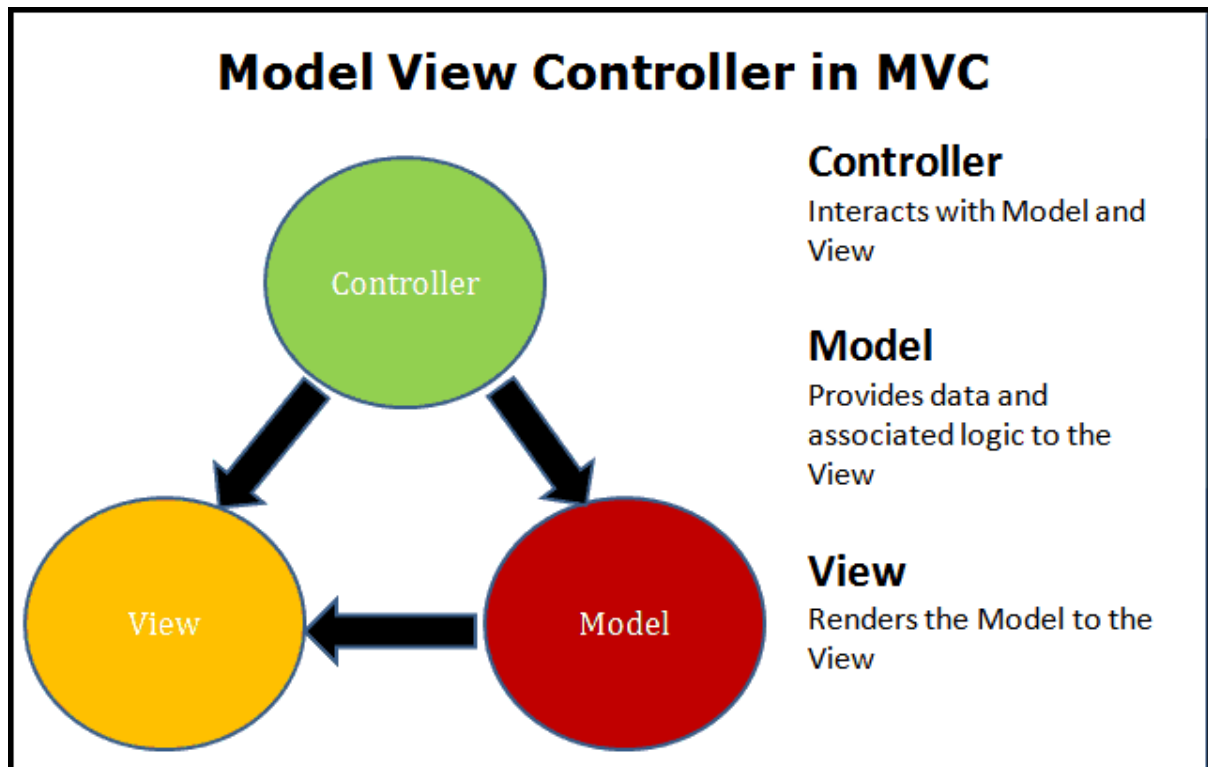


MVC CRUD demo

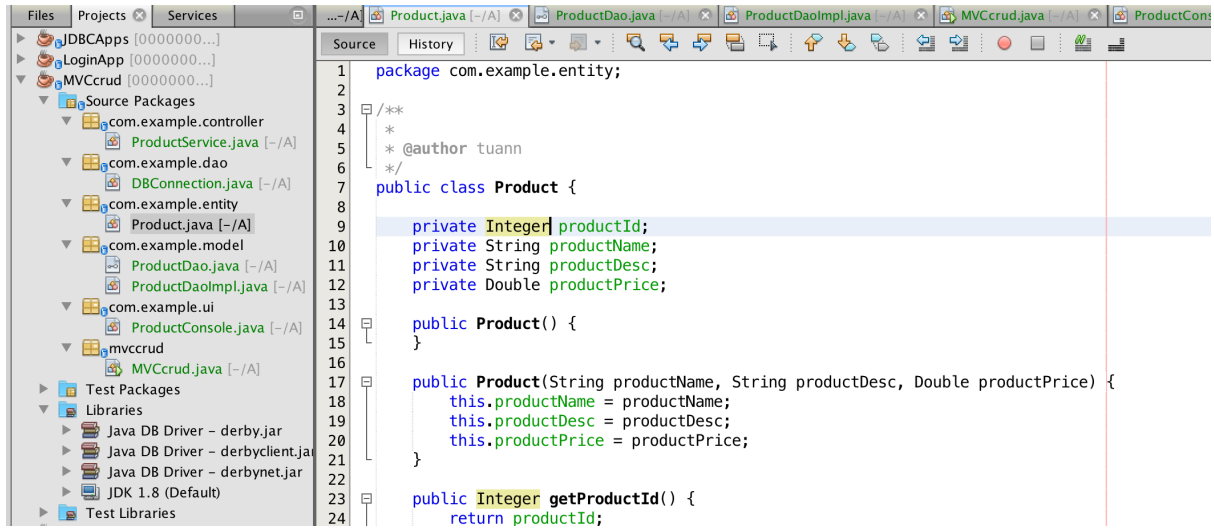


1. Tạo database với tên là MyDB với Table có tên Products cấu trúc như sau:

```
Connection: jdbc:derby://localhost:1527/myDB [sa on SA]

1 CREATE TABLE Products
2 (
3     ID int not null generated always as identity (start with 1, increment by 1) primary key,
4     ProName varchar(50),
5     ProDesc varchar(50),
6     Price double
7 )
```

2. Tạo Project có cấu trúc như sau:



3. Product entity(thực thể này sẽ mapping đến table trong CSDL):

```

1  package com.example.entity;
2
3  /**
4   *
5   * @author tuann
6   */
7  public class Product {
8
9      private Integer productId;
10     private String productName;
11     private String productDesc;
12     private Double productPrice;
13
14     public Product() {
15     }
16
17     public Product(String productName, String productDesc, Double productPrice) {
18         this.productName = productName;
19         this.productDesc = productDesc;
20         this.productPrice = productPrice;
21     }
22
23     public Integer getProductId() {
24         return productId;
25     }
26
27     public void setProductId(Integer productId) {
28         this.productId = productId;
29     }
30
31     public String getProductName() {
32         return productName;
33     }
34
35     public void setProductName(String productName) {
36         this.productName = productName;
37     }
38

```

4. DBConnection class(lớp kết nối đến CSDL):

```

1 package com.example.dao;
2
3 /**
4  *
5  * @author tuann
6  */
7 import java.sql.Connection;
8 import java.sql.DriverManager;
9 import java.sql.SQLException;
10 import java.util.logging.Level;
11 import java.util.logging.Logger;
12
13 public class DBConnection {
14
15     public static Connection createConnection() {
16         String dbUrl = "jdbc:derby://localhost:1527/myDB";
17         Connection conn = null;
18         try {
19             conn = DriverManager.getConnection(dbUrl, "sa", "sa");
20         } catch (SQLException ex) {
21             Logger.getLogger(DBConnection.class.getName()).log(Level.SEVERE, null, ex);
22         }
23         return conn;
24     }
25 }
26
27

```

5. ProductDao interface (interface này mô tả các phương thức sẽ triển khai trong lớp Biz)

```

1 package com.example.model;
2
3 /**
4  *
5  * @author tuann
6  */
7 import com.example.entity.Product;
8 import java.util.ArrayList;
9
10 public interface ProductDao {
11
12     public void createProduct(Product product);
13
14     public Product getProductById(int productId);
15
16     public ArrayList<Product> getAllProducts();
17
18     public void updateProduct(Product product);
19
20     public boolean deleteProduct(int productId);
21
22 }
23

```

6. ProductDaoImpl class (implements các phương thức mô tả trong interface)

```

1 package com.example.model;
2
3 /**
4  *
5  * @author tuann
6  */
7 import com.example.dao.DBConnection;
8 import com.example.entity.Product;
9 import java.sql.Connection;
10 import java.sql.PreparedStatement;
11 import java.sql.ResultSet;
12 import java.sql.SQLException;
13 import java.sql.Statement;
14 import java.util.ArrayList;
15 import java.util.logging.Level;
16 import java.util.logging.Logger;
17
18 public class ProductDaoImpl implements ProductDao {
19
20     private final Connection conn = DBConnection.createConnection();
21     private final String SQL_CREATE_PRODUCT = "INSERT INTO products (productName, productDesc, productPrice) VALUES (?, ?, ?)";
22     private final String SQL_GET_PRODUCT_BY_ID = "SELECT * FROM products WHERE Id=?";
23     private final String SQL_GET_ALL_PRODUCTS = "SELECT * FROM products";
24     private final String SQL_UPDATE_PRODUCT = "UPDATE products SET productName=?, productDesc=?, productPrice=? WHERE Id=?";
25     private final String SQL_DELETE_PRODUCT = "DELETE FROM products WHERE ID=?";
26
27     @Override
28     public void createProduct(Product product) {
29         try (PreparedStatement pstmt = conn.prepareStatement(SQL_CREATE_PRODUCT, Statement.RETURN_GENERATED_KEYS)) {
30             pstmt.setString(1, product.getProductName());
31             pstmt.setString(2, product.getProductDesc());
32             pstmt.setDouble(3, product.getProductPrice());
33             pstmt.executeUpdate();
34             try (ResultSet generatedKeys = pstmt.getGeneratedKeys()) {
35                 if (generatedKeys.next()) {
36                     product.setProductId(generatedKeys.getInt(1));
37                 }
38             }
39         } catch (SQLException ex) {
40             Logger.getLogger(ProductDaoImpl.class.getName()).log(Level.SEVERE, null, ex);
41         }
42     }
43
44     @Override
45     public Product getProductById(int productId) {
46         Product product = new Product();
47         try (PreparedStatement pstmt = conn.prepareStatement(SQL_GET_PRODUCT_BY_ID)) {
48             pstmt.setInt(1, productId);
49             try (ResultSet rs = pstmt.executeQuery()) {
50                 while (rs.next()) {
51                     product.setProductId(rs.getInt(1));
52                     product.setProductName(rs.getString(2));
53                     product.setProductDesc(rs.getString(3));
54                     product.setProductPrice(rs.getDouble(4));
55                 }
56             }
57         } catch (SQLException ex) {
58             Logger.getLogger(ProductDaoImpl.class.getName()).log(Level.SEVERE, null, ex);
59         }
60         return product;
61     }
62
63 }

```

```

64  @Override
65  public ArrayList<Product> getAllProducts() {
66      ArrayList<Product> allProducts = new ArrayList();
67      try (PreparedStatement pstmt = conn.prepareStatement(SQL_GET_ALL_PRODUCTS);
68          ResultSet rs = pstmt.executeQuery()) {
69          while (rs.next()) {
70              Product product = new Product();
71              product.setProductId(rs.getInt(1));
72              product.setProductName(rs.getString(2));
73              product.setProductDesc(rs.getString(3));
74              product.setProductPrice(rs.getDouble(4));
75              allProducts.add(product);
76          }
77      } catch (SQLException ex) {
78          Logger.getLogger(ProductDaoImpl.class.getName()).log(Level.SEVERE, null, ex);
79      }
80      return allProducts;
81  }
82
83  @Override
84  public void updateProduct(Product product) {
85      try (PreparedStatement pstmt = conn.prepareStatement(SQL_UPDATE_PRODUCT)) {
86          pstmt.setString(1, product.getProductName());
87          pstmt.setString(2, product.getProductDesc());
88          pstmt.setDouble(3, product.getProductPrice());
89          pstmt.setInt(4, product.getProductId());
90          pstmt.executeUpdate();
91      } catch (SQLException ex) {
92          Logger.getLogger(ProductDaoImpl.class.getName()).log(Level.SEVERE, null, ex);
93      }
94  }
95
96  @Override
97  public boolean deleteProduct(int productId) {
98      try (PreparedStatement pstmt = conn.prepareStatement(SQL_DELETE_PRODUCT)) {
99          pstmt.setInt(1, productId);
100         pstmt.executeUpdate();
101         return true;
102     } catch (SQLException ex) {
103         Logger.getLogger(ProductDaoImpl.class.getName()).log(Level.SEVERE, null, ex);
104     }
105     return false;
106 }
107
108 }
109

```

7. ProductConsole class(Lớp “view” đóng vai trò tương tác với người dùng)

```

1 package com.example.ui;
2
3 /**
4  *
5  * @author tuann
6  */
7 import com.example.entity.Product;
8 import java.io.IOException;
9 import java.util.Scanner;
10 import com.example.controller.ProductService;
11 import java.util.ArrayList;
12
13 /**
14  *
15  * @author tuann
16  */
17 public class ProductConsole {
18
19     private final Scanner sc;
20     ProductService productService = new ProductService();
21
22     public ProductConsole() {
23         this.sc = new Scanner(System.in);
24     }
25
26     private int menu() {
27         System.out.println("----PRODUCT MENU----");
28         System.out.println("1. Add product");
29         System.out.println("2. Show all product");
30         System.out.println("3. Remove product");
31         System.out.println("0. Exit");
32         int choice = readInt(0, 3);
33         return choice;
34     }
35
36     public void start() {
37         int choice;
38         do {
39             choice = menu();
40             switch (choice) {
41                 case 1:
42                     addProduct();
43                     break;
44                 case 2:
45                     showAllProduct();
46                     break;
47                 case 3:
48                     removeProduct();
49                     break;
50                 case 0:
51                     break;
52             }
53         } while (choice != 0);
54     }
55
56     private void addProduct() {
57         Product product = new Product();
58         product.setName(readString());
59         product.setPrice(readInt(0, 1000000));
60         productService.addProduct(product);
61     }
62
63     private void showAllProduct() {
64         List<Product> products = productService.getAllProduct();
65         for (Product product : products) {
66             System.out.println(product);
67         }
68     }
69
70     private void removeProduct() {
71         int id = readInt(0, 1000000);
72         productService.removeProduct(id);
73     }
74
75     private String readString() {
76         return sc.nextLine();
77     }
78
79     private int readInt(int min, int max) {
80         int choice;
81         do {
82             choice = sc.nextInt();
83         } while (choice < min || choice > max);
84         return choice;
85     }
86 }

```

```

37 public void start() {
38     while (true) {
39         int choice = menu();
40         switch (choice) {
41             case 0:
42                 System.exit(0);
43                 break;
44             case 1:
45                 addProduct();
46                 break;
47             case 2:
48                 showAll();
49                 break;
50             case 3:
51                 removeProduct();
52                 break;
53             default:
54                 throw new AssertionError();
55         }
56     }
57 }

58 private int readInt(int min, int max) {
59     int choice;
60     while (true) {
61         try {
62             choice = Integer.parseInt(sc.nextLine());
63             if (choice >= min && choice <= max) {
64                 break;
65             }
66         } catch (NumberFormatException e) {
67         }
68     }
69     return choice;
70 }
71

```

```

72 private double readDouble(int min, double max) {
73     double price;
74     while (true) {
75         try {
76             price = Double.parseDouble(sc.nextLine());
77             if (price >= min && price <= max) {
78                 break;
79             }
80         } catch (NumberFormatException e) {
81             System.out.println("Invalid value. Try to enter a float value");
82         }
83     }
84     return price;
85 }

86 private void addProduct() {
87     System.out.println("Enter product name: ");
88     String name = sc.nextLine();
89     System.out.println("Enter product desc: ");
90     String desc = sc.nextLine();
91     System.out.println("Enter product price: ");
92     double price = readDouble(0, Double.MAX_VALUE);
93     Product product = new Product(name, desc, price);
94     productService.createProduct(product);
95 }

96 private void showAll() {
97     System.out.println("--All product--");
98     System.out.println("ID\tName\tDesc\tPrice");
99     ArrayList<Product> allProducts = productService.getAllProducts();
100     allProducts.forEach((product) -> {
101         System.out.println(product.getProductId() + "\t" + product.getProductName() + "\t"
102             + product.getProductDesc() + "\t" + product.getProductPrice());
103     });
104 }
105

```



```

110 private void removeProduct() {
111     System.out.println("Enter ID of product");
112     int id = readInt(0, Integer.MAX_VALUE);
113     boolean result = productService.deleteProduct(id);
114     if (result) {
115         System.out.println("Product was removed");
116     } else {
117         System.out.println("Product not found");
118     }
119 }
120 }
121

```

8. MVCcrud -> run application

```

1 package mvccrud;
2
3 import java.io.IOException;
4 import com.example.ui.ProductConsole;
5
6 /**
7  *
8  * @author tuann
9  */
10 public class MVCcrud {
11
12     /**
13      * @param args the command line arguments
14      * @throws java.io.IOException
15      */
16     public static void main(String[] args) throws IOException {
17         ProductConsole pc = new ProductConsole();
18         pc.start();
19     }
20
21 }
22

```