MVC патерн

Model-View-Controller – це архітектурний шаблон проектування, який передбачає поділ системи на три частини: модель даних, інтерфейс та контролер, який з'єднує дві попередні частини. Даний патерн застосовується для полегшення роботи над кодом, оскільки всі три частини чітко відокремлені і можуть розроблятися окремо. Таким чином зміни в інтерфейсі користувача мінімально впливають на модель і, відповідно, навпаки, зміни в моделі, мінімально впливають на інтерфейс. В моїй системі даний патерн представлений таким чином:

Model. В даній частині у мене розміщені моделі даних (entity), взаємодія з базою даних (dao, DBManager), а також компаратори, які використовуються для порівняння моделей.

View. В даній частині знаходяться всі jsp-сторінки, які відображаються у користувача. Також, тут же знаходяться стилі до цих сторінок та дескриптор розгортання веб-додатку (web.xml).

Controller. В даній частині знаходиться головна частина, яка керує всім проектом. В моєму проекті, контролерами виступають спеціальні класи – сервлети, які приймають дані з jsp-сторінок, обробляють їх, та відсилають відповідь. Також тут розміщені спеціальні класи – фільтри, які обмежують доступ до тих чи інших сторінок.

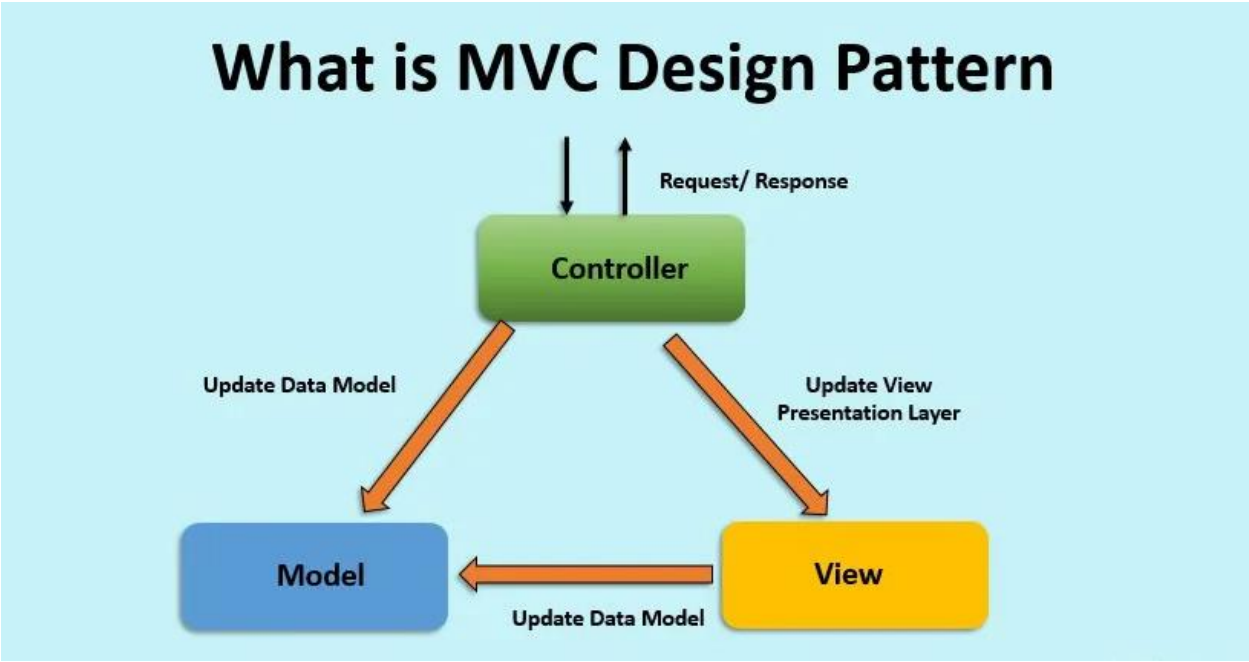


Рис. 1. MVC Pattern

Servlet Servlet – це класи-контролери, які керують проектом.

При виконанні коду цього класу створюються всі необхідні об’єкти для jsp сторінки. Після чого вони передаються на неї і відображаються у користувача. При виконанні користувачем певної дії, наприклад натиснення кнопки реєстрації, йде запит на сервер – до сервлета, де цей запит і обробляється.

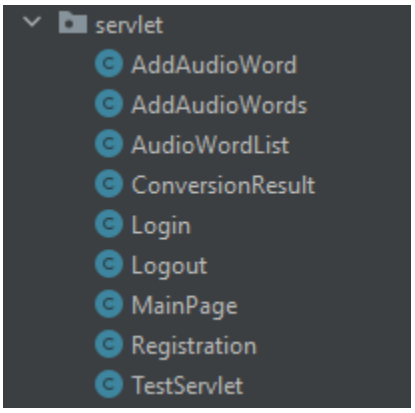
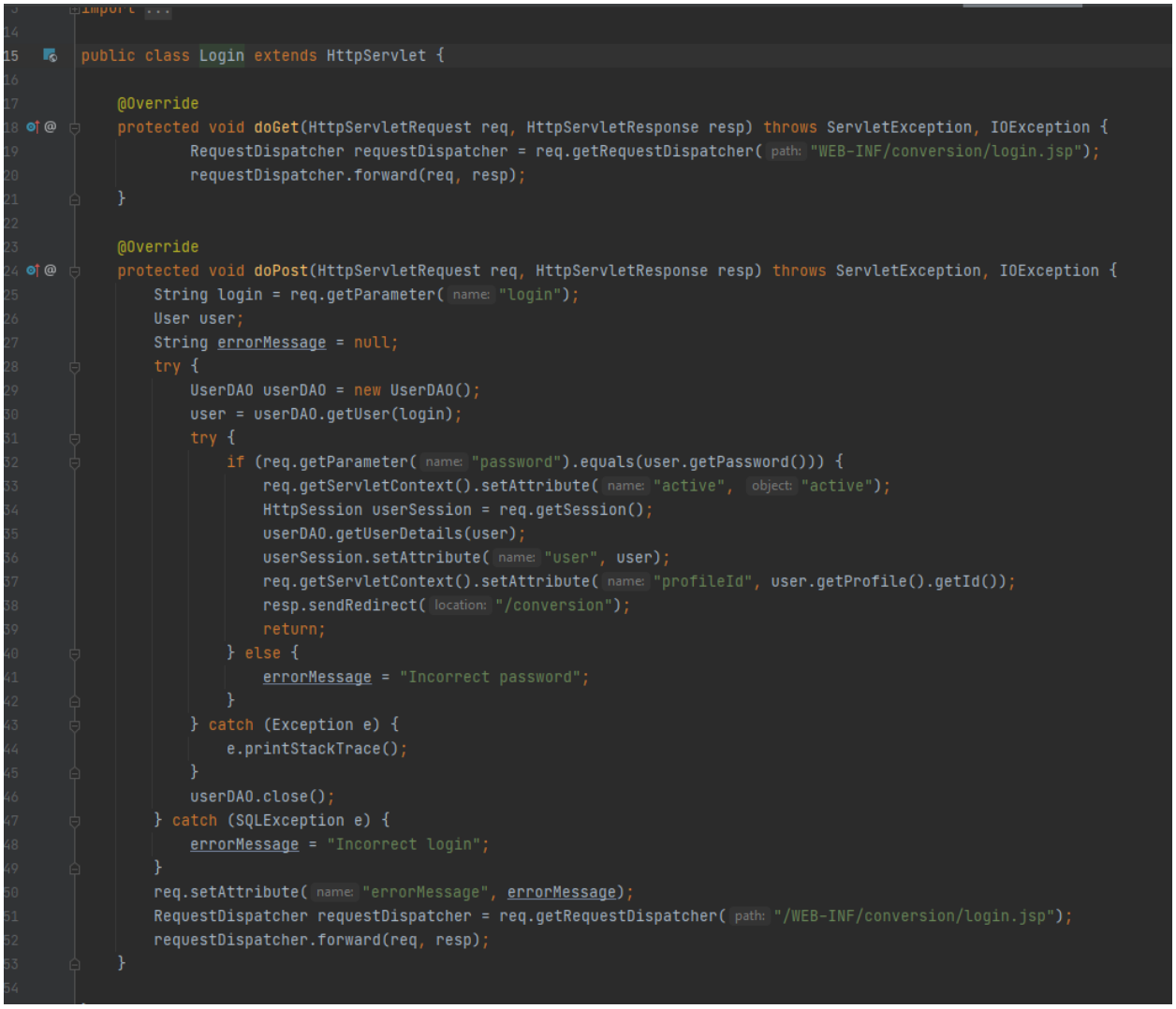


Рис. 2. Servlet класи системи

Приклад сервлету для сторіки логіну. При зверненні на сторінку сервлет обробляє запит методом doGet (рис. 3), який в даному випадку просто перенаправляє користувача на jsp сторінку з логіном.

У випадку натиснення кнопки логіну, запит обробляється іншим методом – doPost (рис. 3). Який метод буде обробляти запит – вказується в формі на jsp сторінці за допомогою атрибута method. В даному прикладі (рис. 3) метод doPost звертається до бази даних і намагається знайти там користувача з введеним логіном. Якщо ж такого користувача не знайдено, то користувачу буде надіслано повідомлення, що логін не вірний. Якщо ж такий користувач існує, то йде перевірка правильності вводу пароля. При успішному логіненню користувач потрапляє на головну сторінку для конвертації текстових файлів.



```java
import ...

public class Login extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
            RequestDispatcher requestDispatcher = req.getRequestDispatcher( path: "WEB-INF/conversion/login.jsp");
            requestDispatcher.forward(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        String login = req.getParameter( name: "login");
        User user;
        String errorMessage = null;
        try {
            UserDAO userDAO = new UserDAO();
            user = userDAO.getUser(login);
            try {
                if (req.getParameter( name: "password").equals(user.getPassword())) {
                    req.getServletContext().setAttribute( name: "active",  object: "active");
                    HttpSession userSession = req.getSession();
                    userDAO.getUserDetails(user);
                    userSession.setAttribute( name: "user", user);
                    req.getServletContext().setAttribute( name: "profileId", user.getProfile().getId());
                    resp.sendRedirect( location: "/conversion");
                    return;
                } else {
                    errorMessage = "Incorrect password";
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
            userDAO.close();
        } catch (SQLException e) {
            errorMessage = "Incorrect login";
        }
        req.setAttribute( name: "errorMessage", errorMessage);
        RequestDispatcher requestDispatcher = req.getRequestDispatcher( path: "/WEB-INF/conversion/login.jsp");
        requestDispatcher.forward(req, resp);
    }
```

Рис. 3. Login Servlet

Web module

В даній частині знаходяться всі jsp-сторінки, які відображаються користувачам за допомогою servlet. Дескриптор розгортання веб-додатку (web.xml). **SP** (**Java Server Pages**) — технологія, що дозволяє веб-розробникам динамічно генерувати HTML, XML та інші веб-сторінки. У цьому допомагає бібліотека jstl, з основними тегами <c:if>(для створення умов щоб згенерувати ту чи іншу частину сторінки) та <c:forEach>(для роботи з ітеруючими об’єктами)

Нижче показаний приклад частини jsp сторінки яка відображає аудіо слова

```
            </li>
            <li>
                <a href="/conversionResult">Conversion Result</a>
            </li>
            <c:if test="${sessionScope.user.profile.developerName == 'Administrator'}">
                <li>
                    <a href="/audioWordList?name=standard">Standard Dictionary</a>
                </li>
            </c:if>
        </ul>
    </nav>
</div>
</div>

<div class="container">
    <div class="container text-center">
        <div class="basic-login">
            <div class="row">
                <a class="btn pull-right" href="/addAudioWord?name=${param.get('name')}">Add Audio Word</a>
                <c:forEach items="${wordList}" var="word">
                    <div class="col-lg-4 col-md-6 col-sm-12">
                        <div class="blog-post">
                            <form style="..." method="post"  >
                                <label for="wordName" class="form-label">Word "${word.wordString}"</label>
                                <audio id="wordName" controls name="media">
                                    <source src="data:" type="audio/mpeg">
                                </audio>
                                <input type="hidden" name="wordId" class="btn btn-sm btn-dark" value="${word.id}">
                                <input type="hidden" name="wordName" class="btn btn-sm btn-dark" value="${word.wordString}">
                                <div>
                                    <button type="submit" name="action" class="btn pull-right" value="Upload New Audio File" style="...">Upload New Audio File</
                                    <button type="submit" name="action" class="btn pull-right" value="Delete" style="...">Delete</button>
                                </div>
                            </form>
                        </div>
                    </div>
                </c:forEach>
            </div>
        </div>
    </div>
```

Рис. 4. Частина jsp сторінки

Перелік jsp сторінок використаних для моєї системи

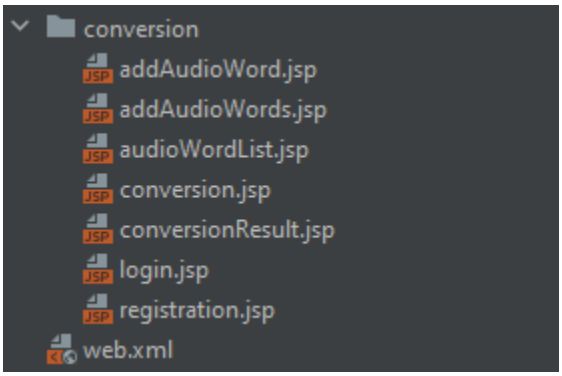


Рис. 5. jsp сторінки

Web.xml

Даний конфігураційний файл являє собою дескриптор розгортання веб-додатку. Форматом цього файлу є xml формату, який у більшості випадках використовується для конфігурації систем та додатків. Головною превагою таких файлів є те, що розробник сам може створити розмітку, яка потрібна йому, таким чином він не обмежений ситаксичними правилами мови програмування.

Для web.xml файлу описуються та мапуються (задаються шляхи) всі сервлети системи, фільтри , а також, задаються загальні параметри.

Web.xml створюється автоматично при додаванні до проекту вебмодуля

```
<servlet>
    <servlet-name>conversion</servlet-name>
    <servlet-class>controller.servlet.MainPage</servlet-class>
    <async-supported>true</async-supported>
    <multipart-config>
        <location>/</location>
        <max-file-size>20848820</max-file-size>
        <max-request-size>418018841</max-request-size>
        <file-size-threshold>1048576</file-size-threshold>
    </multipart-config>
</servlet>

<servlet-mapping>
    <servlet-name>conversion</servlet-name>
    <url-pattern>/conversion</url-pattern>
</servlet-mapping>
```

Рис. 6. Мапування сервлета

```xml
<filter>
    <filter-name>administratorFilter</filter-name>
    <filter-class>controller.filter.AdministratorFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>administratorFilter</filter-name>
    <servlet-name>addAudioWord</servlet-name>
    <servlet-name>audioWordList</servlet-name>
</filter-mapping>
```

Рис. 7. Мапування фільтру

Pom.xml

Даний конфігураційний файл створюється автоматично при створенні додатку з структурою Maven. Pom.xml містить таку інформацію про проект як groupId, artifactId, version (рис. 8). Також цей файл відповідає за додавання залежностей в проект (рис. 9). При додаванні будь-якої залежності в цей файл, вона буде автоматично скачана, встановлена, налаштована та готова до використання, що є дуже зручним.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>andrii.makarchuk</groupId>
    <artifactId>Conversion</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>15</maven.compiler.source>
        <maven.compiler.target>15</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>commons-dbcp</groupId>
            <artifactId>commons-dbcp</artifactId>
            <version>1.4</version>
        </dependency>

        <dependency>
            <groupId>jstl</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.47</version>
        </dependency>

        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
        </dependency>
```

Рис. 8. pom.xml і залежності

База даних

База даних складається з 6 таблиць :

3 з них використовуються для збереження даних про користувачів(user1, user_details, profile)

user1 зберігає основну інформацію користувача, пароль і логін

user_details зберігають додаткові розширені дані користувача і

profile зберігає профайли, які використовуються для для визначення можливостей користувача в системі

audio_word зберігає аудіо слова, які формують словники для користувачі , за допомогою яких буде відбуватись конвертація.

word_end зберігає закінчення слів такі як –s, -es, -est, -ing, -er, -ed, -ings, які використовуються для пришвидшення конвертації, оскільки якщо в словнику є запис слова “word” значить слово “words” буде також розпізнане. За допомогою поля created_by можливо визначити яку словнику якого юзера воно належить, is_standard поле ігнорує created_by і додається до стандартного системного словника

conversion_record зберігає результат конвертації(destination_file_blob), і також джерело з якого була здійснена конвертація(destination_file_blob), поле created_date потрібне для перевірки коли був створений запис, і якщо пройшов певний час то здійснити його видалення оскільки не можливо вічно зберігати конвертовані файли оскільки місце рано чи пізно закінчиться.

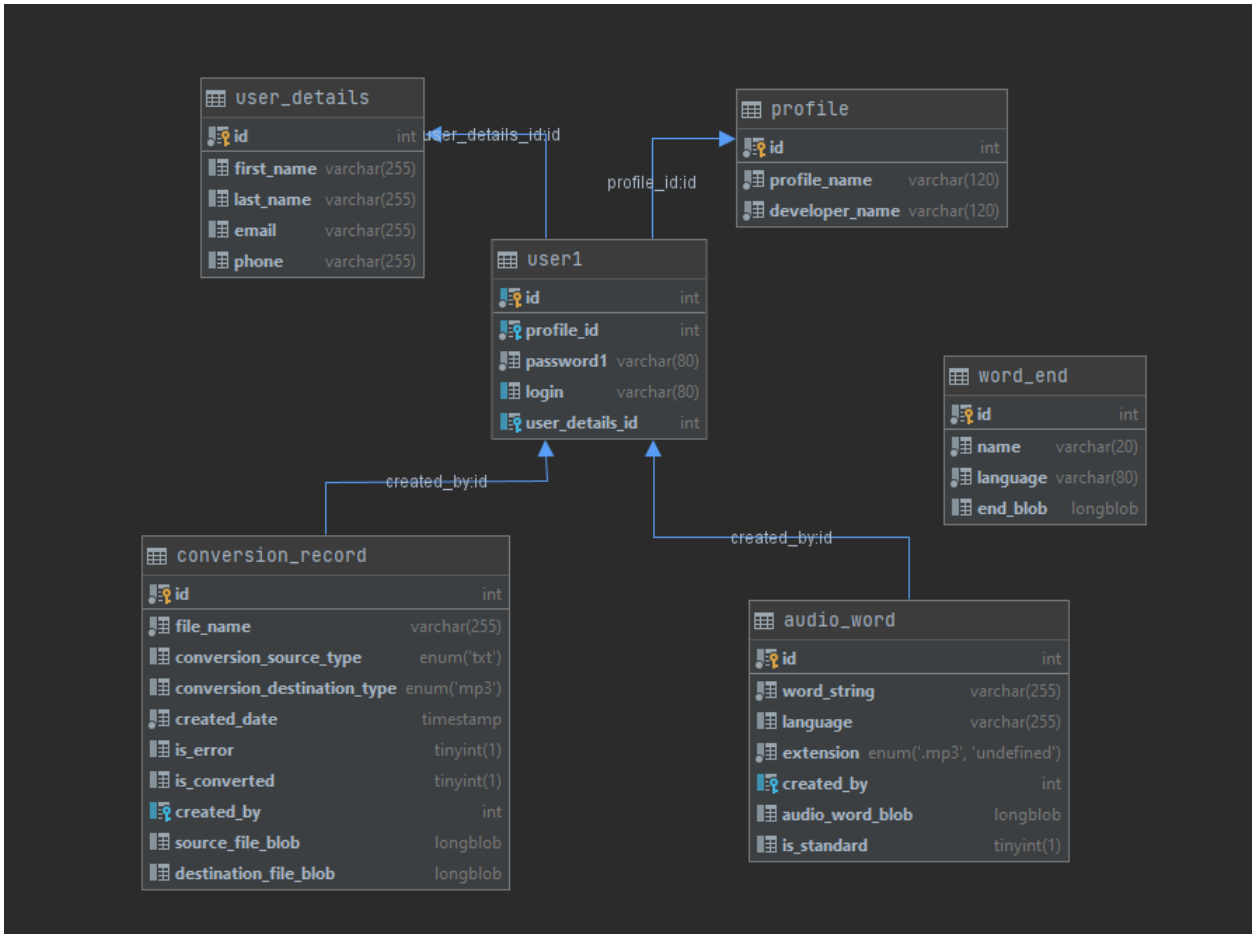


Рис. 9. Вигляд бази даних