

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**ЛАБОРАТОРНА РОБОТА № 14**

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

«Створення GenericDAO в проектах з JPA»

Виконав студент групи КП-61с

Литвин Вадим Олександрович

Київ 2017

**Мета роботи**: Розробка та імплементація інтерфейсів GenericDAO і DomainSuperClass.

Тема «ІС хлібозаводу».

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

**Хід роботи**

Для виконання поточної лабораторної роботи використовуємо базу даних з попередньої, тринадцятої, лабораторної роботи. В структуру проекту було додано незначні зміни. В пакет ua.bakery.logic було додано новий клас Lab14.class в якому відбувалася реалізація поставленого в поточній лабораторній роботі завдання. В деяких класах було перевизначено метод toString(). Також додано нові класи та інтерфейси для GenericDAO.

|  |
| --- |
|  |

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

Використовувана СУБД в даній роботі – MySQL.

На рис. 1 відображено структуру проекту даної лабораторної роботи.

Демонстрація роботи програми (вивідсписку клієнтів та M-to-M прайс-листа):

Всі клієнти:

"Кузьменко" Рокитне- доставка

Тутченко Рокитне- доставка

Магазин Бам Рокитне- доставка

Шевченко Рокитне з хлібом Б.Ц.

Подорожня Рокитне з хлібом Б.Ц.

Магазин Ірина Рокитне з хлібом Б.Ц.

Люда Базар Рокитне- доставка

Магазин Лідія Рокитне- доставка

Бусол Села

Сенявське-Першотравневе Села

Пономаренко Села

Калинівка Боровик Села

Калинівка Синявське Села

Мухіна Сільпо№1 Біла Церква-День

Мухіна Сільпо№2 Біла Церква-День

Піонерська Базар Біла Церква-День

Свіжий хліб Біла Церква-День

Арка11111 Біла Церква-День

Мухіна Торговий Центр Біла Церква-День

Курсова Біла Церква-День

Магазин Снайпер Біла Церква-День

Жуковський Біла Церква-День

Леван Базар Біла Церква-День

Посьолок Біла Церква-День

Піонерська Біла Церква-День

Таращанська Біла Церква-День

Критий ринок Біла Церква-День

Некрасова №2 Біла Церква-День

Вокзальна Біла Церква-День

Супрун Біла Церква-День

Курочкіна Біла Церква-День

Магазин Юліана Біла Церква-День

Село Поправка Біла Церква-День

100 Метровка Біла Церква-День

Некрасова 99 Біла Церква-День

Торговий Центр Біла Церква-День

Магазин Рось Біла Церква-День

Магазин Ваді Рокитне з хлібом Б.Ц.

Воловик Рокитне- доставка

Кагарлик Ринок Кагарлик ринок

Фастов Фастов - Кагарлик- Б.Ц. Ніч

Каменотрус Фастов - Кагарлик- Б.Ц. Ніч

Валя Ніч Фастов - Кагарлик- Б.Ц. Ніч

Магазин Центр Магазини

Магазин Пекарня Магазини

Агрофірма Колгоспи

Колгосп Нива Колгоспи

Дитячий будинок Світанок Колгоспи

Колгосп Росія Колгоспи

Чебурашка Колгоспи

Дитсад. Насташка Колгоспи

Дитсад.Остров Колгоспи

Антон Фастов - Кагарлик- Б.Ц. Ніч

Синявський Фастов - Кагарлик- Б.Ц. Ніч

Тесленко Тесленко

Тимофій З Пекарні

Сущев З Пекарні

Ватутіно З Пекарні

Рухаленко З Пекарні

Григоренко З Пекарні

Соловйов З Пекарні

Звенигородка З Пекарні

Кривенкова З Пекарні

Городня З Пекарні

Будова З Пекарні

Кусливий З Пекарні

Тетіїв З Пекарні

Салиха З Пекарні

Левада З Пекарні

Зайцева КиІв

Куручекова КиІв

Сидорчук 2 КиІв

Василик КиІв

Гриценко КиІв

Зайцева 2 КиІв

Румянцева КиІв

Шевченко КиІв

Соловей КиІв

Солдатенко КиІв

Сидорчук 1 КиІв

Мілікян КиІв

Юхимець КиІв

Куцуляк КиІв

Богуслав Богуслав

Богуслав№2 Богуслав

Миронівка Богуслав

Тараща Тараща

Горлиця Рокитне- доставка

Іванова Рокитне- доставка

Паша Коровін З Пекарні

Хіцька З Пекарні

Воловик Ромашки Села

Піонерська Ларьок Біла Церква-День

Зарічна З Пекарні

Мухіна Піонерська Біла Церква-День

Дитячий будинок Рокитне Колгоспи

клієнт\_\_\_\_\_\_\_\_\_\_\_ З Пекарні

на ЗП З Пекарні

Брак Зміни З Пекарні

Залишок Галі Залишок Зміни(остаток в к.д.)

Мухіна Сільпо Гребінки Біла Церква-День

Петюх З Пекарні

Саша Ніч Фастов - Кагарлик- Б.Ц. Ніч

Магазин Продукти З Пекарні

калинівка вергун Села

вергун першотравневе Села

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Прайс-лист по містам:

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: 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: ЗП,Списано,Брак

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 Lab14.java |
| **package** ua.bakery.logic;  **import** javax.persistence.Persistence;  **import** ua.bakery.db.DAO.IGenericDAO;  **import** ua.bakery.db.DAO.impl.GenericDAOImpl;  **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 №14  \* **@author** Vadym  \*/  **public** **class** Lab14 {  **public** **static** **void** main(String[] args) {  IGenericDAO<Client> clientDAO = **new** GenericDAOImpl<Client>(Client.**class**,  Persistence.*createEntityManagerFactory*("CISBakeryJPA"));  IGenericDAO<Place> placesDAO = **new** GenericDAOImpl<Place>(Place.**class**,  Persistence.*createEntityManagerFactory*("CISBakeryJPA"));      String pattern = "%-35s %35s\n";  System.***out***.println("Всі клієнти:");  **for**(Client client : clientDAO.getAll()){  System.***out***.format(pattern, client.getName(), client.getPlace().getName());  }  System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.***out***.println("Прайс-лист по містам:");    **for**(Place place : placesDAO.getAll()){  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 Client.java |
| **package** ua.bakery.db.jpa;  **import** java.io.Serializable;  **import** java.util.ArrayList;  **import** java.util.List;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.ManyToOne;  **import** javax.persistence.OneToMany;  **import** javax.persistence.Table;  @Entity  @Table(name = "Client")  **public** **class** Client **extends** DomainSuperClass **implements** Serializable {    @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** **int** id;  **private** String name;    @ManyToOne  **private** Place place;    @OneToMany(mappedBy="Client")  **private** List<Request> requests = **new** ArrayList<>();    @Override  **public** String toString(){  **return** /\*"Клієнт №" + id + ": " + name + ", Місто: " + this.place.getName()\*/ **this**.name;  }  **public** **int** getId() {  **return** id;  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** Place getPlace() {  **return** place;  }  **public** **void** setPlace(Place place) {  **this**.place = place;  }  **public** List<Request> getRequests() {  **return** requests;  }  **public** **void** setRequests(List<Request> requests) {  **this**.requests = requests;  }  } |

|  |
| --- |
| Лістинг 3. Інтерфейс IGenericDAO<T>.java |
| **package** ua.bakery.db.DAO;  **import** java.util.Collection;  /\*\*  \* Общие методы для всех сущностей вынесены в интерфейс IGenericDAO  \* **@author** Vadym  \*  \* **@param** <T>  \*/  **public** **interface** IGenericDAO<T> {  /\*\*  \* Сохраняем новый объект в базу данных.  \* **@param** entity Объект для сохранения  \*/  **public** **void** save(T entity);  /\*\*  \* Обновляем объект в базе данных.  \* **@param** entity Объект для обновления  \*/  **public** **void** update(T entity);  /\*\*  \* Удаление объекта из базы данных  \* **@param** entity Объект для удаления  \*/  **public** **void** delete(T entity);  /\*\*  \* Находим объект по его уникальному идентификатору  \* **@param** entityId Уникальный идентификатор объекта  \* **@return** Найденный объект или <code>null</code>  \*/  **public** T getById(Integer entityId);  /\*\*  \* Получаем количество записей в БД  \* **@return** Количество записей в БД  \*/  **public** Integer getAllCount();  /\*\*  \* Получаем коллекцию всех объектов  \* **@return** Коллекция всех объектов  \*/  **public** Collection<T> getAll();  } |

|  |
| --- |
| Лістинг 4. Інтерфейс IClientDAOjava |
| **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 **extends** IGenericDAO<Client> {  **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 5. Клас GenericDAOImpl<T> java |
| **package** ua.bakery.db.DAO.impl;  **import** java.util.Collection;  **import** java.util.List;  **import** javax.persistence.EntityManager;  **import** javax.persistence.EntityManagerFactory;  **import** javax.persistence.Query;  **import** ua.bakery.db.DAO.IGenericDAO;  /\*\*  \* Реализация общих методов для всех сущностей  \* в интерфейсе IGenericDAO  \* **@param** <T>  \*/  **public** **class** GenericDAOImpl<T> **implements** IGenericDAO<T> {  **private** EntityManagerFactory emf = **null**;  **private** EntityManager em = **null**;  /\*\*  \* Создание ссылки Class<T> persistentClass. Эта ссылка используется  \* для указания на персистенс класс, передаваемый в качестве параметра T  \*/  **protected** Class<T> persistentClass;  /\*\*  \* В тексте запроса тип представлен подстановочной строкой %s,  \* которая обрабатывается с помощью String.format  \*/  //Запросы настраиваются с помощью String.format  **private** **static** **final** String ***QUERY\_SELECT\_ALL*** = "SELECT x FROM %s x";  **private** **static** **final** String ***QUERY\_COUNT\_ALL*** = "SELECT COUNT(x) FROM %s x";  /\*\*  \* EntityManagerFactory создается в DAOFactory и передается классу  \* через параметр конструктора.  \* Конструктор с 2-мя полями  \* Класс, который будет использовать этот DAO  \* **@param** persistentClass Используемый EntityManager  \* **@param** emf  \*/  **public** GenericDAOImpl (Class<T> persistentClass, EntityManagerFactory emf) {  **super**(); // Чей super конструктор ???  **this**.persistentClass = persistentClass;  **this**.emf = emf; // Создаем EntityManager для работы с сущностями класса  }  **public** **void** save(T entity) {  **if** (entity == **null**) {  System.***out***.println("Entity для сохранения не может быть null!");  }  **try** {  em = emf.createEntityManager();  em.getTransaction().begin();  em.persist(entity); // Сохранение экземпляра класса  // Сохранить объект в БД можно и с помощью JPQL  em.getTransaction().commit();  } **catch** (Exception e) {  System.***out***.println("Ошибка сохранения");  em.getTransaction().rollback();  } **finally** {  //Хорошая практика закрыть entity manager  **if** (em != **null** && em.isOpen()) {  em.close();  }  }  }  **public** **void** update(T entity) {  **if** (entity == **null**) {  System.***out***.println("Entity для обновления не может быть null!");  }  **try** {  em = emf.createEntityManager();  em = emf.createEntityManager();  em.getTransaction().begin();  em.merge(entity); // Обновление сущности в БД  // Обновить объект в БД можно и с помощью JPQL  em.getTransaction().commit();  } **catch** (Exception e) {  System.***out***.println("Ошибка при вставке");  } **finally** {  **if** (em != **null** && em.isOpen()) {  em.close();  }  }  }  **public** **void** delete(T entity) {  **if** (entity == **null**) {  System.***out***.println("Entity для удаления не может быть null!");  }  **try** {  em = emf.createEntityManager();  em.getTransaction().begin();  em.remove(entity);  em.getTransaction().commit();  } **catch** (Exception e) {  System.***out***.println("Ошибка при удалении");  em.getTransaction().rollback();  } **finally** {  **if** (em != **null** && em.isOpen()) {  em.close();  }  }  }  **public** **void** delete(Integer entityId) {  delete(getById(entityId));  }  **public** T getById(Integer entityId) {  T entity = **null**;  **if** (entityId == **null**) {  System.***out***.println("Id сущности для извлечения null!");  }  **try** {  em = emf.createEntityManager();  em.getTransaction().begin();  entity = (T) em.find(persistentClass, entityId);  } **catch** (Exception e) {  System.***out***.println("Ошибка 'findById'");  em.getTransaction().rollback();  } **finally** {  **if** (em != **null** && em.isOpen()) {  em.close();  }  }  **return** entity;  }  **public** Integer getAllCount() {  Integer cnt = **null**;  **try** {  em = emf.createEntityManager();  em.getTransaction().begin();  Query query = em.createQuery(String.*format*(***QUERY\_COUNT\_ALL***, persistentClass.getSimpleName()));  cnt = (Integer) query.getSingleResult();  em.getTransaction().commit();  } **catch** (Exception e) {  System.***out***.println("Ошибка getAllCount");  } **finally** {  **if** (em != **null** && em.isOpen()) {  em.close();  }  }  **return** cnt;  }  **public** Collection<T> getAll() {  List<T> entites = **null**;  **try** {  em = emf.createEntityManager();  em.getTransaction().begin();  Query query = em.createQuery(String.*format*(***QUERY\_SELECT\_ALL***, persistentClass.getSimpleName()));  entites = (List<T>) query.getResultList();  em.getTransaction().commit();  } **catch** (Exception e) {  System.***out***.println("ООшибка 'getAll'");  } **finally** {  **if** (em != **null** && em.isOpen()) {  em.close();  }  }  **return** entites;  }  } |

|  |
| --- |
| Лістинг 6. Інтерфейс 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 **extends** GenericDAOImpl<Client> **implements** IClientDAO {  **private** **static** **final** String ***PERSISTENCE\_UNIT\_NAME*** = "CISBakeryJPA";  **private** EntityManagerFactory entityMF = **null**;  **private** EntityManager entityMng = **null**;    **public** ClientDAOImpl() {  **this**(Persistence.*createEntityManagerFactory*(***PERSISTENCE\_UNIT\_NAME***));  }    **public** ClientDAOImpl (EntityManagerFactory emf) {  **super**(Client.**class**, emf); //Р—Р°РїСѓСЃРєР°РµРј РєРѕРЅСЃС‚СЂСѓРєС‚РѕСЂ РґР»СЏ GenericDAOImpl  **this**.entityMF = emf;  }  @Override  **public** **void** save(Client entity) {  **this**.saveClient(entity);  }  @Override  **public** **void** update(Client entity) {  **this**.updateClient(entity);  }  @Override  **public** **void** delete(Client entity) {  **this**.deleteClient(entity);  }  @Override  **public** Client getById(Integer entityId) {  **return** **this**.getClientById(entityId);  }  @Override  **public** Integer getAllCount() {  **return** **this**.getAllClientsCount();  }  @Override  **public** Collection<Client> getAll() {  **return** **this**.getAllClients();  }  @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;  }  } |