第 05 次作业：为校园选课系统添加数据库支持

项目介绍

本次作业延续第 04 次作业的校园选课系统，你需要将**内存存储升级为数据库持久化**方案，掌握 Spring Data JPA 与事务管理，并为后续的微服务改造打好数据层基础。

版本信息：

- 版本号：`v1.1.0`
- Git 分支：`main`
- 项目阶段：单体架构（数据库持久化）
- 基于版本：`v1.0.0`（第 04 次作业）

学习目标

- 配置 Spring Data JPA 与 MySQL/H2
- 为领域模型添加持久化映射
- 设计 Repository 接口与自定义查询
- 在 Service 层实现事务与数据校验
- 编写数据库初始化脚本与多环境配置
- 完成持久化相关的功能与集成测试

前置要求

必须完成

✅ **第 04 次作业**：具备课程、学生、选课的 REST API 与业务规则。

- 课程管理 API
- 学生管理 API
- 选课管理 API

环境准备

1. 准备 MySQL 8.0+（或兼容版本），确认服务可用。
2. 安装客户端工具（命令行或 GUI）以便执行 SQL。
3. 预留一个新的数据库实例（例如 `course_db`）。

> 若本地无法安装 MySQL，可在开发环境使用 H2 内存数据库，生产环境再切换至 MySQL。

核心任务

任务一：项目依赖与配置

- 在 pom.xml 添加：
  - spring-boot-starter-data-jpa
  - mysql-connector-j（runtime 作用域）
  - （可选）com.h2database:h2 便于本地开发
- 将数据库配置拆分为多环境：
  - application-dev.yml：使用 H2（内存库）
  - application.yml 或 application-prod.yml：指向 MySQL，并包含连接池、方言、DDL 策略等配置
- 配置 JPA 常用参数（如 ddl-auto, show-sql, hibernate.dialect）。
  - 建议：dev 环境 ddl-auto=update、开放 SQL 日志；prod 环境 ddl-auto=validate、关闭 show-sql。

任务二：持久化建模

为以下领域对象添加 JPA 注解，确保字段映射、约束与索引符合业务需求。

实体	表名	关键要求
Course	courses	课程代码唯一；包含容量、已选人数、授课教师和排课信息；自动维护创建时间
Student	students	学号与邮箱唯一；记录专业、年级等字段；自动维护创建时间
Enrollment	enrollments	课程与学生双重唯一约束；包含选课时间与状态枚举；支持查询索引

- 使用嵌入式对象表示讲师信息、排课信息等复合字段。
- 使用枚举类型表示选课状态（如 ACTIVE、DROPPED、COMPLETED）。
- 通过 @PrePersist 等生命周期回调填充时间戳与默认值。

任务三：Repository 设计

为每个实体定义 JpaRepository 接口，并实现常用查询方法：

- CourseRepository
  - 按课程代码、讲师编号查询
  - 统计或筛选有剩余容量的课程
  - 支持标题关键字模糊查询
- StudentRepository
  - 按学号、邮箱唯一查询
  - 判重检查（existsBy...）
  - 按专业或年级筛选

- `EnrollmentRepository`
  - 按课程、学生、状态组合查询
  - 统计课程活跃人数
  - 判断学生是否已选课

方法命名请遵循 Spring Data 规范，必要时使用 `@Query` 编写 JPQL。

任务四：业务层重构

- Service 层改为依赖 Repository，实现 CRUD 与校验逻辑。
- 使用 `@Transactional` 确保选课、退课等操作的一致性。
- 保留原有业务规则：课程容量限制、学号/邮箱唯一性、删除前的关联检查等。
- 删除内存集合或模拟数据，改为数据库读写。
- 优化异常处理：不存在的资源返回 404，业务冲突返回 409/400。

任务五：数据初始化与迁移

- 在 `src/main/resources/db/`（或自行约定路径）准备初始化脚本：
  - 建表语句或 `schema.sql`
  - 基础测试数据（课程、学生、选课）或 `data.sql`
- 根据环境选择是否自动运行脚本（H2 可自动执行，MySQL 可手动导入）。
- 记录初始化步骤与示例命令，方便批量重建。
- 移除课堂示例中的 `@PostConstruct`/内存初始化逻辑，确保所有基线数据由脚本或迁移工具提供。

任务六：配置与部署验证

- H2 开发模式：确认 H2 控制台可访问，数据持久化至内存。
- MySQL 生产模式：配置连接池（如 HikariCP），验证应用启动日志无错误。
- 实现 `/health/db` 或等价接口/脚本，用于快速验证数据库连通性，并在作业中附上调用示例（如 `curl localhost:8080/health/db` 输出）。
- 在 `README` 或 `docs/` 中补充数据库配置说明、环境切换方式。

测试与验收

功能验证

- 启动应用（dev 与 prod 两种配置），确认数据库连接成功。
- 使用 API/集成测试验证：
  - 新增、查询、更新、删除课程/学生均能持久化
  - 选课行为（选课、退课、容量校验）与数据库同步
  - 重启应用后数据仍然存在
- 运行已有 `mvn test`（或新增的 Repository/Service 测试）确保通过。

数据验证

- 在数据库客户端查询表结构与索引，确认符合设计。
- 使用 SELECT 语句验证初始化数据与业务操作结果。
- 若使用多环境配置，演示环境切换脚本或命令。
- 调用数据库健康检查接口或脚本，展示成功与失败两种返回结果的处理方式。

提交材料

- 更新后的源码（实体、Repository、Service、配置文件等）
- 初始化脚本与使用说明
- 测试文档或截图（包含数据库验证、API 调用结果）
- Git tag：`coursehub-week-05`

提交规范

- 提交信息遵循 Conventional Commits（例如 `feat(persistence): add jpa repositories`）。
- 推送前执行 `mvn -pl services/catalog-service -am test` 或等价命令。
- 在作业报告中总结遇到的问题、解决方案以及数据库迁移的后续计划。