java |
| **package** ua.bakery.logic;  **import** java.util.List;  **import** ua.bakery.db.DAO.factory.DAOFactory;  **import** ua.bakery.db.jpa.Client;  **import** ua.bakery.db.jpa.Place;  **import** ua.bakery.db.jpa.ProductPrice;  /\*\*  \* Клас для реалізації завдання з лабораторної роботи №12  \* **@author** Vadym  \*  \*/  **public** **class** Lab12 {    **public** **static** **void** main(String[] args) {  String pattern = "%-40s %10s\n";  System.***out***.println("Price-list на продукцію по містам:");  List<Place> places = (List<Place>) DAOFactory.*getInstance*().getPlaceDAO().getAllPlace();    **for**(Place place : places){  System.***out***.println("\nМісто: " + place.getName());    **if**(place.getProductionPrice().size() == 0 )  System.***out***.println("Дані відсутні");    **for**(ProductPrice pp : place.getProductionPrice()){  System.***out***.format(pattern,  pp.getProduction().getName(),  Float.*toString*(pp.getPrice()));  }  }  }  } |

|  |
| --- |
| Лістинг 2. Клас DAOFactory.java |
| **package** ua.bakery.db.DAO.factory;  **import** ua.bakery.db.DAO.\*;  **import** ua.bakery.db.DAO.impl.\*;  **public** **class** DAOFactory {    **private** **static** DAOFactory *\_instance* = **null**;  **private** IBrakDAO brakDAO = **null**;  **private** IClientDAO clientDAO = **null**;  **private** IEmployeeDAO employeeDAO = **null**;  **private** IIngridientDAO ingridientDAO = **null**;  **private** ILogDAO logDAO = **null**;  **private** IOldNewReportDAO oldNewReportDAO = **null**;  **private** IPlaceDAO placeDAO = **null**;  **private** IProducedDAO producedDAO = **null**;  **private** IProductionDAO productionDAO = **null**;  **private** IProductionIngridientDAO productionIngridientDAO = **null**;  **private** IProductionOldNewReportDAO productionOldNewReportDAO = **null**;  **private** IProductionReportTeamDAO productionReportTeamDAO = **null**;  **private** IProductPriceDAO productionPriceDAO = **null**;  **private** IRequestDAO requestDAO = **null**;  **private** IRequestProductionDAO requestProductionDAO = **null**;  **private** IUserDAO userDAO = **null**;  **private** IWriteOffDAO writeOffDAO = **null**;    **private** DAOFactory(){    }    **public** **static** DAOFactory getInstance(){  **if**(*\_instance* == **null**)  *\_instance* = **new** DAOFactory();    **return** *\_instance*;  }  **public** IBrakDAO getBrakDAO() {  **if**(brakDAO == **null**)  brakDAO = **new** BrakDAOImpl();  **return** brakDAO;  }  **public** IClientDAO getClientDAO() {  **if**(clientDAO == **null**)  clientDAO = **new** ClientDAOImpl();  **return** clientDAO;  }  **public** IEmployeeDAO getEmployeeDAO() {  **if**(employeeDAO == **null**)  employeeDAO = **new** EmployeeDAOImpl();  **return** employeeDAO;  }  **public** IIngridientDAO getIngridientDAO() {  **if**(ingridientDAO == **null**)  ingridientDAO = **new** IngridientDAOImpl();  **return** ingridientDAO;  }  **public** ILogDAO getLogDAO() {  **if**(logDAO == **null**)  logDAO = **new** LogDAOImpl();  **return** logDAO;  }  **public** IOldNewReportDAO getOldNewReportDAO() {  **if**(oldNewReportDAO == **null**)  oldNewReportDAO = **new** OldNewReportDAOImpl();  **return** oldNewReportDAO;  }  **public** IPlaceDAO getPlaceDAO() {  **if**(placeDAO == **null**)  placeDAO = **new** PlaceDAOImpl();  **return** placeDAO;  }  **public** IProducedDAO getProducedDAO() {  **if**(producedDAO == **null**)  producedDAO = **new** ProducedDAOImpl();  **return** producedDAO;  }  **public** IProductionDAO getProductionDAO() {  **if**(productionDAO == **null**)  productionDAO = **new** ProductionDAOImpl();  **return** productionDAO;  }  **public** IProductionIngridientDAO getProductionIngridientDAO() {  **if**(productionIngridientDAO == **null**)  productionIngridientDAO = **new** ProductionIngridientDAOImpl();  **return** productionIngridientDAO;  }  **public** IProductionOldNewReportDAO getProductionOldNewReportDAO() {  **if**(productionOldNewReportDAO == **null**)  productionOldNewReportDAO = **new** ProductionOldNewReportDAOImpl();  **return** productionOldNewReportDAO;  }  **public** IProductionReportTeamDAO getProductionReportTeamDAO() {  **if**(productionReportTeamDAO == **null**)  productionReportTeamDAO = **new** ProductionReportTeamDAOImpl();  **return** productionReportTeamDAO;  }  **public** IProductPriceDAO getProductionPriceDAO() {  **if**(productionPriceDAO == **null**)  productionPriceDAO = **new** ProductPriceDAOImpl();  **return** productionPriceDAO;  }  **public** IRequestDAO getRequestDAO() {  **if**(requestDAO == **null**)  requestDAO = **new** RequestDAOImpl();  **return** requestDAO;  }  **public** IRequestProductionDAO getRequestProductionDAO() {  **if**(requestProductionDAO == **null**)  requestProductionDAO = **new** RequestProductionDAOImpl();  **return** requestProductionDAO;  }  **public** IUserDAO getUserDAO() {  **if**(userDAO == **null**)  userDAO = **new** UserDAOImpl();  **return** userDAO;  }  **public** IWriteOffDAO getWriteOffDAO() {  **if**(writeOffDAO == **null**)  writeOffDAO = **new** WriteOffDAOImpl();  **return** writeOffDAO;  }  } |

|  |
| --- |
| Лістинг 3. Клас IClientDAO.java |
| **package** ua.bakery.db.DAO;  **import** java.util.Collection;  **import** ua.bakery.db.jpa.Client;  **import** ua.bakery.db.jpa.Place;  /\*\*  \* **@author** vadym  \* **@since** 0.12  \*/  **public** **interface** IClientDAO {  **public** **void** saveClient(Client client);  **public** **void** updateClient(Client client);  **public** Client getClientById(Integer client\_id);  **public** Collection<Client> getClientsByPlace(Place place);  **public** Integer getAllClientsCount();  **public** Collection<Client> getAllClients();  **public** **void** deleteClient(Client client);  } |

|  |
| --- |
| Лістинг 4. Клас ClientDAOImpl.java |
| **package** ua.bakery.db.DAO.impl;  **import** ua.bakery.db.DAO.IClientDAO;  **import** ua.bakery.db.jpa.Client;  **import** ua.bakery.db.jpa.Place;  **import** java.util.Collection;  **import** java.util.List;  **import** javax.persistence.EntityManager;  **import** javax.persistence.EntityManagerFactory;  **import** javax.persistence.Persistence;  **import** javax.persistence.Query;  /\*\*  \* **@author** vadym  \* **@since** 0.12  \*/  **public** **class** ClientDAOImpl **implements** IClientDAO {  **private** **static** **final** String ***PERSISTENCE\_UNIT\_NAME*** = "CISBakeryJPA";  **private** EntityManagerFactory entityMF = **null**;  **private** EntityManager entityMng = **null**;  **public** ClientDAOImpl() {  entityMF = Persistence.*createEntityManagerFactory*(***PERSISTENCE\_UNIT\_NAME***);  }  @Override  **public** **void** saveClient(Client client) {  **try** {  entityMng = entityMF.createEntityManager();  entityMng.getTransaction().begin();  entityMng.persist(client); // Сохранение экземпляра класса  entityMng.getTransaction().commit();  } **catch** (Exception e) {  **new** Exception("Помилка при збереженні!", e.getCause());  } **finally** {  **if** (entityMng != **null** && entityMng.isOpen()) {  entityMng.close();  }  }  }  @Override  **public** **void** updateClient(Client client) {  **try** {  entityMng = entityMF.createEntityManager();  entityMng.getTransaction().begin();  entityMng.merge(client);  entityMng.getTransaction().commit();  } **catch** (Exception e) {  **new** Exception("Помилка при оновленні!", e.getCause());  } **finally** {  **if** (entityMng != **null** && entityMng.isOpen()) {  entityMng.close();  }  }  }  @Override  **public** Client getClientById(Integer client\_id) {  Client client = **null**;  **try** {  entityMng = entityMF.createEntityManager();  entityMng.getTransaction().begin();  client = entityMng.find(Client.**class**, client\_id);  entityMng.getTransaction().commit();  } **catch** (Exception e) {  **new** Exception("Помилка 'findById'", e.getCause());  } **finally** {  **if** (entityMng != **null** && entityMng.isOpen()) {  entityMng.close();  }  }  **return** client;  }  @Override  **public** Integer getAllClientsCount() {  Integer cnt = **null**;  **try** {  entityMng = entityMF.createEntityManager();  entityMng.getTransaction().begin();  Query query = entityMng.createQuery("SELECT COUNT c FROM Client c");  cnt = (Integer) query.getSingleResult();  entityMng.getTransaction().commit();  } **catch** (Exception e) {  **new** Exception("Помилка ", e.getCause());  } **finally** {  **if** (entityMng != **null** && entityMng.isOpen()) {  entityMng.close();  }  }  **return** cnt;  }  @SuppressWarnings("unchecked")  @Override  **public** Collection<Client> getAllClients() {  List<Client> clients = **null**;  **try** {  entityMng = entityMF.createEntityManager();  entityMng.getTransaction().begin();  Query query = entityMng.createQuery("SELECT c from Client c");  clients = (List<Client>)query.getResultList();  entityMng.getTransaction().commit();  } **catch** (Exception e) {  **new** Exception("Помилка 'getAll'", e.getCause());  } **finally** {  **if** (entityMng != **null** && entityMng.isOpen()) {  entityMng.close();  }  }  **return** clients;  }  @Override  **public** **void** deleteClient(Client client) {  **try** {  entityMng = entityMF.createEntityManager();  entityMng.getTransaction().begin();  entityMng.remove(client);  entityMng.getTransaction().commit();  } **catch** (Exception e) {  **new** Exception("Помилка при видаленні!", e.getCause());  } **finally** {  **if** (entityMng != **null** && entityMng.isOpen()) {  entityMng.close();  }  }  }  @Override  **public** Collection<Client> getClientsByPlace(Place place) {  List<Client> clients = **null**;  **try** {  entityMng = entityMF.createEntityManager();  entityMng.getTransaction().begin();  Query query = entityMng.createQuery("SELECT c from Client c WHERE c.place = :place");  clients = (List<Client>)query.setParameter("place", place).getResultList();  entityMng.getTransaction().commit();  } **catch** (Exception e) {  **new** Exception("Помилка 'getAll'", e.getCause());  } **finally** {  **if** (entityMng != **null** && entityMng.isOpen()) {  entityMng.close();  }  }  **return** clients;  }  } |