Building a 3-Layered Desktop Application in Java

Swing GUI, MySQL Database, and DTO Integration

Affan Rauf

National University of Computer and Emerging Sciences

October 13, 2024

Overview of Three-Layered Architecture

- This application follows a three-layered architecture:
  1. **Presentation Layer**: Swing-based user interface.
  2. **Business Logic Layer**: Processes input, interacts with the data layer using DTOs.
  3. **Data Access Layer**: Communicates with MySQL database.
- We will build an application to search for books by their title and display results from a MySQL database.

Prerequisites

- Java Development Kit (JDK)
- MySQL Server
- MySQL JDBC Driver (`mysql-connector-java`)

Step 1: MySQL Database Setup

- Create a MySQL database:

```
CREATE DATABASE bookdb;

USE bookdb;

CREATE TABLE books (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255),
    year INT
);

INSERT INTO books (title, author, year) VALUES
('Laash ka Qehqaha', 'Ibn e Safi', 1954);
('Jiraal ka Mansubah', 'Ishtiaq Ahmad', 1985);
('Ameer Hamza Koh-e-Qaf Mein', 'Maqbool Jahangir', 1951),
('Umro ki Ghaddari', 'Akhtar Rizvi', 1960),
('Chalosak Malosak Sabz Sitaray Mein', 'Mazhar Kaleem', 1980),
('Hajjaam aur Qazzaaq', 'Iqbal Kardar', 1949);
```

Step 2: Project Structure

- Project folders:
- `src/model` (Data Access Layer)
- `src/logic` (Business Logic Layer)
- `src/view` (Presentation Layer)
- `src/dto` (Data Transfer Objects)

Step 3: Data Access Layer

- The Data Access Layer handles database interaction via JDBC.
- Set up database connection:

```
// DatabaseConfig.java
package model;
import java.sql.*;

public class DatabaseConfig {
    private static final String URL = "jdbc:mysql://localhost:3306/bookdb";
    private static final String USER = "root";
    private static final String PASSWORD = "password";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

Step 3: Data Access Layer - BookDAO

- Create a `BookDAO.java` class to search for books by title:

```
// BookDAO.java
package model;
import java.sql.*;
import java.util.*;
import dto.BookDTO;

public class BookDAO {
    public List<BookDTO> searchBooksByTitle(String title) {
        List<BookDTO> books = new ArrayList<>();
        String query = "SELECT * FROM books WHERE title LIKE ?";

        try (Connection conn = DatabaseConfig.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query)) {

            stmt.setString(1, "%" + title + "%");
            ResultSet rs = stmt.executeQuery();

            while (rs.next()) {
                books.add(new BookDTO(rs.getString("title"),
                                      rs.getString("author"),
                                      rs.getInt("year")));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return books;
    }
}
```

Step 4: Business Logic Layer

- The Business Logic Layer handles the interaction between the Presentation and Data Access layers.

```
// BookBO.java
package logic;
import model.BookDAO;
import dto.BookDTO;
import java.util.*;

public class BookBO {
    private BookDAO bookDAO;

    public BookBO() {
        this.bookDAO = new BookDAO();
    }

    public List<BookDTO> searchBooks(String title) {
        return bookDAO.searchBooksByTitle(title);
    }
}
```

Step 5: Data Transfer Object (DTO)

- The DTO pattern decouples data between layers.
- Define a `BookDTO.java` to represent book data during transfer:

```
// BookDTO.java
package dto;

public class BookDTO {
    private String title;
    private String author;
    private int year;

    public BookDTO(String title, String author, int year) {
        this.title = title;
        this.author = author;
        this.year = year;
    }

    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }
    public String getAuthor() { return author; }
    public void setAuthor(String author) { this.author = author; }
    public int getYear() { return year; }
    public void setYear(int year) { this.year = year; }

    @Override
    public String toString() {
        return title + " by " + author + " (" + year + ")";
    }
}
```

Step 6: Presentation Layer

- The Presentation Layer uses Java Swing to create a user interface.

```
// BookSearchApp.java
package view;
import logic.BookBO;
import dto.BookDTO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.List;

public class BookSearchApp extends JFrame {
    private JTextField searchField;
    private JButton searchButton;
    private JTextArea resultArea;
    private BookBO bookBO;

    public BookSearchApp() {
        bookBO = new BookBO();
        setTitle("Book Search Application");
        setSize(500, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        setLayout(new BorderLayout());
        JPanel topPanel = new JPanel();
        topPanel.add(new JLabel("Search for a book: "));
        searchField = new JTextField(20);
        topPanel.add(searchField);
        searchButton = new JButton("Search");
        topPanel.add(searchButton);
        add(topPanel, BorderLayout.NORTH);
```

Step 6: Presentation Layer

```
        resultArea = new JTextArea();
        resultArea.setEditable(false);
        add(new JScrollPane(resultArea), BorderLayout.CENTER);

        searchButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                searchBooks();
            }
        });
    }

    private void searchBooks() {
        String title = searchField.getText();
        List<BookDTO> books = bookBO.searchBooks(title);
        resultArea.setText("");
        if (books.isEmpty()) {
            resultArea.append("No books found.");
        } else {
            for (BookDTO book : books) {
                resultArea.append(book.toString() + "\n");
            }
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new BookSearchApp().setVisible(true);
            }
        });
    }
}
```

Running the Application

- Compile all Java files and ensure the MySQL JDBC driver is in your classpath.
- Run `BookSearchApp.java`.
- Enter a book title in the search field and click "Search" to see results.

Summary

- We built a 3-layered application using Java Swing and MySQL.
- Introduced a Data Transfer Object (DTO) to decouple data between layers.
- Explored Java code implementation for each layer.

Business Objects (BOs)

- **Definition:** Represent real-world entities or concepts within a business domain. They encapsulate both data and behavior (business logic) related to those entities.
- **Responsibilities:**
  - Hold and manage data relevant to the business domain.
  - Implement business rules and validation logic.
  - Interact with Data Access Layer (usually via DAOs).
- **Example:** A ‘Product‘ BO with attributes like ‘id‘, ‘name‘, ‘price‘ and methods like ‘calculateDiscount()‘.

Data Access Objects (DAOs)

- **Definition:** Interact with data sources on behalf of business objects. They abstract away the underlying data storage details.
- **Responsibilities:**
  - Perform CRUD (Create, Read, Update, Delete) operations.
  - Handle database connections and queries.
  - Translate Business Objects into a suitable format for storage.
- **Example:** A ‘ProductDAO‘ with methods like ‘findById(int id)‘, ‘save(Product product)‘, and ‘delete(Product product)‘.

Data Transfer Objects (DTOs)

- **Definition:** Simple objects used to transfer data between different layers of an application. They often contain only data members, without any methods.
- **Responsibilities:**
  - Act as a container for data transferred within the system.
  - Minimize the number of method calls by grouping related data.
- **Example:** A ‘ProductDTO‘ with fields like ‘id‘, ‘name‘, and ‘price‘, focusing solely on data representation.

Usage of DTOs

- **Between Presentation Layer and Business Layer:**
  - Encapsulate data being transmitted, standardizing the format.
  - Carry input data (form submissions) to the business layer and output data back to the presentation layer.
- **Between Business Layer and Data Access Layer:**
  - Represent data structures needed for persistence.
  - Optimize data retrieval by combining multiple entities into a single DTO.

Summary

- **Business Objects (BOs):** Encapsulate business logic and represent domain entities.
- **Data Access Objects (DAOs):** Provide an interface for interacting with the underlying data sources.
- **Data Transfer Objects (DTOs):** Transfer data between layers, enhancing maintainability and separation of concerns.