User . CRUD

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

public class UserCRUD {

private EntityManagerFactory emf = Persistence.createEntityManagerFactory("my-persistence-unit");

private EntityManager em = emf.createEntityManager();

// CREATE: Add a new User

public void addUser(User user) {

em.getTransaction().begin(); // Start transaction

em.persist(user); // Add the User object to the database

em.getTransaction().commit(); // Commit transaction to save changes

System.out.println("User added successfully: " + user);

}

// READ: Find a User by ID

public User getUser(String userId) {

User user = em.find(User.class, userId); // Retrieve User by primary key

if (user != null) {

System.out.println("User found: " + user);

} else {

System.out.println("User with ID " + userId + " not found.");

}

return user;

}

// UPDATE: Update User details

public void updateUser(String userId, String newPassword, String newLoginStatus) {

User user = em.find(User.class, userId); // Find the User

if (user != null) {

em.getTransaction().begin(); // Start transaction

user.setPassword(newPassword); // Update password

user.setLoginStatus(newLoginStatus); // Update login status

em.getTransaction().commit(); // Save changes

System.out.println("User updated successfully: " + user);

} else {

System.out.println("User with ID " + userId + " not found. Update failed.");

}

}

// DELETE: Remove a User by ID

public void deleteUser(String userId) {

User user = em.find(User.class, userId); // Find the User

if (user != null) {

em.getTransaction().begin(); // Start transaction

em.remove(user); // Remove User from database

em.getTransaction().commit(); // Commit transaction

System.out.println("User deleted successfully with ID: " + userId);

} else {

System.out.println("User with ID " + userId + " not found. Deletion failed.");

}

}

// CLOSE: Clean up resources

public void close() {

em.close();

emf.close();

}

}

Customer.CRUD

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

public class CustomerCRUD {

private EntityManagerFactory emf = Persistence.createEntityManagerFactory("my-persistence-unit");

private EntityManager em = emf.createEntityManager();

// CREATE: Add a new Customer

public void addCustomer(Customer customer) {

em.getTransaction().begin(); // Start transaction

em.persist(customer); // Add the Customer object to the database

em.getTransaction().commit(); // Commit the transaction

System.out.println("Customer added successfully: " + customer);

}

// READ: Find a Customer by Name

public Customer getCustomer(String customerName) {

Customer customer = em.find(Customer.class, customerName); // Retrieve customer

if (customer != null) {

System.out.println("Customer found: " + customer);

} else {

System.out.println("Customer with name " + customerName + " not found.");

}

return customer;

}

// UPDATE: Update Customer details

public void updateCustomer(String customerName, String newAddress, String newEmail, String newShippingInfo) {

Customer customer = em.find(Customer.class, customerName); // Find customer

if (customer != null) {

em.getTransaction().begin(); // Start transaction

customer.setAddress(newAddress); // Update address

customer.setEmail(newEmail); // Update email

customer.setShippingInfo(newShippingInfo); // Update shipping info

em.getTransaction().commit(); // Commit changes

System.out.println("Customer updated successfully: " + customer);

} else {

System.out.println("Customer with name " + customerName + " not found. Update failed.");

}

}

// DELETE: Remove a Customer

public void deleteCustomer(String customerName) {

Customer customer = em.find(Customer.class, customerName); // Find customer

if (customer != null) {

em.getTransaction().begin(); // Start transaction

em.remove(customer); // Remove customer

em.getTransaction().commit(); // Commit transaction

System.out.println("Customer deleted successfully with name: " + customerName);

} else {

System.out.println("Customer with name " + customerName + " not found. Deletion failed.");

}

}

// CLOSE: Clean up resources

public void close() {

em.close();

emf.close();

}

}

Order . CRUD

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

public class OrderCRUD {

private EntityManagerFactory emf = Persistence.createEntityManagerFactory("my-persistence-unit");

private EntityManager em = emf.createEntityManager();

// CREATE: Add a new Order

public void addOrder(Order order) {

em.getTransaction().begin(); // Start transaction

em.persist(order); // Add the Order object to the database

em.getTransaction().commit(); // Commit the transaction

System.out.println("Order added successfully: " + order);

}

// READ: Find an Order by ID

public Order getOrder(int orderId) {

Order order = em.find(Order.class, orderId); // Retrieve order

if (order != null) {

System.out.println("Order found: " + order);

} else {

System.out.println("Order with ID " + orderId + " not found.");

}

return order;

}

// UPDATE: Update Order details

public void updateOrder(int orderId, String newStatus, String newDateShipped) {

Order order = em.find(Order.class, orderId); // Find order

if (order != null) {

em.getTransaction().begin(); // Start transaction

order.setStatus(newStatus); // Update order status

order.setDateShipped(newDateShipped); // Update shipping date

em.getTransaction().commit(); // Commit changes

System.out.println("Order updated successfully: " + order);

} else {

System.out.println("Order with ID " + orderId + " not found. Update failed.");

}

}

// DELETE: Remove an Order

public void deleteOrder(int orderId) {

Order order = em.find(Order.class, orderId); // Find order

if (order != null) {

em.getTransaction().begin(); // Start transaction

em.remove(order); // Remove order

em.getTransaction().commit(); // Commit transaction

System.out.println("Order deleted successfully with ID: " + orderId);

} else {

System.out.println("Order with ID " + orderId + " not found. Deletion failed.");

}

}

// CLOSE: Clean up resources

public void close() {

em.close();

emf.close();

}

}

Order Details. CRUD

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

public class ShippingInfoCRUD {

private EntityManagerFactory emf = Persistence.createEntityManagerFactory("my-persistence-unit");

private EntityManager em = emf.createEntityManager();

// CREATE: Add a new Shipping Info

public void addShippingInfo(ShippingInfo shippingInfo) {

em.getTransaction().begin(); // Start transaction

em.persist(shippingInfo); // Save the ShippingInfo object

em.getTransaction().commit(); // Commit transaction

System.out.println("Shipping Info added successfully: " + shippingInfo);

}

// READ: Find Shipping Info by ID

public ShippingInfo getShippingInfo(int shippingId) {

ShippingInfo shippingInfo = em.find(ShippingInfo.class, shippingId); // Retrieve shipping info

if (shippingInfo != null) {

System.out.println("Shipping Info found: " + shippingInfo);

} else {

System.out.println("Shipping Info with ID " + shippingId + " not found.");

}

return shippingInfo;

}

// UPDATE: Update Shipping Info details

public void updateShippingInfo(int shippingId, String newType, int newCost, int newRegionId) {

ShippingInfo shippingInfo = em.find(ShippingInfo.class, shippingId);

if (shippingInfo != null) {

em.getTransaction().begin(); // Start transaction

shippingInfo.setShippingType(newType);

shippingInfo.setShippingCost(newCost);

shippingInfo.setShippingRegionId(newRegionId);

em.getTransaction().commit(); // Commit changes

System.out.println("Shipping Info updated successfully: " + shippingInfo);

} else {

System.out.println("Shipping Info with ID " + shippingId + " not found. Update failed.");

}

}

// DELETE: Remove Shipping Info

public void deleteShippingInfo(int shippingId) {

ShippingInfo shippingInfo = em.find(ShippingInfo.class, shippingId);

if (shippingInfo != null) {

em.getTransaction().begin(); // Start transaction

em.remove(shippingInfo); // Remove shipping info

em.getTransaction().commit(); // Commit transaction

System.out.println("Shipping Info deleted successfully with ID: " + shippingId);

} else {

System.out.println("Shipping Info with ID " + shippingId + " not found. Deletion failed.");

}

}

// CLOSE: Release resources

public void close() {

em.close();

emf.close();

}

}

Shopping Cart . CRUD

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

public class ShoppingCartCRUD {

private EntityManagerFactory emf = Persistence.createEntityManagerFactory("my-persistence-unit");

private EntityManager em = emf.createEntityManager();

// CREATE: Add a new ShoppingCart item

public void addCartItem(ShoppingCart cartItem) {

em.getTransaction().begin(); // Start transaction

em.persist(cartItem); // Save the ShoppingCart object

em.getTransaction().commit(); // Commit transaction

System.out.println("Cart Item added successfully: " + cartItem);

}

// READ: Get ShoppingCart item by ID

public ShoppingCart getCartItem(int cartId) {

ShoppingCart cartItem = em.find(ShoppingCart.class, cartId); // Retrieve cart item

if (cartItem != null) {

System.out.println("Cart Item found: " + cartItem);

} else {

System.out.println("Cart Item with ID " + cartId + " not found.");

}

return cartItem;

}

// UPDATE: Update quantity for a ShoppingCart item

public void updateCartItem(int cartId, int newQuantity) {

ShoppingCart cartItem = em.find(ShoppingCart.class, cartId);

if (cartItem != null) {

em.getTransaction().begin(); // Start transaction

cartItem.setQuantity(newQuantity); // Update quantity

em.getTransaction().commit(); // Commit transaction

System.out.println("Cart Item updated successfully: " + cartItem);

} else {

System.out.println("Cart Item with ID " + cartId + " not found. Update failed.");

}

}

// DELETE: Remove a ShoppingCart item

public void deleteCartItem(int cartId) {

ShoppingCart cartItem = em.find(ShoppingCart.class, cartId);

if (cartItem != null) {

em.getTransaction().begin(); // Start transaction

em.remove(cartItem); // Remove cart item

em.getTransaction().commit(); // Commit transaction

System.out.println("Cart Item deleted successfully with ID: " + cartId);

} else {

System.out.println("Cart Item with ID " + cartId + " not found. Deletion failed.");

}

}

// CLOSE: Release resources

public void close() {

em.close();

emf.close();

}

}