spring.application.name=playball

# mysql éç½®

spring.datasource.url=jdbc:mysql://localhost:3306/playball?useSSL=false&serverTimezone=Asia/Taipei&useLegacyDatetimeCode=false

spring.datasource.username=root

spring.datasource.password=pETER2600

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl

spring.jpa.hibernate.naming.implicit-strategy=org.hibernate.boot.model.naming.ImplicitNamingStrategyLegacyJpaImpl

#redis

spring.redis.host=localhost

spring.redis.port=6379

spring.redis.database=1

2024/12/20 下午 06:25 <DIR> common

2024/12/20 下午 06:25 <DIR> controller

2024/12/20 下午 06:25 <DIR> dao

2024/12/20 下午 06:25 629 PlayballApplication.java

2024/12/20 下午 06:25 1,769 RedisConfig.java

2024/12/20 下午 06:25 <DIR> service

2024/12/20 下午 06:25 422 ServletInitializer.java

2024/12/20 下午 06:25 <DIR> vo

2024/12/20 下午 06:25 1,625 AdminFilter.java

2024/12/20 下午 06:25 1,669 UserFilter.java

2024/12/20 下午 06:25 1,440 AdminController.java

2024/12/20 下午 06:25 3,590 ReservationsController.java

2024/12/20 下午 06:25 3,289 UserController.java

2024/12/20 下午 06:25 231 AdminDAO.java

2024/12/20 下午 06:25 <DIR> impl

2024/12/20 下午 06:25 613 ReservationsDAO.java

2024/12/20 下午 06:25 389 UserDAO.java

2024/12/20 下午 06:25 153 AdminService.java

2024/12/20 下午 06:25 <DIR> impl

2024/12/20 下午 06:25 537 ReservationsService.java

2024/12/20 下午 06:25 348 UserService.java

2024/12/20 下午 06:25 1,343 AdminVO.java

2024/12/20 下午 06:25 2,635 ReservationsVO.java

2024/12/20 下午 06:25 1,710 UserVO.java

2024/12/20 下午 06:25 7,411 adminhome.html

2024/12/20 下午 06:25 2,061 adminlogin.html

2024/12/20 下午 06:25 7,738 home.html

2024/12/20 下午 06:25 <DIR> js

2024/12/20 下午 06:25 2,196 register.html

2024/12/20 下午 06:25 174,526 jquery.js

主要功能結構

前端 (HTML, JS)

用戶端的 UI，提供用戶界面和交互邏輯。

發送請求給後端的控制器。

控制器 (Controller)

控制器層接收前端請求，並根據請求調用相應的服務邏輯。

例如：UserController, AdminController, ReservationsController。

控制器將處理結果返回給前端。

業務邏輯層 (Service)

服務層處理具體業務邏輯，如用戶認證、資料處理、預約管理等。

例如：UserService, AdminService, ReservationsService。

數據層 (DAO)

DAO 層負責與資料庫交互，執行 CRUD（創建、讀取、更新、刪除）操作。

例如：UserDAO, AdminDAO, ReservationsDAO。

數據庫

系統使用 MySQL 作為數據庫，通過 JPA (Hibernate) 進行操作。

Redis 用於快速數據存儲和緩存。

流程說明

前端請求： 用戶通過網頁界面發送請求（如登入、註冊、預約等）。

控制器接收請求： 控制器接收到前端的請求後，將請求交給對應的業務邏輯服務（例如：UserController → UserService）。

業務邏輯處理： 業務邏輯服務（UserService）處理具體的邏輯，例如用戶註冊、登錄驗證、資料檢查等。

DAO 層與資料庫交互： 在業務邏輯處理過程中，DAO 層負責與資料庫進行交互，查詢或更新資料（例如：UserDAO進行用戶資料的查詢或更新）。

返回結果： 服務層將結果返回給控制器，然後控制器再將結果返回給前端，完成整個處理流程。

配置與技術棧

MySQL：用來存儲系統的數據，並透過 JPA（Hibernate）來操作。

Redis：用於存儲臨時或快取的資料。

Spring Boot：用來快速構建和配置應用。

前端 (HTML/JS)：用於與用戶進行交互。

Spring Security：如果需要的話，處理認證與授權（此部分根據需求可能會包括在服務層中）。

提示

若系統有高頻度資料存取，可能會需要更多的關於 Redis 和 MySQL 的集成詳情。

每一層（Controller、Service、DAO）都應該清楚分工，並且能夠對外提供簡單、清晰的 API 接口，便於維護和擴展。

簡報呈現建議

在簡報中，您可以將這些部分用流程圖或架構圖的形式呈現，並對每個層次的角色與功能進行簡單介紹。使用類似這樣的方式呈現會讓觀眾更容易理解系統的設計，並且能幫助他們快速理解各個模塊間的關係和數據流動過程。

如果有任何其他問題或需要調整的地方，隨時告訴我！

**主要功能結構**

**前端 (HTML, JS)**

**用戶端的 UI，提供用戶界面和交互邏輯。**

**發送請求給後端的控制器。**

**控制器 (Controller)**

**控制器層接收前端請求，並根據請求調用相應的服務邏輯。**

**例如：UserController, AdminController, ReservationsController。**

**控制器將處理結果返回給前端。**

**業務邏輯層 (Service)**

**服務層處理具體業務邏輯，如用戶認證、資料處理、預約管理等。**

**例如：UserService, AdminService, ReservationsService。**

**數據層 (DAO)**

**DAO 層負責與資料庫交互，執行 CRUD（創建、讀取、更新、刪除）操作。**

**例如：UserDAO, AdminDAO, ReservationsDAO。**

**數據庫**

**系統使用 MySQL 作為數據庫，通過 JPA (Hibernate) 進行操作。**

**Redis 用於快速數據存儲和緩存。**

**流程說明**

**前端請求： 用戶通過網頁界面發送請求（如登入、註冊、預約等）。**

**控制器接收請求： 控制器接收到前端的請求後，將請求交給對應的業務邏輯服務（例如：UserController → UserService）。**

**業務邏輯處理： 業務邏輯服務（UserService）處理具體的邏輯，例如用戶註冊、登錄驗證、資料檢查等。**

**DAO 層與資料庫交互： 在業務邏輯處理過程中，DAO 層負責與資料庫進行交互，查詢或更新資料（例如：UserDAO進行用戶資料的查詢或更新）。**

**返回結果： 服務層將結果返回給控制器，然後控制器再將結果返回給前端，完成整個處理流程。**

**配置與技術棧**

**MySQL：用來存儲系統的數據，並透過 JPA（Hibernate）來操作。**

**Redis：用於存儲臨時或快取的資料。**

**Spring Boot：用來快速構建和配置應用。**

**前端 (HTML/JS)：用於與用戶進行交互。**

**Spring Security：如果需要的話，處理認證與授權（此部分根據需求可能會包括在服務層中）。**

**提示**

**若系統有高頻度資料存取，可能會需要更多的關於 Redis 和 MySQL 的集成詳情。**

**每一層（Controller、Service、DAO）都應該清楚分工，並且能夠對外提供簡單、清晰的 API 接口，便於維護和擴展。**

**簡報呈現建議**

**在簡報中，您可以將這些部分用流程圖或架構圖的形式呈現，並對每個層次的角色與功能進行簡單介紹。使用類似這樣的方式呈現會讓觀眾更容易理解系統的設計，並且能幫助他們快速理解各個模塊間的關係和數據流動過程。**

**如果有任何其他問題或需要調整的地方，隨時告訴我！**

**你說：**

**您提供的系統架構流程能夠當作簡報其中一頁使用嗎**

**ChatGPT 說：**

**ChatGPT**

**是的，我提供的系統架構流程完全可以用作簡報的其中一頁。它將幫助觀眾清晰地了解您的系統設計，並且能夠簡潔明了地展示各個模塊如何互動和協作。以下是幾個關於簡報頁面設計的建議，您可以根據需求進行調整：**

**1. 視覺化圖表**

**您可以將我提供的文字描述轉化為更直觀的圖表。在簡報中，使用 流程圖 或 系統架構圖 來展現數據流動和模塊間的關係。這樣的圖表不僅能增強理解，還能讓您的簡報更加專業。**

**2. 簡單明瞭的文字補充**

**在圖表的下方或旁邊，可以加入簡單的文字補充說明，幫助解釋每一層的功能。比如：**

**前端 (HTML/JS)：負責與用戶進行交互，發送請求。**

**控制器 (Controller)：處理請求，並調用業務邏輯。**

**業務邏輯層 (Service)：處理具體業務邏輯。**

**數據層 (DAO)：與資料庫交互，執行 CRUD 操作。**

**3. 強調關鍵技術**

**在簡報頁面中，可以簡要列出使用的關鍵技術堆棧，如：**

**MySQL：數據存儲。**

**Redis：數據緩存。**

**Spring Boot：應用框架。**

**JPA (Hibernate)：資料庫操作。**

**這樣能幫助聽眾理解系統架構的具體實現和技術選擇。**

**4. 頁面設計建議**

**色彩搭配：使用適當的色彩區分不同的模塊（例如，控制器層可以用藍色，業務邏輯層用綠色，DAO層用黃色等），這樣讓各層級區分更加明顯。**

**簡潔的文字：保持每個部分的文字簡單明瞭，不要過多解釋技術細節，簡單概述每個模塊的功能即可。**

**範例結構：**

**標題：系統架構設計**

**圖表：將我提供的系統架構流程圖做成簡單的流程圖或層次結構圖。**

**文字補充：根據每一層簡單描述功能和責任，簡潔易懂。**

**如果需要幫助製作圖表或進一步調整結構，我也可以提供更具體的建議或範例！**