享健你

遇見更好的自己

# 大綱

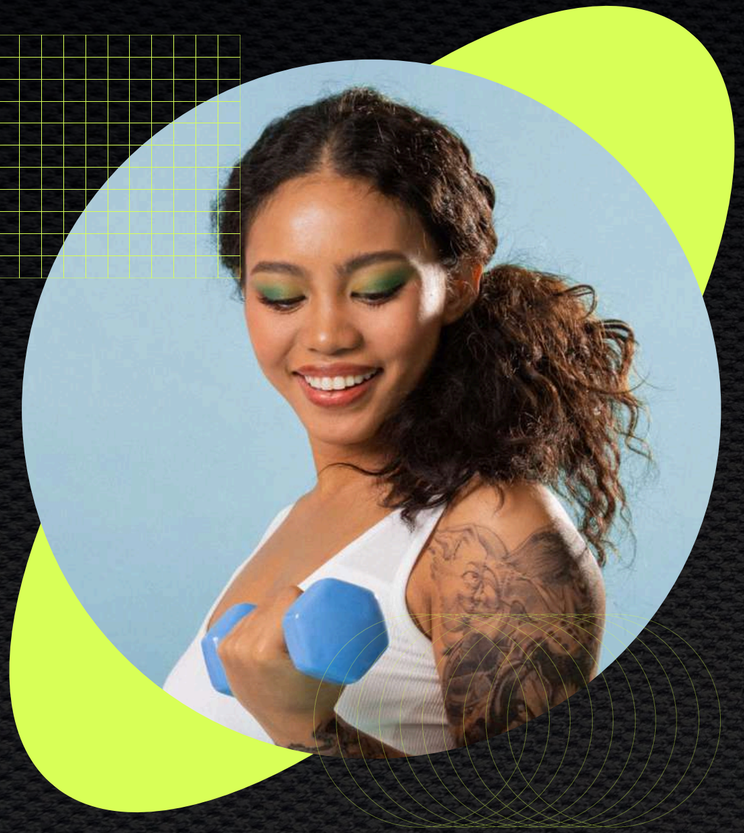
會員管理

健身成效

社群論壇

商城購物

課程管理



# 會員管理系統

提供使用者註冊、登入、修改、刪除會員資料的功能

# 大綱

開發規範與協作

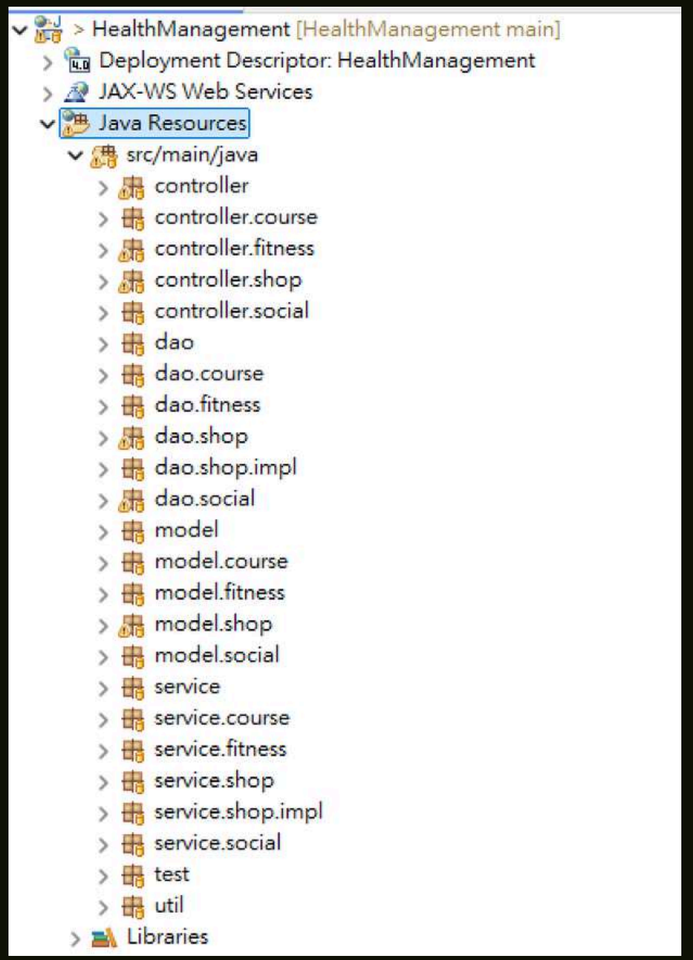
程式碼架構

系統功能與實作

# 開發規範與協作

## 後端規範

- Controller: 負責處理請求，如 LoginController
- Service: 負責業務邏輯，如 UserService
- DAO: 負責與資料庫交互，如 UserDAO
- Model: Java Bean，如 User.java
- Util: 公用工具，如 DBUtil



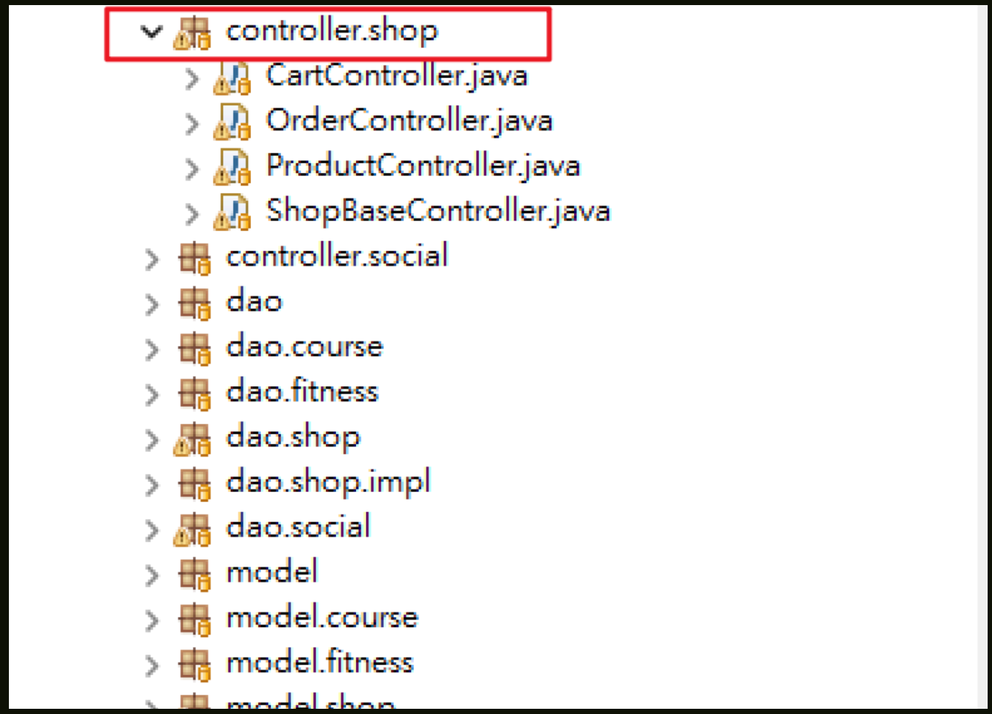
# 開發規範與協作

## ＡＰＩ規範

/api/{package}/{功能}

範例:

- 會員登入: /api/member/login
- 商城購物: /api/shop/cart/add
- 健身成效: /api/fitness/progress
- 課程管理: /api/course/enroll
- 社群論壇: /api/forum/post




```
@WebServlet("/api/shop/cart/*")
public class CartController extends ShopBaseController {
```

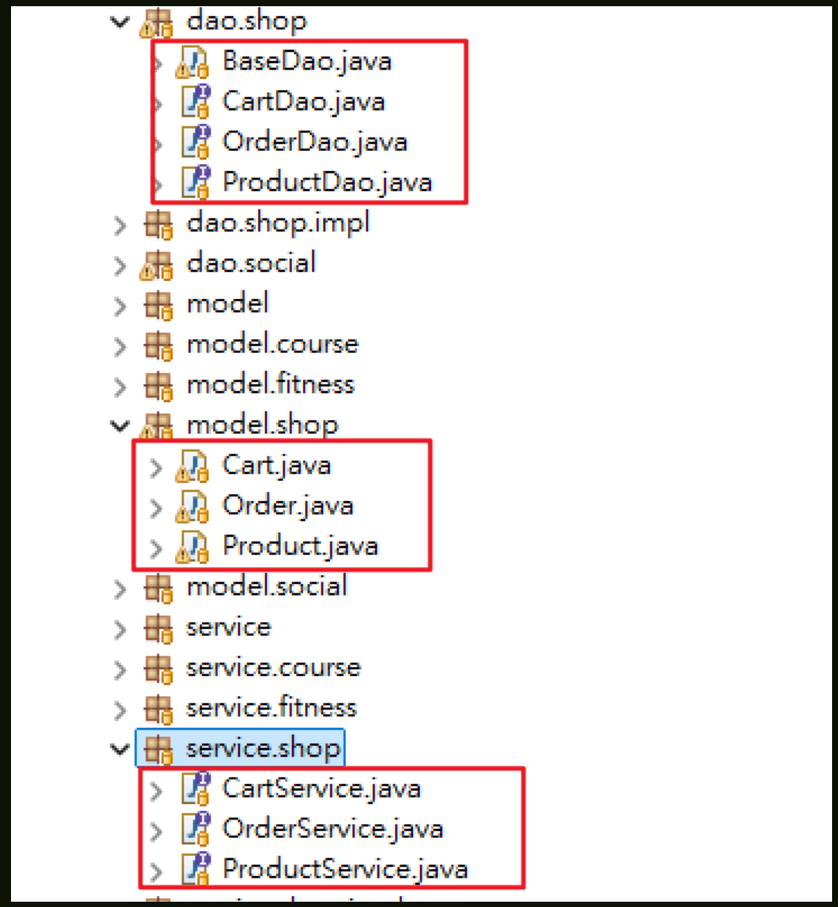
# 開發規範與協作

## 程式碼命名規範

DAO
- UserDAO.java
- OrderDAO.java
- ProgressDAO.java

Model
- User.java
- Order.java
- Progress.java

Service
- UserService.java
- OrderService.java
- ProgressService.java



# 開發規範與協作

## GIT協作

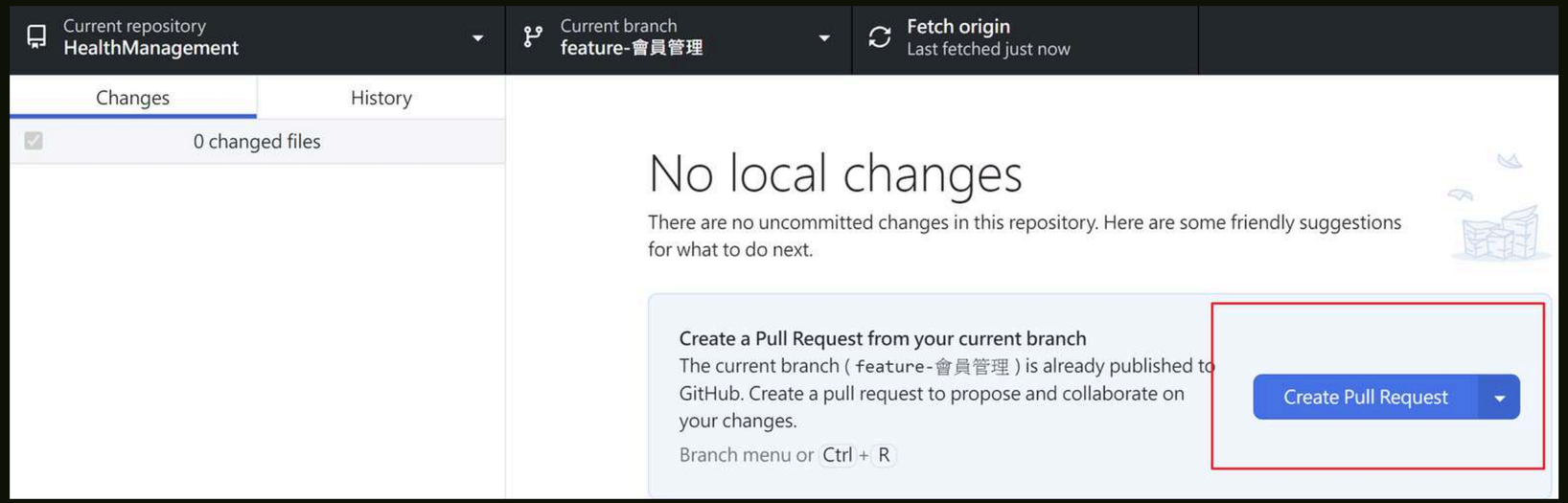
開發流程：

```
git checkout main    （確保在 main 分支）
git pull origin main    （拉取最新的 main）
git checkout -b feature    （創建新分支）

git add .
git commit -m "完成 XXX 功能"
git push origin feature-xxx    （推送到 GitHub）
```

# 開發規範與協作

## GIT協作

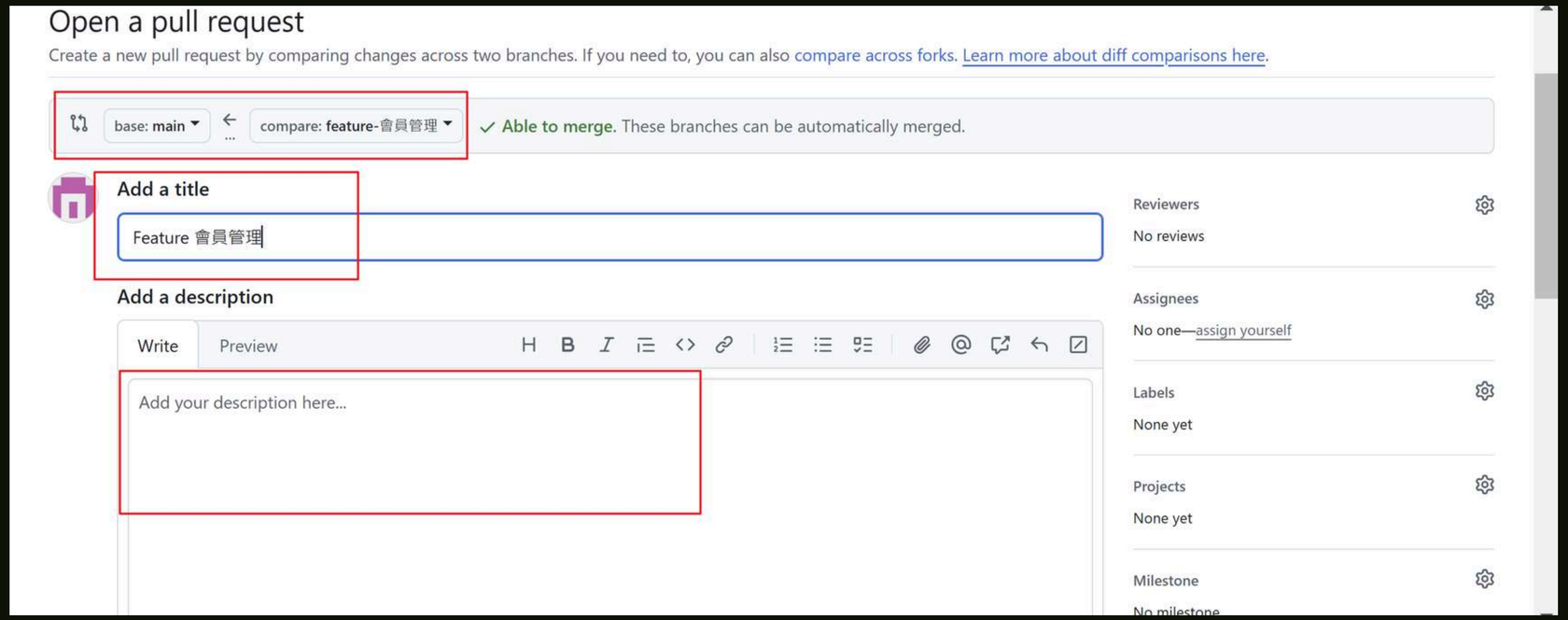
### 發送 Pull Request (PR)：



# 開發規範與協作

## GIT協作

### Pull Request (PR)：



# 開發規範與協作

GIT協作

審核 PR：

- GitHub 查看 PR
- 下載分支，本地測試
  git fetch origin
  git checkout feature-xxx
- Approve PR，Merge 到 main

# 開發規範與協作

## GIT協作

PR合併後：

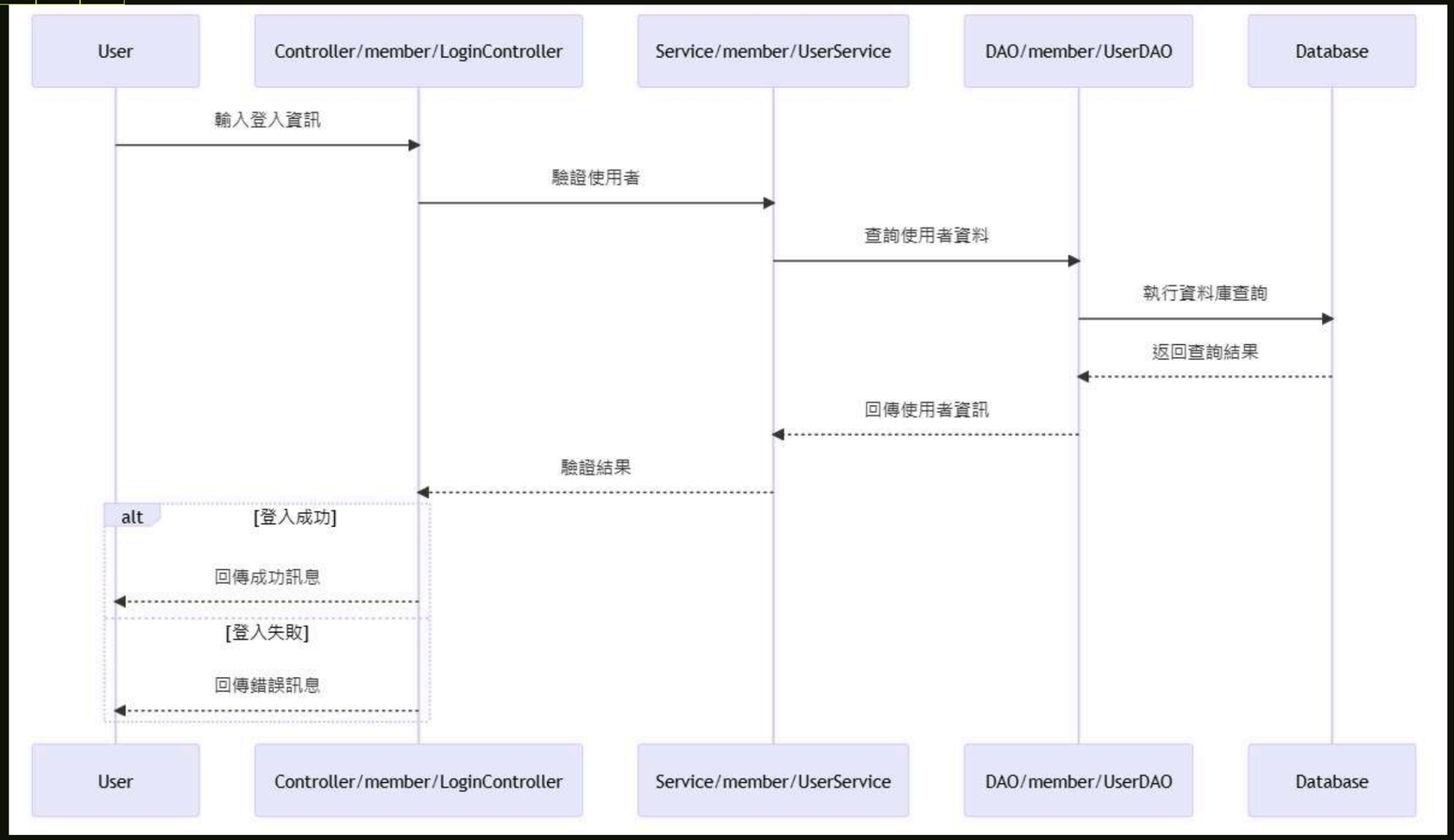
- 組員同步 main
  git checkout main → git pull origin main

- 組員繼續開發（開新分支）
  git checkout -b feature-新的功能

- 繼續舊分支開發
  git checkout feature-xxx → git pull --rebase origin main

# 程式碼架構

## MVC三層架構



# 系統功能與實作

SQL建立TABLE

```
CREATE TABLE users (
    id INT PRIMARY KEY IDENTITY(1,1),                          -- 使用者ID
    name VARCHAR(50) NOT NULL,                                 -- 使用者名稱
    email VARCHAR(100) UNIQUE NOT NULL,                        -- 郵件
    password VARCHAR(255) NOT NULL,                            -- 密碼
    gender CHAR(1) CHECK (gender IN ('M', 'F', 'O')),          -- 姓名
    bio TEXT                                                   -- 自我介紹
);
```

# 系統功能與實作

## 資料庫連線

```java
package util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBUtil {
    //指定SQL Server 的位置
    private static final String URL ="jdbc:sqlserver://localhost:1433;databaseName=Health
    private static final String USER = "Ivan";
    private static final String PASSWORD = "abcd+1234";
    //載入驅動
    static {
        try {
            //用來 註冊 SQL Server JDBC 驅動
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
    //建立一個 Connection 物件，讓其他類別可以調用，進行 SQL 查詢與操作。
    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

# DAO資料庫連線

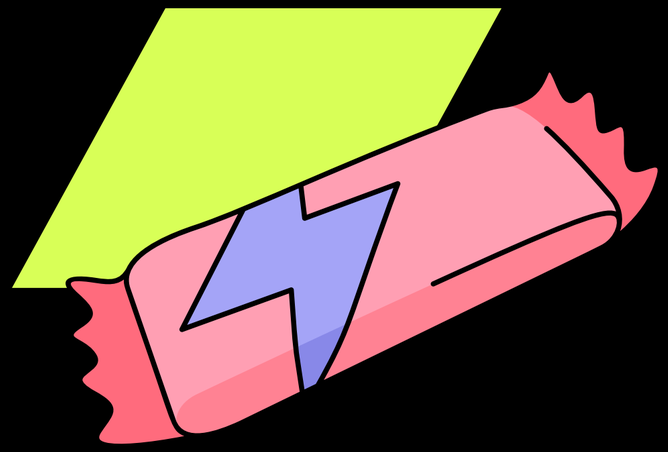
```
    }

    // 6. 刪除會員
    public void deleteUser(String email) throws SQLException {
        Connection conn = null;
        PreparedStatement stmt1 = null;
        PreparedStatement stmt2 = null;
        try {
            conn = DBUtil.getConnection();
            conn.setAutoCommit(false); //  ◇ 開啟事務，確保數據一致性

            // ◇ 先刪除 `exercise_records` 內與該會員相關的數據
            String deleteExerciseRecordsSQL = "DELETE FROM exercise_records WHERE
            stmt1 = conn.prepareStatement(deleteExerciseRecordsSQL);
            stmt1.setString(1, email);
            stmt1.executeUpdate();

            // ◇ 再刪除 `users` 表中的會員
            String deleteUserSQL = "DELETE FROM users WHERE email = ?";
            stmt2 = conn.prepareStatement(deleteUserSQL);
            stmt2.setString(1, email);
            stmt2.executeUpdate();
```

# 系統功能與實作


享健你，遇見更好的自己。
你，今天健了嗎？
首頁
會員管理
商城購物
健身成效
課程管理
社群論壇
會員登入
輸入 Email
輸入密碼
登入 註冊
© 2025 享健你. 讓運動成為習慣，遇見更好的自己。

# 線上健身商城

線上商城提高健身房營收。提供健身補給品與運動裝備的便利購買選項。會員專屬折扣與積分回饋機制。



## 健身補給品

多樣選擇，滿足您的營養需求。

## 運動裝備

最新潮流，提升您的運動表現。

## 會員專屬優惠

折扣與積分，回饋您的支持。

# 目錄 - 商城購物

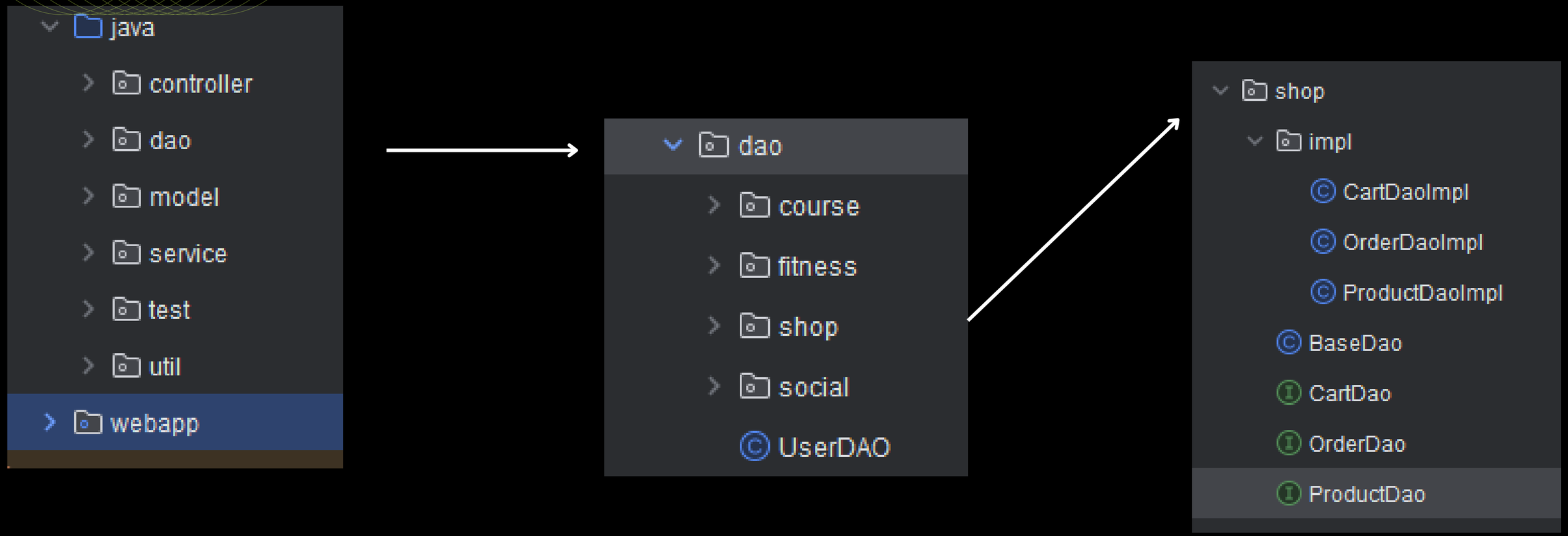
# Database

```sql
CREATE TABLE [products] (
    [product_id] INT PRIMARY KEY IDENTITY(1, 1),
    [name] VARCHAR(255) NOT NULL,
    [description] VARCHAR(MAX),
    [price] DECIMAL(10,2) NOT NULL,
    [stock_quantity] INT NOT NULL,
    [category_id] INT,
    [image_url] VARCHAR(500),
    [created_at] DATETIME DEFAULT GETDATE(),
    [updated_at] DATETIME DEFAULT GETDATE()
)
GO
```

```sql
CREATE TABLE [cart] (
    [cart_id] INT PRIMARY KEY IDENTITY(1, 1),
    [user_id] INT,
    [product_id] INT,
    [quantity] INT NOT NULL DEFAULT (1),
    [created_at] DATETIME DEFAULT GETDATE()
)
GO
```

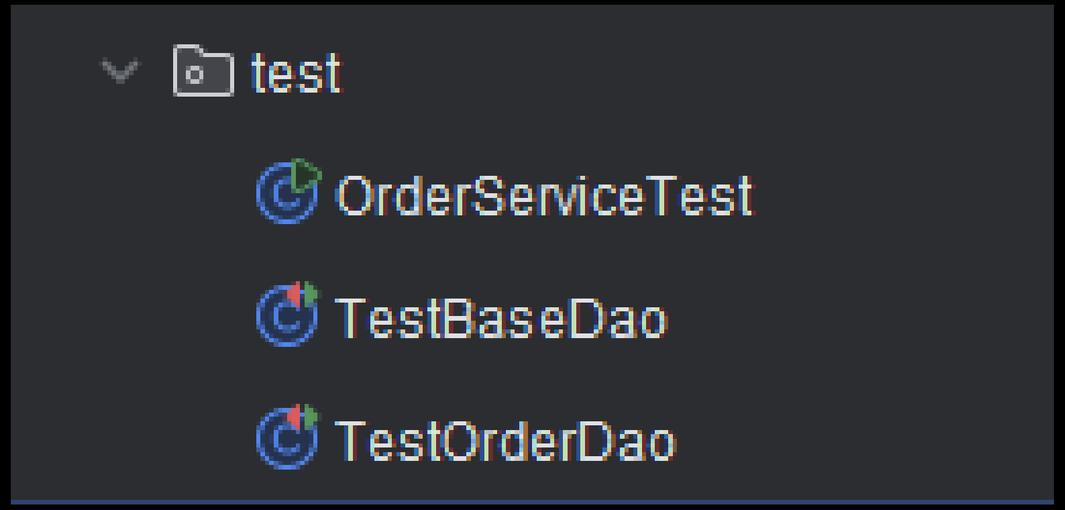
```sql
CREATE TABLE [orders] (
    [order_id] INT PRIMARY KEY IDENTITY(1, 1),
    [user_id] INT,
    [total_amount] DECIMAL(10,2) NOT NULL,
    [status_id] INT NOT NULL FOREIGN KEY REFERENCES [order_status]([status_id]),
    [created_at] DATETIME DEFAULT GETDATE()
)
GO
```

# CRUD




重複代碼分開，利用繼承減少代碼

# 單元測試




利用JUNIT新增單元測試代碼，方便除錯

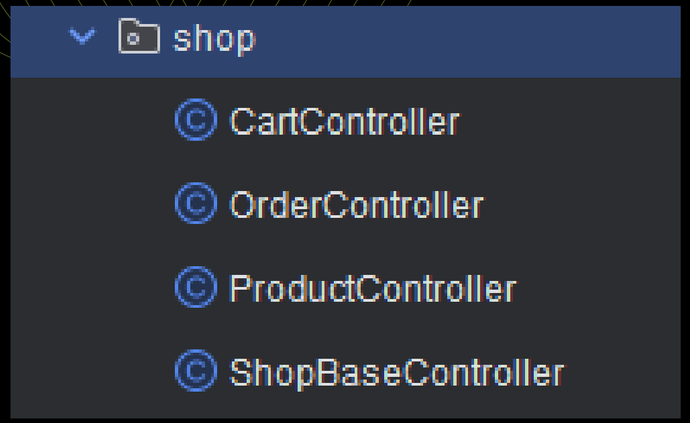
# 技術邏輯

```
     */
    int addProduct(Product product);  2 usages   1 implementation   pa00110059

    /**
     * 用於查詢數據庫中所有的產品記錄
     * @return 返回一個List集合,集合中存儲所有的產品記錄
     */
```

```
@Override  2 usages   pa00110059
public int addProduct(Product product) {
    String sql = "INSERT INTO products (name, description, price, stock_quantity, category_id, image_url, created_at, updated_at) VALUES (?, ?, ?, ?, ?, ?, ?,
    return baseUpdate(sql, product.getName(), product.getDescription(), product.getPrice(), product.getStockQuantity(), product.getCategoryId(), product.getIm
}
```

介面及實作分開，方便其他組員調用及查閱

# 技術邏輯

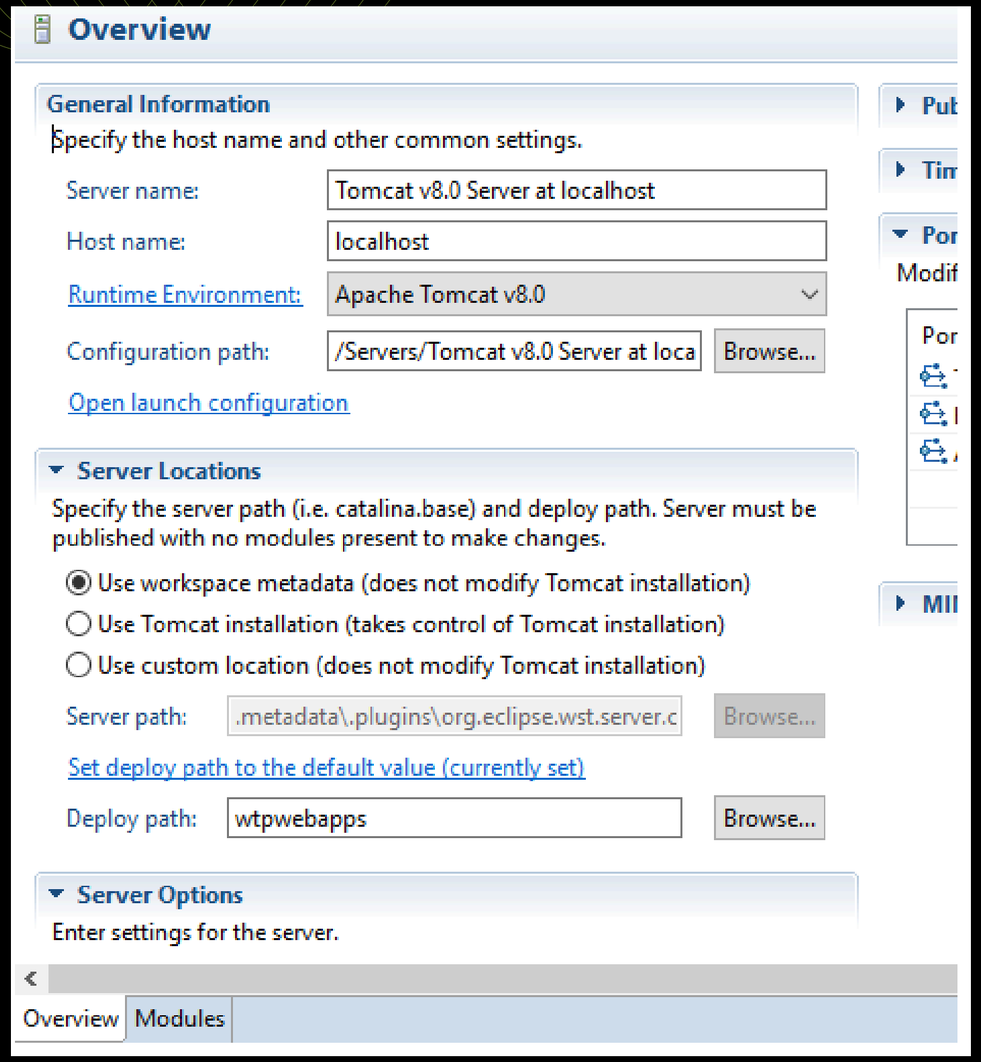


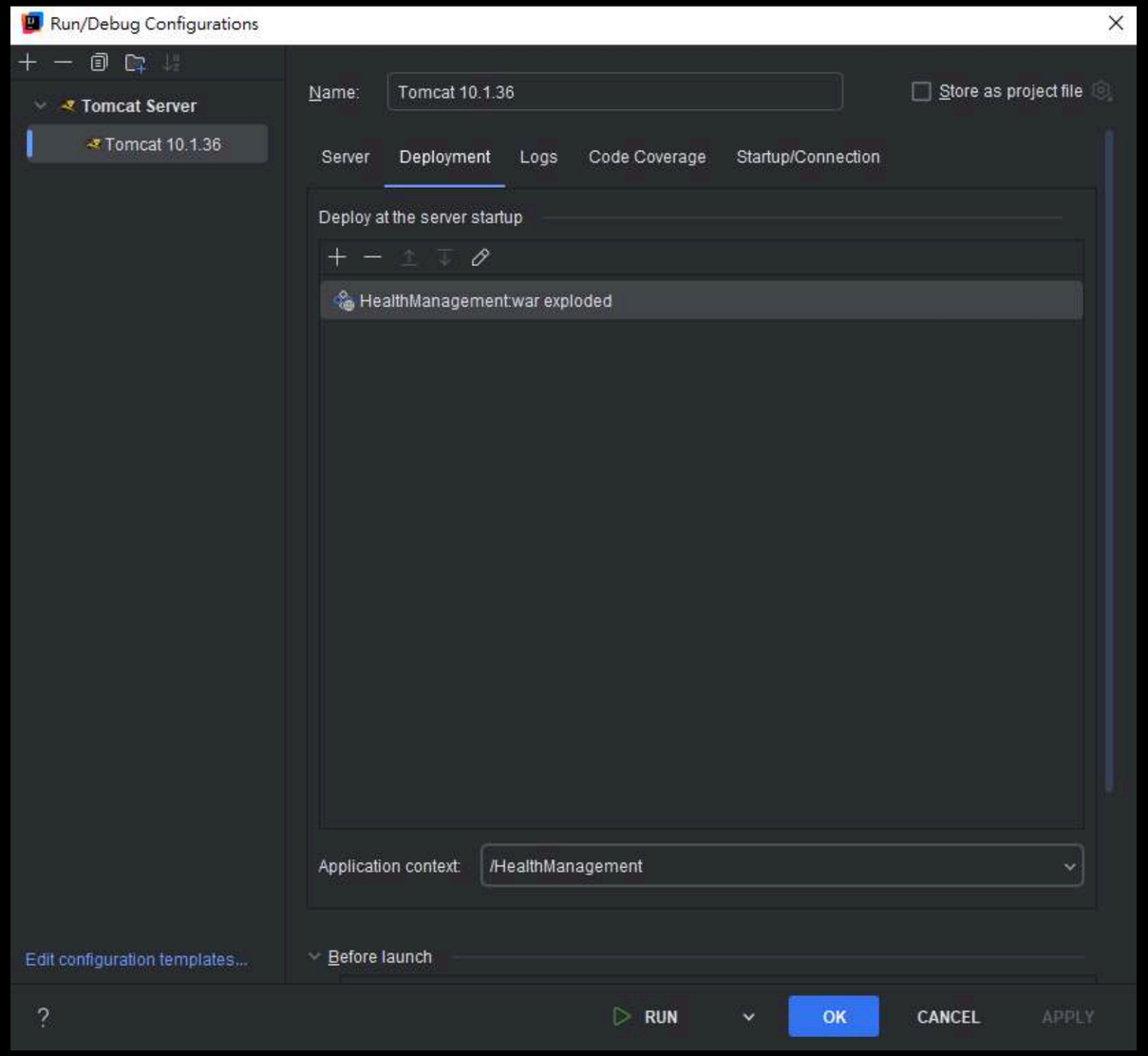

```
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.SerializationFeature;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;
```

```
// 回傳 JSON 資料
private void sendJsonResponse(HttpServletResponse res, Map<String, Object> responseData) throws IOException {  7 usages  pa00110059
    res.setContentType("application/json");
    res.setCharacterEncoding("UTF-8");
    PrintWriter out = res.getWriter();
    ObjectMapper objectMapper = new ObjectMapper();
    // 註冊 JavaTimeModule
    objectMapper.registerModule(new JavaTimeModule());
    // 以字串形式輸出日期，不使用數字時間戳
    objectMapper.disable(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS);

    out.write(objectMapper.writeValueAsString(responseData));
    out.flush();
}
```

使用JACKSON API以AJAX技術來回傳遞JSON格式的資料

# 技術難題




IDE佈署預設路徑不一致

# 技術難題

```
@AllArgsConstructor  29 usages  pa00110059
@NoArgsConstructor
@Data
public class Order implements Serializable {
```

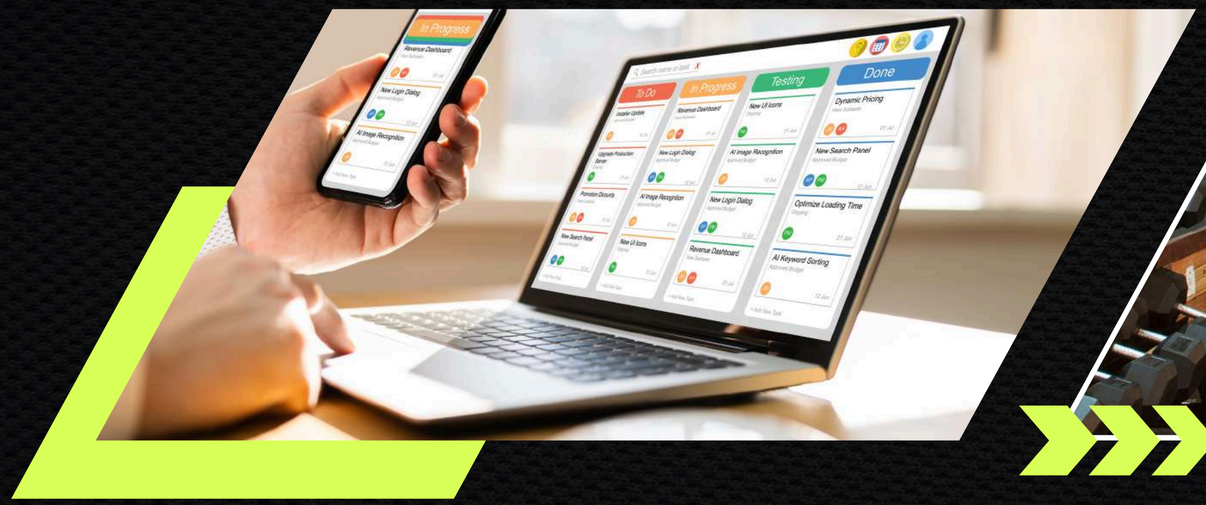


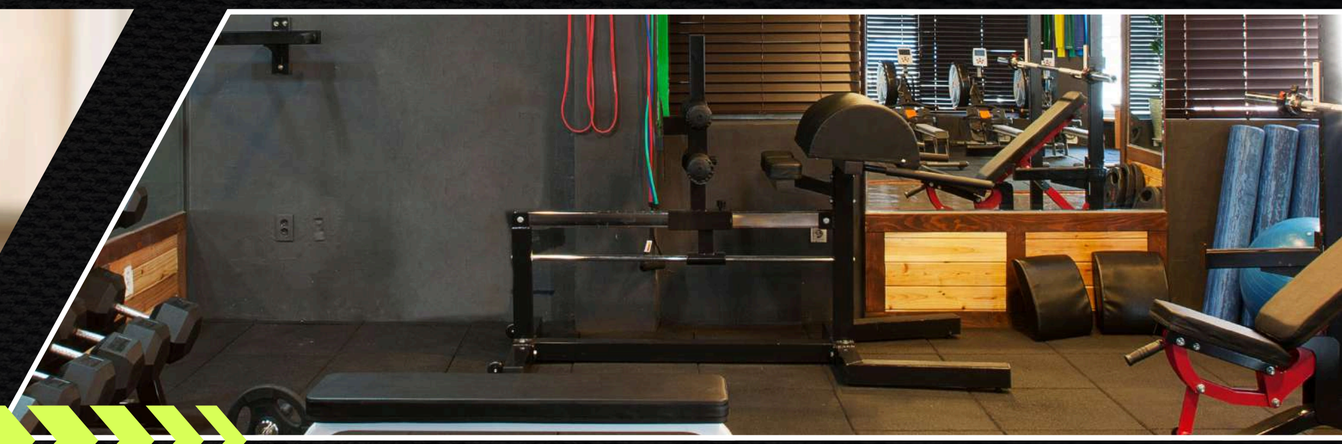
```
public class Order implements Serializable {

    private int orderId;


    private int userId;


    private BigDecimal totalAmount;


    private int statusId;


    private Timestamp createdAt;


    public int getOrderId() { return orderId; }

    public void setOrderId(int orderId) { this.orderId = orderId; }

    public int getUserId() { return userId; }

    public void setUserId(int userId) { this.userId = userId; }

    public BigDecimal getTotalAmount() { return totalAmount; }

    public void setTotalAmount(BigDecimal totalAmount) { this.totalAmount = totalAmount; }

    public int getStatusId() { return statusId; }

    public void setStatusId(int statusId) { this.statusId = statusId; }

    public Timestamp getCreatedAt() { return createdAt; }

    public void setCreatedAt(Timestamp createdAt) { this.createdAt = createdAt; }
```

LOMBOK API沒有辦法在不同IDE上實作

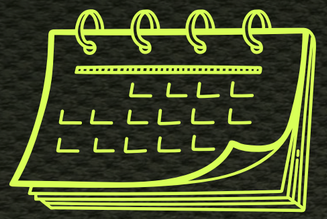
# 智慧課程管理

智慧化課程排程讓會員方便預約與取消。教練與課程評價系統，提升教練與學員互動。



# 提升預約效率



## 課程排程

省時方便，不再錯過任何健身機會。

## 評價系統

提升教練與學員互動品質。

## 熱門推薦

數據驅動，精準推薦熱門課程。

# 目錄 - 課程管理

# 資料庫結構設計

SQL建立TABLE

```
CREATE TABLE courses (
 course_id INT PRIMARY KEY,                          -- 課程ID
 course_name NVARCHAR(255) NOT NULL,                 -- 課程名稱
 course_description NVARCHAR(1000)NOT NULL,          -- 課程描述
 course_date DATE NOT NULL,                          -- 課程日期
 coach_id INT NOT NULL,                              -- 教練ID
 coach_name NVARCHAR(255) NOT NULL,                  -- 教練姓名
 course_time INT NOT NULL,                           -- 課程時長（分鐘）
 max_capacity INT NOT NULL,                          -- 最大容納人數
);
```

# CRUD 新增

```
protected void doInsert(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    String course_id = request.getParameter("course_id");
    String course_name = request.getParameter("course_name");
    String course_description = request.getParameter("course_description");
    String course_date = request.getParameter("course_date");
    String coach_id = request.getParameter("coach_id");
    String coach_name = request.getParameter("coach_name");
    String course_time = request.getParameter("course_time");
    String max_capacity = request.getParameter("max_capacity");
    try {
        Context context = new InitialContext();
        DataSource ds = (DataSource)context
                .lookup("java:/comp/env/jdbc/servdb");
        conn = ds.getConnection();
        String SQL = "INSERT INTO courses(course_id, course_name, course_description, course_date, coach_id, coach_name, course_time, max_capacity)"
                   + "VALUES(?,?,?,?,?,?,?,?)";
        PreparedStatement stmt = conn.prepareStatement(SQL);
        stmt.setString(1, course_id);
        stmt.setString(2, course_name);
        stmt.setString(3, course_description);
        stmt.setString(4, course_date);
        stmt.setString(5, coach_id);
        stmt.setString(6, coach_name);
        stmt.setString(7, course_time);
        stmt.setString(8, max_capacity);
        stmt.execute();
```

# CRUD 删除

```
protected void doDelete(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    String course_id = request.getParameter("course_id");
    try {
        Context context = new InitialContext();
        DataSource ds = (DataSource)context
                .lookup("java:/comp/env/jdbc/servdb");
        conn = ds.getConnection();
        String SQL = "DELETE FROM courses WHERE course_id = ?";
        PreparedStatement stmt = conn.prepareStatement(SQL);
        stmt.setString(1, course_id);
        stmt.execute();

        Course course = new Course();
        course.setcourse_id(course_id);

        String successMessage = "课程删除成功！";
        request.setAttribute("successMessage", successMessage);
        request.setAttribute("course", course);
        stmt.close();
        request.getRequestDispatcher("/jsp/course/delete.jsp")
            .forward(request, response);
```

# CRUD 查詢單筆

```
protected void doFind(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    String course_id = request.getParameter("course_id");
    Course course = null;
    try {
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");;
        conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);
        String SQL = "SELECT * FROM courses WHERE course_id = ?";
        PreparedStatement stmt = conn.prepareStatement(SQL);
        stmt.setString(1, course_id);
        ResultSet rs = stmt.executeQuery();
        if(rs.next()) {
            course = new Course();
            course.setcourse_id(rs.getString("course_id"));
            course.setcourse_name(rs.getString("course_name"));
            course.setcourse_description(rs.getString("course_description"));
            course.setcourse_date(rs.getString("course_date"));
            course.setcoach_id(rs.getString("coach_id"));
            course.setcoach_name(rs.getString("coach_name"));
            course.setcourse_time(rs.getString("course_time"));
            course.setmax_capacity(rs.getString("max_capacity"));
        }
        request.setAttribute("course", course);
        stmt.close();
        request.getRequestDispatcher("/jsp/course/schedule.jsp")
            .forward(request, response);
```

# CRUD 查詢整筆

```
protected void doFindAll(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    try {
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");;
        conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);
        String SQL = "SELECT * FROM courses";
        PreparedStatement stmt = conn.prepareStatement(SQL);
        ResultSet rs = stmt.executeQuery();
        List<Course> courses = new ArrayList<>();
        Course course = null;
        while(rs.next()) {
            course = new Course();
            course.setcourse_id(rs.getString("course_id"));
            course.setcourse_name(rs.getString("course_name"));
            course.setcourse_description(rs.getString("course_description"));
            course.setcourse_date(rs.getString("course_date"));
            course.setcoach_id(rs.getString("coach_id"));
            course.setcoach_name(rs.getString("coach_name"));
            course.setcourse_time(rs.getString("course_time"));
            course.setmax_capacity(rs.getString("max_capacity"));
            courses.add(course);
        }
        request.setAttribute("courses", courses);
        stmt.close();
        request.getRequestDispatcher("/jsp/course/allschedule.jsp")
            .forward(request, response);
```

# CRUD 修改

## 查詢單筆

```
protected void doUpdateData(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    String course_id = request.getParameter("course_id");
    Course course = null;
    try {
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");;
        conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);
        String SQL = "SELECT * FROM courses WHERE course_id = ?";
        PreparedStatement stmt = conn.prepareStatement(SQL);
        stmt.setString(1, course_id);
        ResultSet rs = stmt.executeQuery();
        if(rs.next()) {
            course = new Course();
            course.setcourse_id(rs.getString("course_id"));
            course.setcourse_name(rs.getString("course_name"));
            course.setcourse_description(rs.getString("course_description"));
            course.setcourse_date(rs.getString("course_date"));
            course.setcoach_id(rs.getString("coach_id"));
            course.setcoach_name(rs.getString("coach_name"));
            course.setcourse_time(rs.getString("course_time"));
            course.setmax_capacity(rs.getString("max_capacity"));
        }
        request.setAttribute("course", course);
        stmt.close();
        request.getRequestDispatcher("/jsp/course/updatedata.jsp")
```

## 使用查詢的資料修改

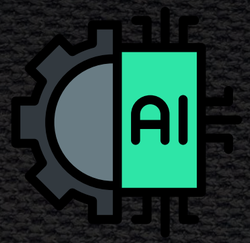
```
protected void doUpdate(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    String course_id = request.getParameter("course_id");
    String course_name = request.getParameter("course_name");
    String course_description = request.getParameter("course_description");
    String course_date = request.getParameter("course_date");
    String coach_id = request.getParameter("coach_id");
    String coach_name = request.getParameter("coach_name");
    String course_time = request.getParameter("course_time");
    String max_capacity = request.getParameter("max_capacity");
    try {
        Context context = new InitialContext();
        DataSource ds = (DataSource)context
                .lookup("java:/comp/env/jdbc/servdb");
        conn = ds.getConnection();
        String SQL = "UPDATE courses SET course_name = ?, course_description = ?"
                + ", course_date = ?, coach_id = ?, coach_name = ?, course_time = ?"
                + ", max_capacity = ? WHERE course_id = ?";
        PreparedStatement stmt = conn.prepareStatement(SQL);
        stmt.setString(8, course_id);
        stmt.setString(1, course_name);
        stmt.setString(2, course_description);
        stmt.setString(3, course_date);
        stmt.setString(4, coach_id);
        stmt.setString(5, coach_name);
        stmt.setString(6, course_time);
        stmt.setString(7, max_capacity);
        stmt.execute();

        Course course = new Course();
        course.setcourse_id(course_id);
        course.setcourse_name(course_name);
        course.setcourse_description(course_description);
        course.setcourse_date(course_date);
        course.setcoach_id(coach_id);
        course.setcoach_name(coach_name);
        course.setcourse_time(course_time);
        course.setmax_capacity(max_capacity);

        String successMessage = "課程修改成功！";
        request.setAttribute("successMessage", successMessage);
```

# 成效追蹤系統

會員透過APP記錄健身數據。AI 運動分析與個性化訓練建議。數據圖表視覺化，一目了然。


## 數據記錄

APP記錄健身數據。




## AI分析

AI運動分析建議。

## 視覺呈現

圖表視覺化數據。

# 目錄 - 成效追蹤

●●●建立table SQL指令

# 資料庫結構設計

```
CREATE TABLE exercise_records (
    record_id INT IDENTITY(1,1) PRIMARY KEY,
    user_id INT NOT NULL,
    exercise_type NVARCHAR(50) NOT NULL,
    exercise_duration INT NOT NULL, -- 運動時間（單位：分鐘）
    calories_burned numeric(12, 4) NOT NULL,    -- 消耗的卡路里
    exercise_date DATE NOT NULL,    -- 運動日期
    FOREIGN KEY (user_id) REFERENCES users(id)
    );
```

```
CREATE TABLE exercise_type_coefficients (
    ExerciseTypeID INT PRIMARY KEY IDENTITY(1,1),
    ExerciseName NVARCHAR(50) NOT NULL,    -- 運動名稱
    MET DECIMAL(5,2) NOT NULL              -- 該運動的 MET 值
);
```

```
身體記錄表（用戶端）
ATE TABLE BodyMetrics (
 BodyMetricID INT PRIMARY KEY IDENTITY(1,1), -- 自增ID作為主鍵
 UserID INT, -- 用戶ID
 Date DATETIME DEFAULT GETDATE(), -- 記錄的日期，默認為當前時間
 Weight DECIMAL(5,2), -- 體重 (kg)
 BodyFat DECIMAL(5,2), -- 體脂 (%)
 MuscleMass DECIMAL(5,2), -- 肌肉量 (kg)
 WaistCircumference DECIMAL(5,2), -- 腰圍 (cm)
 HipCircumference DECIMAL(5,2), -- 臀圍 (cm)
 Height DECIMAL(5,2), -- 身高 (cm)
 BMI DECIMAL(5,2), -- 身體質量指數
 FOREIGN KEY (UserID) REFERENCES users(id) -- 用戶ID作為外鍵，與
rs 表關聯
```

# CRUD

新增！

```java
// 新增運動紀錄
public boolean addExerciseRecord(ExerciseRecord record) {
    String sql = "INSERT INTO exercise_records (user_id, exercise_type, exercise_duration, calories_burned, exercise_date) " +
                 "VALUES (?, ?, ?, ?, ?)";
    try (PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setInt(1, record.getUserId());
        ps.setString(2, record.getExerciseType());
        ps.setInt(3, record.getExerciseDuration());
        ps.setDouble(4, record.getCaloriesBurned());
        ps.setDate(5, Date.valueOf(record.getExerciseDate()));
        int rowsAffected = ps.executeUpdate();
        return rowsAffected > 0;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}
```

```java
// 取得運動類型的 METS 值
public double getMETSValue(String exerciseType) throws SQLException {
    String query = "SELECT MET FROM exercise_type_coefficients WHERE ExerciseName = ?";
    try (PreparedStatement ps = conn.prepareStatement(query)) {
        ps.setString(1, exerciseType);  // 使用運動名稱
        try (ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                return rs.getDouble("MET");
            }
        }
    }
    return 0;  // 如果沒有找到對應的 METS 值，返回 0
}

}
```

計算卡路里！
(service)

```java
// 計算卡路里
public double calculateCaloriesBurned(ExerciseRecord record) throws SQLException {
    try {
        double userWeight = getUserWeight(record.getUserId());

        if (userWeight == 0) {
            throw new SQLException("User weight is zero or invalid!");
        }

        try (Connection conn = getConnection()) {
            ExerciseRecordDAO dao = new ExerciseRecordDAO(conn);
            double metValue = dao.getMETSValue(record.getExerciseType());

            if (metValue == 0) {
                throw new SQLException("METS value not found for exercise type: " + record.getExerciseType());
            }

            double exerciseDurationInHours = record.getExerciseDuration() / 60.0;
            return userWeight * metValue * exerciseDurationInHours;
        }

    } catch (SQLException e) {
        System.out.println("Error calculating calories: " + e.getMessage());
        throw e;
    }
}

// 獲取用戶體重
private double getUserWeight(int userId) throws SQLException {
    String sql = "SELECT Weight FROM BodyMetrics WHERE UserID = ?";
    try (Connection conn = getConnection(); PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setInt(1, userId);
        try (ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                return rs.getDouble("Weight");
            } else {
                throw new SQLException("user ID: " + userId +" 請先輸入體重");
            }
        }
    }
```

# CRUD

修改！

```java
// 更新運動紀錄
public boolean updateExerciseRecord(ExerciseRecord record) {
    String query = "UPDATE exercise_records SET exercise_type = ?, exercise_duration = ?, calories_burned = ?, exercise_date = ? WHERE record_id = ?";
    try (PreparedStatement ps = conn.prepareStatement(query)) {
        ps.setString(1, record.getExerciseType());
        ps.setInt(2, record.getExerciseDuration());
        ps.setDouble(3, record.getCaloriesBurned());
        ps.setDate(4, Date.valueOf(record.getExerciseDate()));
        ps.setInt(5, record.getRecordId());
        int rowsAffected = ps.executeUpdate();
        return rowsAffected > 0;
    } catch (SQLException e) {
        e.printStackTrace();
```

使用JOIN查詢！

```java
// 查詢用戶的所有運動紀錄
public List<ExerciseRecord> getExerciseRecords(int userId) {
    String sql = "SELECT e.record_id, e.user_id, e.exercise_type, e.exercise_duration, e.calories_burned, e.exercise_date, u.name " +
                 "FROM exercise_records e " +
                 "JOIN users u ON e.user_id = u.id " +
                 "WHERE e.user_id = ?";
    List<ExerciseRecord> records = new ArrayList<>();

    try (PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setInt(1, userId);
        try (ResultSet rs = ps.executeQuery()) {
            while (rs.next()) {
                ExerciseRecord record = new ExerciseRecord();
                record.setRecordId(rs.getInt("record_id"));
                record.setUserId(rs.getInt("user_id"));
                record.setExerciseType(rs.getString("exercise_type"));
                record.setExerciseDuration(rs.getInt("exercise_duration"));
                record.setCaloriesBurned(rs.getDouble("calories_burned"));
                record.setExerciseDate(rs.getDate("exercise_date").toString());

                // 设置用户信息
                User user = new User();
                user.setName(rs.getString("name"));
                record.setUser(user);

                records.add(record);
```

刪除！

```java
// 刪除運動紀錄
public boolean deleteExerciseRecord(int recordId) {
    String sql = "DELETE FROM exercise_records WHERE record_id = ?";
    try (PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setInt(1, recordId);
        int rowsAffected = ps.executeUpdate();
        return rowsAffected > 0;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}
```

# CRUD

```java
// 查詢用戶的所有運動紀錄，根據用戶名字進行模糊查詢
public List<ExerciseRecord> getExerciseRecordsByName(String name) {
    String sql = "SELECT e.record_id, e.user_id, e.exercise_type, e.exercise_duration, e.calories_burned, e.exercise_date, u.name " +
                 "FROM exercise_records e " +
                 "JOIN users u ON e.user_id = u.id " +
                 "WHERE u.name LIKE ?";
    List<ExerciseRecord> records = new ArrayList<>();

    try (PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setString(1, "%" + name + "%");  // 模糊匹配名字
        try (ResultSet rs = ps.executeQuery()) {
            while (rs.next()) {
                ExerciseRecord record = new ExerciseRecord();
                record.setRecordId(rs.getInt("record_id"));
                record.setUserId(rs.getInt("user_id"));
                record.setExerciseType(rs.getString("exercise_type"));
                record.setExerciseDuration(rs.getInt("exercise_duration"));
                record.setCaloriesBurned(rs.getDouble("calories_burned"));
                record.setExerciseDate(rs.getDate("exercise_date").toString());

                // 设置用户信息
                User user = new User();
                user.setName(rs.getString("name"));
                record.setUser(user);

                records.add(record);
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return records;
}
```

使用like語法進行模糊查詢

查詢所有用戶的所有資料

```java
//查詢所有用戶的所有運動紀錄
public List<ExerciseRecord> getAllExerciseRecords() {
    String sql = "SELECT e.record_id, e.user_id, e.exercise_type, e.exercise_duration, e.calories_burned, e.exercise_date, u.name " +
                 "FROM exercise_records e " +
                 "JOIN users u ON e.user_id = u.id";  // 聯接查詢運動紀錄和用戶表
    List<ExerciseRecord> records = new ArrayList<>();
```

# Service

## 調用DAO方法

```java
    // 刪除運動紀錄
    public boolean deleteExerciseRecord(int recordId) throws SQLException {
        try (Connection conn = getConnection()) {
            ExerciseRecordDAO dao = new ExerciseRecordDAO(conn);
            return dao.deleteExerciseRecord(recordId);
        }
    }

    // 查詢用戶的所有運動紀錄
    public List<ExerciseRecord> getExerciseRecords(int userId) throws SQLException {
        try (Connection conn = getConnection()) {
            ExerciseRecordDAO dao = new ExerciseRecordDAO(conn);
            return dao.getExerciseRecords(userId);
        }
    }

    // 查詢所有用戶的所有運動資訊
    public List<ExerciseRecord> getAllExerciseRecords() throws SQLException {
        try (Connection conn = getConnection()) {
            ExerciseRecordDAO dao = new ExerciseRecordDAO(conn);
            return dao.getAllExerciseRecords();
        }
    }
    // 查詢用戶的運動資訊模糊查詢
        public List<ExerciseRecord> getExerciseRecordByName(String name) throws SQLException {
            try (Connection conn = getConnection()) {
                ExerciseRecordDAO dao = new ExerciseRecordDAO(conn);
                return dao.getExerciseRecordsByName(name);
            }
        }
}
```

```java
    // 新增運動紀錄
    public boolean addExerciseRecord(ExerciseRecord record) throws SQLException {
        double caloriesBurned = calculateCaloriesBurned(record);
        record.setCaloriesBurned(caloriesBurned);

        try (Connection conn = getConnection()) {
            ExerciseRecordDAO dao = new ExerciseRecordDAO(conn);
            return dao.addExerciseRecord(record);
        }
    }

    // 更新運動紀錄
    public boolean updateExerciseRecord(ExerciseRecord record) throws SQLException {
        double caloriesBurned = calculateCaloriesBurned(record);
        record.setCaloriesBurned(caloriesBurned);

        try (Connection conn = getConnection()) {
            ExerciseRecordDAO dao = new ExerciseRecordDAO(conn);
            return dao.updateExerciseRecord(record);
        }
    }
```

# MVC 架構 model

```java
package model.fitness;

import model.User;

public class ExerciseRecord {
    private Integer  recordId;
    private Integer userId;
    private String exerciseType;
    private int exerciseDuration;
    private double caloriesBurned;
    private String exerciseDate;
    private User user;


    public ExerciseRecord(Integer recordId,Integer userId, String exerciseType, int exerciseDuration,
                          double caloriesBurned, String exerciseDate) {
        super();
        this.recordId = recordId;
        this.userId = userId;
        this.exerciseType = exerciseType;
        this.exerciseDuration = exerciseDuration;
        this.caloriesBurned = caloriesBurned;
        this.exerciseDate = exerciseDate;

    }

    public ExerciseRecord() {
        // Default constructor
    }
```

宣告user類型變數

getter/setter!!!!

# MVC 架構 controller

- 顯示所有運動紀錄
- 根據運動紀錄的 ID 顯示並更新運動紀錄
- 新增運動紀錄
- 顯示用戶清單和基於用戶 ID 或名稱來查詢運動紀錄

```java
// 處理 GET 請求，顯示運動紀錄和用戶清單
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    try {
        String action = request.getParameter("action");

        // 如果 action 是 'all'，顯示所有運動紀錄
        if ("all".equals(action)) {
            List<ExerciseRecord> records = exerciseRecordService.getAllExerciseRecords();
            request.setAttribute("records", records);
            RequestDispatcher dispatcher = request.getRequestDispatcher("/jsp/fitness/exerciseRecords.jsp");
            dispatcher.forward(request, response);

        // 如果 action 是 'update'，處理更新操作
        } else if ("update".equals(action)) {
            int recordId = Integer.parseInt(request.getParameter("recordId"));
            try (Connection conn = exerciseRecordService.getConnection()) {
                ExerciseRecordDAO dao = new ExerciseRecordDAO(conn);
                ExerciseRecord record = dao.getExerciseRecordById(recordId);
                if (record == null) {
                    throw new ServletException("找不到運動紀錄，ID: " + recordId);
                }
                request.setAttribute("record", record);
                RequestDispatcher dispatcher = request.getRequestDispatcher("/jsp/fitness/updateExerciseRecord.jsp");
                dispatcher.forward(request, response);
            } catch (Exception e) {
                handleError(e, request, response);
            }

        // 如果 action 是 'add'，處理新增操作
        } else if ("add".equals(action)) {
            User user = getUserFromRequest(request);
            request.setAttribute("userId", user.getId());
            RequestDispatcher dispatcher = request.getRequestDispatcher("/jsp/fitness/addExerciseRecord.jsp");
            dispatcher.forward(request, response);
```

```java
        // 如果沒有提供 'action' 或其他條件，根據 userId 或 name 查詢
        } else {
            String userIdStr = request.getParameter("userId");
            String name = request.getParameter("name");

            // 驗證：如果 userId 和 name 都沒有填寫，設置錯誤訊息
            if ((userIdStr == null || userIdStr.isEmpty()) && (name == null || name.isEmpty())) {
                request.setAttribute("search", "至少填寫一個欄位：用戶 ID 或 用戶名字");
                // 如果都沒填，轉發回查詢頁面
                RequestDispatcher dispatcher = request.getRequestDispatcher("/jsp/fitness/findUserRecords.jsp");
                dispatcher.forward(request, response);
                return; // 防止繼續執行下面的邏輯
            }

            List<ExerciseRecord> records = null;
            List<User> users = userService.getAllUsers();

            if (userIdStr != null && !userIdStr.isEmpty()) {
                // 根據 userId 查詢
                int userId = Integer.parseInt(userIdStr);
                User user = userService.getUserById(userId);
                if (user == null) {
                    throw new ServletException("找不到該用戶，ID: " + userId);
                }
                records = exerciseRecordService.getExerciseRecords(user.getId());
                request.setAttribute("user", user);

            } else if (name != null && !name.isEmpty()) {
                // 根據用戶名字進行模糊查詢
                records = exerciseRecordService.getExerciseRecordByName(name);
                if (records == null || records.isEmpty()) {
                    // 如果沒有找到對應的紀錄，拋出一個 ServletException
                    throw new ServletException("找不到該用戶，名字: " + name);
                }
            }

            request.setAttribute("records", records);
            request.setAttribute("users", users);
            RequestDispatcher dispatcher = request.getRequestDispatcher("/jsp/fitness/findUserRecords.jsp");
            dispatcher.forward(request, response);
        }
    } catch (Exception e) {
        handleError(e, request, response);
    }
}
```

根據action處理不同操作！！！

# MVC 架構 controller

```java
// 處理 POST 請求，新增、更新或刪除運動紀錄
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String action = request.getParameter("action");
    try {
        if ("add".equals(action)) {
            addExerciseRecord(request, response);
        } else if ("update".equals(action)) {
            updateExerciseRecord(request, response);
        } else if ("delete".equals(action)) {
            deleteExerciseRecord(request, response);
        } else {
            throw new ServletException("無效的操作: " + action);
        }
    } catch (Exception e) {
        handleError(e, request, response);
    }
}
```

```java
// 新增運動紀錄
private void addExerciseRecord(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOExcep
    try {
        ExerciseRecord record = extractRecordFromRequest(request);
        double caloriesBurned = exerciseRecordService.calculateCaloriesBurned(record);
        record.setCaloriesBurned(caloriesBurned);

        User user = userService.getUserById(record.getUserId());
        if (user == null) {
            throw new ServletException("找不到用戶，ID: " + record.getUserId());
        }

        exerciseRecordService.addExerciseRecord(record);

        request.setAttribute("successMessage", "新增成功!");


        List<ExerciseRecord> records = exerciseRecordService.getExerciseRecords(record.getUserId());
        request.setAttribute("records", records);

            RequestDispatcher dispatcher = request.getRequestDispatcher("/jsp/fitness/findUserRecords.jsp");
            dispatcher.forward(request, response);

    } catch (Exception e) {
        handleError(e, request, response);
    }
}
```

# MVC 架構 controller

```java
// 刪除運動紀錄
private void deleteExerciseRecord(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    try {
        String recordIdStr = request.getParameter("recordId");
        String userIdStr = request.getParameter("userId");
        String action= request.getParameter("action");

        if (recordIdStr == null || recordIdStr.isEmpty() || userIdStr == null || userIdStr.isEmpty()) {
            throw new ServletException("刪除操作缺少必要的參數");
        }

        int recordId = Integer.parseInt(recordIdStr);
        int userId = Integer.parseInt(userIdStr);

        User user = userService.getUserById(userId);
        if (user == null) {
            throw new ServletException("找不到用戶，ID: " + userId);
        }

        exerciseRecordService.deleteExerciseRecord(recordId);

        request.setAttribute("successMessage", "刪除成功!");

        List<ExerciseRecord> records = exerciseRecordService.getExerci
        request.setAttribute("records", records);

            RequestDispatcher dispatcher = request.getRequestDispatche
            dispatcher.forward(request, response);

    } catch (Exception e) {
        handleError(e, request, response);
    }
}
```

```java
// 更新運動紀錄
private void updateExerciseRecord(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    try {
        ExerciseRecord record = extractRecordFromRequest(request);
        double caloriesBurned = exerciseRecordService.calculateCaloriesBurned(record);
        record.setCaloriesBurned(caloriesBurned);

        User user = userService.getUserById(record.getUserId());
        if (user == null) {
            throw new ServletException("找不到用戶，ID: " + record.getUserId());
        }

        exerciseRecordService.updateExerciseRecord(record);

        request.setAttribute("successMessage", "更新成功!");

        List<ExerciseRecord> records = exerciseRecordService.getExerciseRecords(record.getUserId());
        request.setAttribute("records", records);

            RequestDispatcher dispatcher = request.getRequestDispatcher("/jsp/fitness/findUserRecords.jsp");
            dispatcher.forward(request, response);

    } catch (Exception e) {
        handleError(e, request, response);
    }
}
```

# MVC 架構 controller

封裝成ExerciseRecord物件

```java
// 從請求中提取運動紀錄資料
private ExerciseRecord extractRecordFromRequest(HttpServletRequest request) throws ServletException {
    try {
        ExerciseRecord record = new ExerciseRecord();

        String userIdStr = request.getParameter("userId");
        if (userIdStr == null || userIdStr.isEmpty()) {
            throw new ServletException("用戶 ID 缺失或為空");
        }
        record.setUserId(Integer.parseInt(userIdStr));

        String recordIdStr = request.getParameter("recordId");
        if (recordIdStr != null && !recordIdStr.isEmpty()) {
            record.setRecordId(Integer.parseInt(recordIdStr));
        }

        record.setExerciseType(request.getParameter("exerciseType"));
        record.setExerciseDuration(Integer.parseInt(request.getParameter("exerciseDuration")));
        record.setExerciseDate(request.getParameter("exerciseDate"));

        return record;
    } catch (Exception e) {
        throw new ServletException("資料格式無效", e);
    }
}
```

# MVC 架構 view(jsp)


index.jsp

```jsp
</style>
</head>
<body>
    <div id="header">
        <h1>享健你，遇見更好的自己。</h1>
        <h2>你，今天健了嗎？</h2>
    </div>

    <!-- ✅ 導覽列 -->
    <div id="navigation">
        <ul>
            <li><a
                href="http://localhost:8080/HealthManagement/jsp/course/HealthManagement.jsp">首頁</a></li>
            <li><a
                href="http://localhost:8080/HealthManagement/jsp/membercenter.jsp"
                class="active">會員管理</a></li>
            <li><a href="#">商城購物</a></li>
            <li><a
                href="http://localhost:8080/HealthManagement/jsp/fitness/index.jsp">健身成效</a></li>
            <li><a
                href="http://localhost:8080/HealthManagement/jsp/course/index.jsp">課程管理</a></li>
            <li><a
                href="http://localhost:8080/HealthManagement/api/Social/post">社群論壇</a></li>
            <!-- ✅ 修正網址 -->
        </ul>
    </div>
    <br>
    <div align="center">
        <h2>運動數據記錄</h2>
        <div id="content">
        <ul>
            <!-- 新增運動紀錄的按鈕 -->
            <li>
                <form action="../../jsp/fitness/addExerciseRecord.jsp method="get">
                    <button type="submit">新增運動紀錄</button>
                </form>
            </li><hr>

            <!-- 查詢運動紀錄 -->
            <li>
                <form action="../../jsp/fitness/findUserRecords.jsp method="get">
                    <button type="submit">查詢用戶紀錄</button>
                </form>
            </li><hr>

            <!-- 查詢所有用戶運動紀錄 -->
            <li>
                <form action="../../api/fitness/progress method="get">
                    <button type="submit" name="action" value="all">所有用戶紀錄</button>
                </form>
            </li><hr>
        </ul>
    </div>
    </div>
    <div id="footer">
        <p>&copy;  2025 享健你. All Rights Reserved.</p>
    </div>
</body>
</html>
```

index.jsp | findUserRecords.jsp

```jsp
        <h1>運動紀錄</h1>
        <!-- 顯示錯誤訊息 -->
        <c:if test="${not empty errorMessage}">
            <div class="error-message">
                <strong>Error:</strong> ${errorMessage}
            </div>
        </c:if>

        <!-- 顯示成功訊息 -->
        <c:if test="${not empty successMessage}">
            <div style="color: #808080;">
                <c:choose>
                    <c:when test="${successMessage == 'Record Deleted Successfully'}">
                        <strong>Success:</strong> Record Deleted Successfully
                    </c:when>
                    <c:otherwise>
                    ${successMessage}
                    </c:otherwise>
                </c:choose>
            </div>
        </c:if>
        <!-- 查詢運動紀錄 -->
        <form action="../../api/fitness/progress" method="get">
            <div>
                <label for="userId">用戶 I D :</label> <input type="text"
                    name="userId" placeholder="Enter User ID">
                <button type="submit">查詢</button>
            </div>
            <br> <br>
            <div>
                <label for="name">用戶名字:</label> <input type="text" name="name"
                    placeholder="Enter User Name">
                <button type="submit">查詢</button>
            </div>

        </form>
        <!-- 如果兩者都沒填，顯示錯誤訊息 -->
        <c:if test="${not empty search}">
```

rciseRecords.jsp

HealthManagement/src/main/webapp/jsp/fitness/exerciseRecords.jsp

```jsp
        <h1>運動紀錄</h1>
        <!--
        <c:i
            </div>
        </c:if>

        <!-- 顯示成功訊息 -->
        <c:if test="${not empty successMessage}">
            <div class="alert-success">
                <c:choose>
                    <c:when test="${successMessage == 'Record Deleted Successfully'}">
                        <strong>Success:</strong> Record Deleted Successfully
                    </c:when>
                    <c:otherwise>
                    ${successMessage}
                    </c:otherwise>
                </c:choose>
            </div>
        </c:if>

        <!-- 顯示運動紀錄 -->
        <c:if test="${not empty records}">
            <table>
                <thead>
                    <tr>
                        <th>用戶ID</th>
                        <th>用戶姓名</th>
                        <th>運動類型</th>
                        <th>運動時長（分鐘）</th>
                        <th>卡路里消耗</th>
                        <th>日期</th>
                        <th>管理</th>
                    </tr>
                </thead>
                <tbody>
                    <c:forEach var="record" items="${records}">
                        <tr>
                            <td>${record.userId}</td>
                            <td>${record.user.name}</td>
                            <td>${record.exerciseType}</td>
                            <td>${record.exerciseDuration}</td>
                            <td>${record.caloriesBurned}</td>
                            <td>${record.exerciseDate}</td>
                            <td>
                                <!-- 更新按鈕，會跳轉到更新頁面 --> <a
                                href="../../api/fitness/progress?action=update&recordId=${record.recordId}&
                                    <input type="button" value="更新" class="update">
                                </a> <!-- 刪除表單 -->
                                    <form action="../../api/fitness/progress" method="post"
                                        style="display: inline;">
                                        <input type="hidden" name="action" value="delete"> <input
                                            type="hidden" name="recordId" value="${record.recordId}">
                                        <input type="hidden" name="userId" value="${record.userId}">
                                        <input type="submit" value="刪除" class="delete">
                                    </form>
```

# MVC 架構 view(jsp)

.jsp | findUserRecords.jsp | updateExerciseRecord.jsp

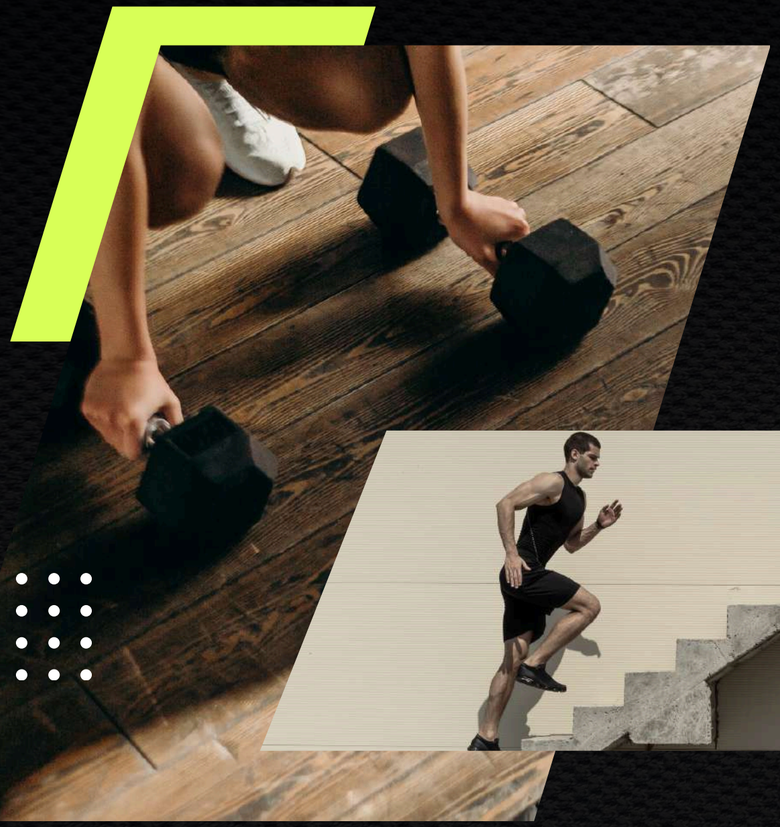
```jsp
<div class="form-container">
    <form action="../../api/fitness/progress" method="post">
        <input type="hidden" name="action" value="update"> <input
            type="hidden" name="recordId" value="${record.recordId}"> <input
            type="hidden" name="userId" value="${record.userId}">

        <table border="1">
            <tr>
                <td>運 動 類 型：</td>
                <td><select name="exerciseType" required>
                        <option value="瑜伽" selected>瑜伽 (Yoga)</option>
                        <option value="重訓">重訓 (Weight Training)</option>
                        <option value="有氧">有氧 (Cardio)</option>
                    </select></td>
            </tr>
            <tr>
                <td>運動時長 (分鐘):</td>
                <td><input type="number" id="exerciseDuration"
                    name="exerciseDuration" value="${record.exerciseDuration}"
                    required min="1"></td>
            </tr>
            <tr>
                <td>運 動 日 期:</td>
                <td><input type="date" id="exerciseDate" name="exerciseDate"
                    value="${record.exerciseDate}" required></td>
            </tr>
        </table>
        <input type="submit" value="確認">
    </form>


    <!-- 返回到紀錄頁面 -->
    <a class="back-button"
        href="../../api/fitness/progress?userId=${record.userId}">返回</a>
</div>
</div>
<div id="footer">
    <p>&copy; 2025 享健你. All Rights Reserved.</p>
```

index.jsp | findUserRecords.jsp | updateExerciseRecord.jsp | addExerciseRecord.jsp

```jsp
            <!-- ✅ 修正網址 -->
        </ul>
    </div>
    <br>
<body>
    <div id="content" align="center">
        <h1>新增運動單筆紀錄</h1>

        <!-- 用戶輸入 User ID 並添加運動紀錄 -->
        <form action="../../api/fitness/progress" method="post">
            <input type="hidden" name="action" value="add">
            <table border="1" id="">
                <tr>
                    <td>用 戶 I D：</td>
                    <td><input type="text" name="userId" required
                        placeholder="Enter User ID"></td>
                </tr>
                <tr>
                    <td>運 動 類 型：</td>
                    <td><select name="exerciseType" required>
                            <option value="瑜伽" selected>瑜伽 (Yoga)</option>
                            <option value="重訓">重訓 (Weight Training)</option>
                            <option value="有氧">有氧 (Cardio)</option>
                        </select></td>
                </tr>
                <tr>
                    <td>運 動 時 長(分鐘)：</td>
                    <td><input type="number" name="exerciseDuration" required
                        placeholder="Enter Duration" value="20"></td>
                </tr>
                <tr>
                    <td>運 動 日 期：</td>
                    <td><input type="date" name="exerciseDate" required
                        id="exerciseDate"></td>
                </tr>

            </table>

            <button type="submit">確認</button>

        </form>


        <script>
            // 獲取當天日期
            const today = new Date().toISOString().split('T')[0]; // 轉換為 'YYYY-MM-DD' 格式
            document.getElementById('exerciseDate').value = today; // 設置為 input 的預設值
        </script>

        <!-- 返回主畫面的按鈕 -->
        <a href="http://localhost:8080/HealthManagement/jsp/fitness/index.jsp"
            class="back-button">返回</a>
```

# 社群互動平台

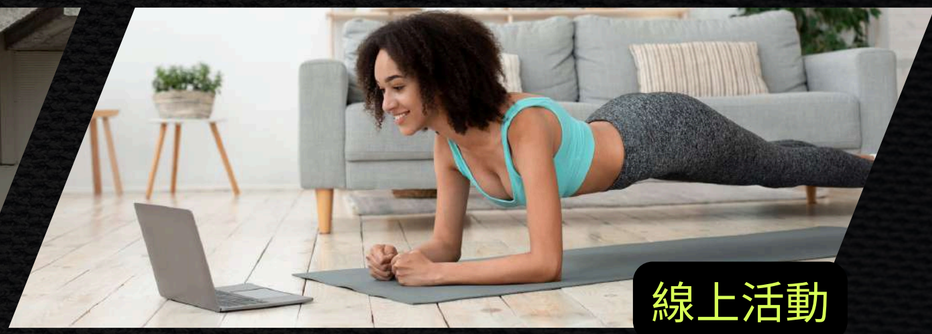
會員分享健身心得與進步成果。討論區、打卡挑戰、線上社群活動。教練與會員之間的即時交流平台。




心得分享



打卡挑戰



線上活動

# 資料庫結構設計

代號、主題、內容、使用者ID、時間

社群資料庫.sql - TIM...3121\tim12 (54))

```
4
5  CREATE TABLE social_posts (
6      article_id INT IDENTITY(1,1) PRIMARY KEY,
7      title NVARCHAR(255) NOT NULL,
8      content NVARCHAR(MAX) NOT NULL,
9      user_id INT NOT NULL,
10     publish_date DATETIME DEFAULT GETDATE()
11 );
12
13 INSERT INTO social_posts (title, content, user_id) VALUES
14 ('訓練後如何恢復', '運動後的恢復至關重要，確保攝取足夠的蛋白質來促進肌肉修復。此外，適當的伸展和泡沫滾筒按摩可以幫助減
15 ('重量訓練入門指南', '對於新手來說，選擇合適的重量並掌握正確的動作至關重要。建議從基本動作如深蹲、硬舉和臥推開始，並按
16 ('徒手訓練的好處', '徒手訓練不需要器械，適合在任何地方進行。俯臥撐、引體向上、深蹲和波比跳等動作能有效鍛煉全身肌群。透
17 ('如何增加肌耐力', '提升肌耐力需要高重複次數的訓練，建議每組 12-15 下，並減少組間休息時間。此外，有氧運動如跳繩、跑步
18 ('健身房禮儀須知', '進入健身房後，請務必遵守基本禮儀，例如使用器材後擦拭汗水、不要長時間占用器材、避免大聲喧嘩或將就
```

100 %

結果 訊息

| | article_id | title | content | user_id | publish_date |
|---|---|---|---|---|---|
| 1 | 1 | 訓練後如何恢復 | 運動後的恢復至關重要，確保攝取足夠的蛋白質來促進肌肉修復。此外，適當的伸... | 832746 | 2025-03-04 06:47:17.210 |
| 2 | 2 | 重量訓練入門指南 | 對於新手來說，選擇合適的重量並掌握正確的動作至關重要。建議從基本動作如深... | 729185 | 2025-03-04 06:47:17.210 |
| 3 | 3 | 徒手訓練的好處 | 徒手訓練不需要器械，適合在任何地方進行。俯臥撐、引體向上、深蹲和波比跳等... | 591237 | 2025-03-04 06:47:17.210 |
| 4 | 4 | 如何增加肌耐力 | 提升肌耐力需要高重複次數的訓練，建議每組 12-15 下，並減少組間休息時間。此... | 874593 | 2025-03-04 06:47:17.210 |
| 5 | 5 | 健身房禮儀須知 | 進入健身房後，請務必遵守基本禮儀，例如使用器材後擦拭汗水、不要長時間占用... | 605138 | 2025-03-04 06:47:17.210 |
| 6 | 6 | 深蹲為何這麼重要 | 深蹲是一項全身性的動作，能夠強化腿部與核心肌群，並提升爆發力與穩定性。正... | 746829 | 2025-03-04 06:47:17.210 |

# CRUD

## 新增

```java
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String pathInfo = request.getPathInfo();
    try {
        String title = request.getParameter("title");
        String content = request.getParameter("content");
        int userId = 1; // 假設的 ID，可從 session 取得

        Date publishDate = new Date();
        socialService.addPost(new SocialPost(0, title, content, userId, publishDate));

        response.sendRedirect(request.getContextPath() + "/api/Social/post");
    } catch (Exception e) {
        e.printStackTrace();
        response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR, "❌ 伺服器錯誤：無法新增文章");
    }
}
```

- 讀取使用者id來新增文章

## 刪除

```java
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    try {
        int articleId = Integer.parseInt(request.getParameter("articleId"));
        socialService.deletePost(articleId);
        response.sendRedirect(request.getContextPath() + "/api/Social/post"); // 刪除後重新導向
    } catch (Exception e) {
        e.printStackTrace();
        response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR, "❌ 伺服器錯誤：無法刪除文章");
    }
}
```

# CRUD

## 修改

```java
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    try {
        int articleId = Integer.parseInt(request.getParameter("articleId"));
        String title = request.getParameter("title");
        String content = request.getParameter("content");

        socialService.updatePost(articleId, title, content);
        response.sendRedirect(request.getContextPath() + "/api/Social/post"); // 更新後重新導向
    } catch (Exception e) {
        e.printStackTrace();
        response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR, "❌ 伺服器錯誤：無法更新文章");
    }
}
```

## 查詢

```java
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String query = request.getParameter("searchQuery");

    if (query == null || query.trim().isEmpty()) {
        request.setAttribute("errorMessage", "請輸入關鍵字進行查詢！");
    } else {
        List<SocialPost> searchResults = socialService.searchPosts(query);
        if (searchResults.isEmpty()) {
            request.setAttribute("errorMessage", "查無符合條件的文章！");
        }
        request.setAttribute("searchResults", searchResults);
    }

    request.getRequestDispatcher("/jsp/social/search.jsp").forward(request, response);
}
```

- 取得請求參數
- 輸入檢查
- 執行搜尋
- 結果處理

# CRUD

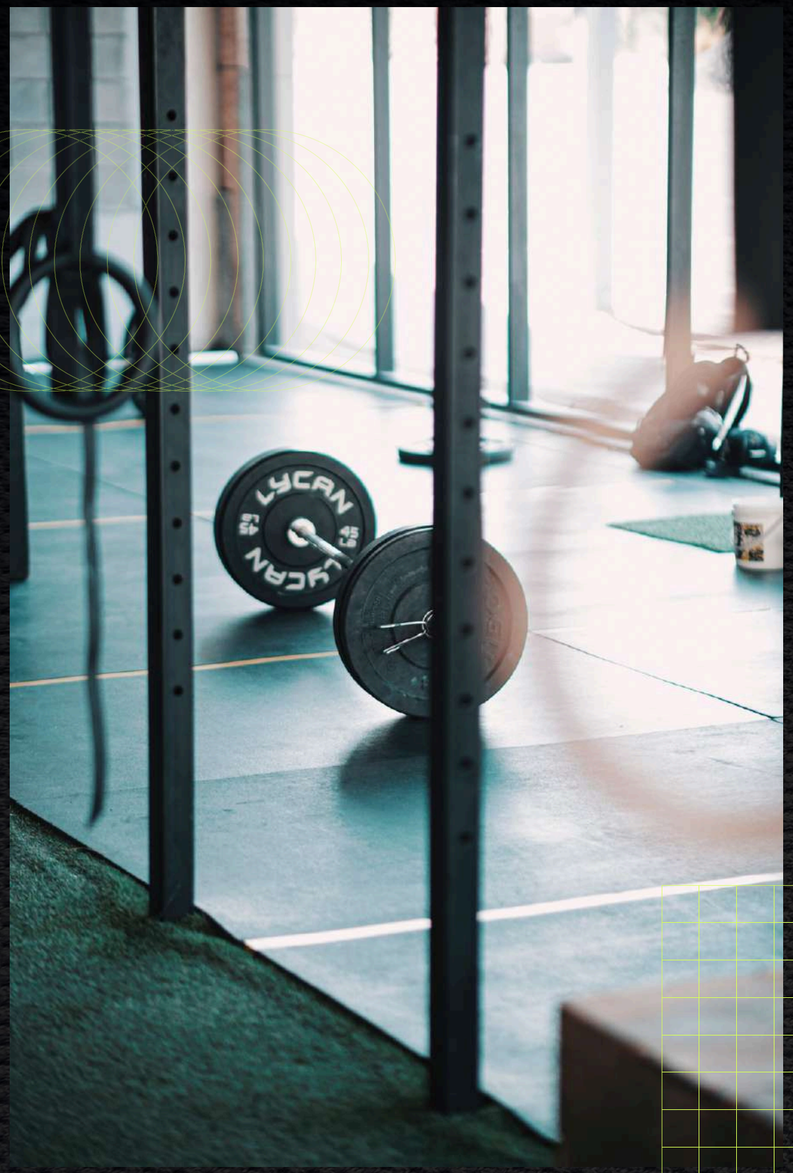

```jsp
<h1>搜尋文章</h1>
<!-- 搜尋表單 -->
<form action="${pageContext.request.contextPath}/api/Social/search" method="get">
    <label>關鍵字：</label>
    <input type="text" name="searchQuery" required>
    <button type="submit" class="search-button">查詢</button>
</form>
<hr>
<!-- 顯示錯誤訊息 -->
<% if (errorMessage != null) { %>
    <p style="color: red;"><%= errorMessage %></p>
<% } %>
<!-- 只有在用戶點擊查詢後，且 `searchResults` 為空時，才顯示 `查無相關文章` -->
<% if (searched && (searchResults == null || searchResults.isEmpty())) { %>
    <p style="color: gray;"> </p>
<% } %>
<!-- 顯示查詢結果 -->
<% if (searched) { %> <!-- 只有當查詢後才顯示 -->
    <% if (searchResults != null && !searchResults.isEmpty()) { %>
        <ul id="postList">
            <% for (SocialPost post : searchResults) {
                String content = post.getContent();
                boolean isLongContent = content.length() > 50;
                String previewText = isLongContent ? content.substring(0, 50).trim() : content;%>
                <li class="post-item" data-id="<%= post.getArticleId() %>">
                <h3><!-- ◇ 文章標題 -->
                    <a href="${pageContext.request.contextPath}/jsp/social/postDetail.jsp?articleId=<%= post.getArticleId() %>">
                        <%= post.getTitle() %></a>
                </h3>
                <!-- ◇ 文章預覽（最多 50 個字） -->
                <p class="preview">
                    <%= previewText %><% if (isLongContent) { %><span class="ellipsis">...</span><a href="${pageContext.request.contextPath}/jsp/social/postDetail.
                </p>
                <!-- ◇ 文章資訊 -->
                <div class="post-meta">
                    <img src="${pageContext.request.contextPath}/images/user.svg" alt="User" width="16px" height="16px"> <%= post.getUserId() %>
                    <img src="${pageContext.request.contextPath}/images/clock.svg" alt="Clock" width="16px" height="16px"> <%= post.getFormattedPublishDate() %>
                </div>
            </li>
        <% } %>
    </ul>
<% } %>
    <!-- ◇ 查詢後才顯示「回到論壇」按鈕 -->
    <div class="buttons-container">
        <button onclick="goToForum()" class="search-button">回到論壇</button>
    </div>
```


- 主頁預覽文章超過100字隱藏
- 點選標題及閱讀更多即可進入文章

```jsp
<!-- ◦ 文章標題 （可點擊進入完整文章） -->
<h3>
    <a href="${pageContext.request.contextPath}/jsp/social/postDetail.jsp?articleId=<%= post.getArticleId() %>">
        <%= post.getTitle() %>
    </a>
</h3>

<!-- ◦ 文章預覽 （最多 100 個字） -->
<p class="preview">
    <%
        String content = post.getContent();
        boolean isLongContent = content.length() > 100;
        String previewText = isLongContent ? content.substring(0, 100).trim() : content;
    %>
    <%= previewText %><% if (isLongContent) { %><span class="ellipsis">...</span>
    <a href="${pageContext.request.contextPath}/jsp/social/postDetail.jsp?articleId=
    <%= post.getArticleId() %>" class="read-more">閱讀更多</a><% } %>
</p>
```


- 用迴圈連接上下筆文章

```jsp
<!-- 返回 & 分頁按鈕 -->
<div class="navigation-buttons">
    <%if (nextId > 0) {%>
    <a href="${pageContext.request.contextPath}/jsp/social/postDetail.jsp?articleId=
    <%= nextId %>"class="btn next-post">← 上一篇</a>
    <%} else {%>
    <a class="btn next-post disabled">← 上一篇</a>
    <%}%>

    <%if (prevId > 0) {%>
    <a href="${pageContext.request.contextPath}/jsp/social/postDetail.jsp?articleId=
    <%= prevId %>"class="btn prev-post">下一篇 →</a>
    <%} else {%>
    <a class="btn prev-post disabled">下一篇 →</a>
    <%}%>

    <a href="${pageContext.request.contextPath}/jsp/social/posts.jsp"
       class="btn back-button">BACK 返回論壇</a>
</div>
```

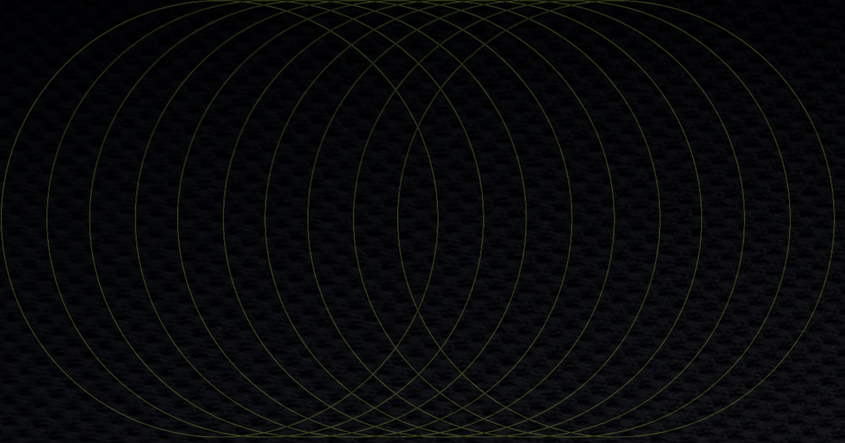



@reallygreatsite

reallygreatsite.com

hello@reallygreatsite.com

+123-456-7890



THANK
YOU!