工程应用开发

Project2 Review

1. 不要直接在代码中直接使用sql语句，可维护性不太好，而且容易被sql注入。

2. 代码参考：https://github.com/9cc9/ai_course

面向对象编程–OOP三大特性

封装（Encapsulation）：将对象的状态（数据）和行为（方法）封装在一起，对外部通过公共方法进行访问和修改。可以隐藏实现细节，提高代码的安全性和可维护性。

继承（Inheritance）：子类继承父类的属性和方法，复用代码，并能扩展或修改父类的功能。继承使得代码更加模块化、可扩展。

多态（Polymorphism）：允许不同的类使用相同的方法名，不同的对象可以表现出不同的行为。多态提高了代码的灵活性和可扩展性。

- 方法重载：同一个方法名，根据参数类型不同，调用不同的方法。
- 方法重写：通过继承和方法重写（override）实现，子类可以重新定义父类的方法，从而表现出不同的行为。

面向对象编程 OOP

应该把所有的业务逻辑都放到对象里吗？

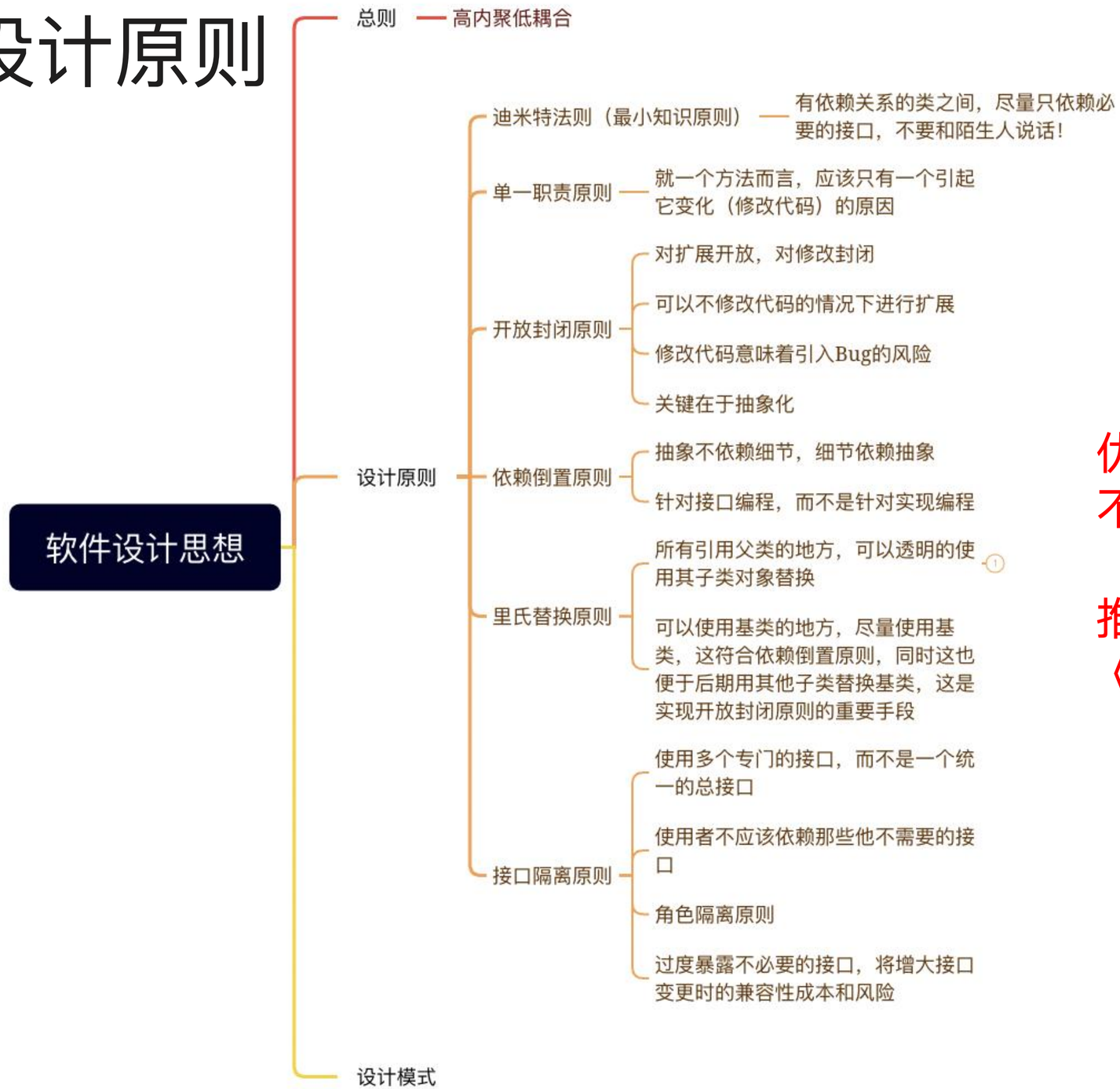
领域驱动设计（DDD）衍生出两种实现模式

1. 贫血模型 + 服务（User + UserService）=>userService.save

2. 充血模型 + 模型内部行为（User）=>user.save

=>混合使用，根据实际使用决策

软件设计原则



优先面向接口编程，而不是面向实现

推荐：Martin Fowler《重构》

Spring Boot 简介

Spring Boot 是一个基于 Spring 框架的快速开发框架，简化了 Spring 应用的开发和部署。可以实现快速的web/core系统(微服务)开发及部署。

1 快速开发

Spring Boot 提供了自动配置和约定优于配置，减少了大量的配置和代码编写。

2 默认配置

Spring Boot 尽可能地简化配置，只需很少的配置就能启动和运行一个 Web 应用。

3 嵌入式服务器支持

Spring Boot 内置了 Tomcat、Jetty、Undertow 等嵌入式服务器，方便部署和运行应用。

官网文档

中文：
https://springdoc.cn/spring-boot/

英文：
https://spring.io/quickstart

新建 Spring Boot 工程

创建一个新的 Spring Boot 项目，选择所需依赖库和版本。

1. **创建项目**
   使用 Spring Initializr 网站或者直接命令行创建项目，选择 Maven 或 Gradle 构建工具、Spring Boot 版本以及所需的依赖库。
2. **选择依赖**
   在pom文件中补充需要的依赖jar包/starter。
3. **导入项目**
   将生成的项目导入到您的 IDE 中，例如 IntelliJ IDEA/vscode。

安装 maven作为包管理器

新建工程命令行：

```
mvn archetype:generate -
DgroupId=org.uestc.weglas -
DartifactId=ai_manager -
Dversion=1.0-SNAPSHOT -
DarchetypeArtifactId=maven-
archetype-quickstart -
DinteractiveMode=false
```

设计思想：约定优于配置

Spring Boot 遵循“约定优于配置”的设计理念，提供自动配置和默认行为。

核心思想：如果开发者没有显式指定某些设置或配置，框架会提供合理的默认值和默认行为。
开发者只需关注自己特有的需求，减少不必要的配置。

提供实现

默认约定

Spring Boot 提供了一些默认约定，例如使用嵌入式 Tomcat 服务器，默认端口为 8080。

自动配置

Spring Boot 会自动配置应用程序所需的基本功能，例如嵌入式服务器、数据源配置等。

自定义配置

可以通过配置文件 application.properties 或 application.yml 自定义配置，覆盖默认设置。

Spring Boot 工程目录结构

Spring Boot 工程目录结构包含主代码、资源文件、配置文件和测试代码。

src/main/java

主代码目录，包含应用程序的主要逻辑。

src/main/resources

资源文件目录，包含配置文件、静态文件等。

src/test/java

测试代码目录，包含应用程序的测试代码。

Spring Boot 工程目录结构

1. 目录结构
    - src/main/java
        - controller：web层控制器文件
        - biz: 业务层
            - manager:业务层服务，组装领域服务
            - dto：业务对象
        - core：领域层
            - service: 领域层服务，对核心服务能力的抽象
            - model：领域模型
        - base：基础层
            - mapper: ibatis数据表映射文件
            - entity：数据层模型
        - util：工具类（日志、错误码、异常等）
        - AiManageApplication.java :应用启动文件
    - src/main/resources
        - db：sql目录
        - application.yml 配置文件
        - log4j2-spring.xml 日志配置文件，log4j2
        - mappers:ibatis数据表映射文件
        - static:静态文件，js、css、logo等
        - templates：项目自己实现的前端页面
    - src/test/java 测试用例

后端调用链路：

https请求
->controller
->->biz#manager(根据需求复杂程度，可选)
->->->core#service
->->->->base#mapper
->controller#构造https响应

前端方案选型

Thymeleaf：服务器端渲染（SSR）。Thymeleaf 是一个模板引擎，通常与 Spring Boot 一起使用，它将模板渲染为 HTML 并在服务器端生成最终页面返回给浏览器。适用于传统的 Web 应用，尤其是在不需要太多动态交互的情况下。

React：客户端渲染（CSR）。React 是一个 JavaScript 库，适用于构建单页应用（SPA），客户端渲染整个页面结构。它通过与 API 后端交互来更新页面内容，通常依赖于前端框架和状态管理。

前端方案选型

对比维度	Thymeleaf	React
渲染方式	服务器端渲染（SSR），由后端生成 HTML 页面	客户端渲染（CSR），由浏览器通过 JavaScript 渲染
适用场景	传统的 Web 应用，页面更新较少，交互简单	单页应用（SPA），需要动态交互或频繁更新的应用
用户交互	适用于表单提交、页面跳转等少量交互	适用于高频、实时的用户交互，如聊天、动态更新等
开发模式	后端渲染，直接返回视图，通常与 Spring Boot 配合使用	前后端分离，使用 React 框架进行组件化开发
性能	页面每次请求都需要重新渲染，适合页面较少变动	使用虚拟 DOM，适合高频交互和动态内容的渲染
维护性	渲染和逻辑紧密耦合，修改界面时需要改动后端代码	前后端分离，独立维护前端和后端代码
适用项目类型	企业管理系统、内容管理系统（CMS）、后台管理系统等	社交平台、电子商务网站、实时数据展示等
开发效率	开发速度较快，尤其适合已有后端架构的应用	初期开发可能较慢，但前后端解耦后更易扩展和维护
灵活性	相对有限，主要依赖后端控制渲染逻辑	高度灵活，前后端解耦，支持丰富的前端功能和 UI

Spring Boot 核心组件

Spring Boot 包含了多个核心组件，共同构建了一个强大的开发框架。

组件	功能
Spring Boot Starter	简化依赖管理，提供常用功能的 starter 包。
Spring Boot Autoconfiguration	自动配置机制的实现原理及应用。
Spring Boot Actuator	提供健康检查、性能指标、请求跟踪等生产环境所需的功能。
Spring Boot DevTools	提供自动重启、快速调试等开发工具功能。
嵌入式服务器	支持嵌入式服务器，简化应用部署。

Spring Boot 的常见功能与扩展

Spring Boot 提供了丰富的功能，并可以扩展以满足各种开发需求。

mybatis

数据库表结构设计及ORM框架

Spring Security

提供用户认证、权限管理以及 Web 应用的防护机制，保护应用程序的安全。

RESTful API

创建和配置 RESTful API 服务，实现前后端分离。

WebSocket

实现实时通信，例如聊天应用和数据推送功能。

Project3--一个简单的会话管理页面

需求说明：实现聊天会话的生命周期管理（增删查改）

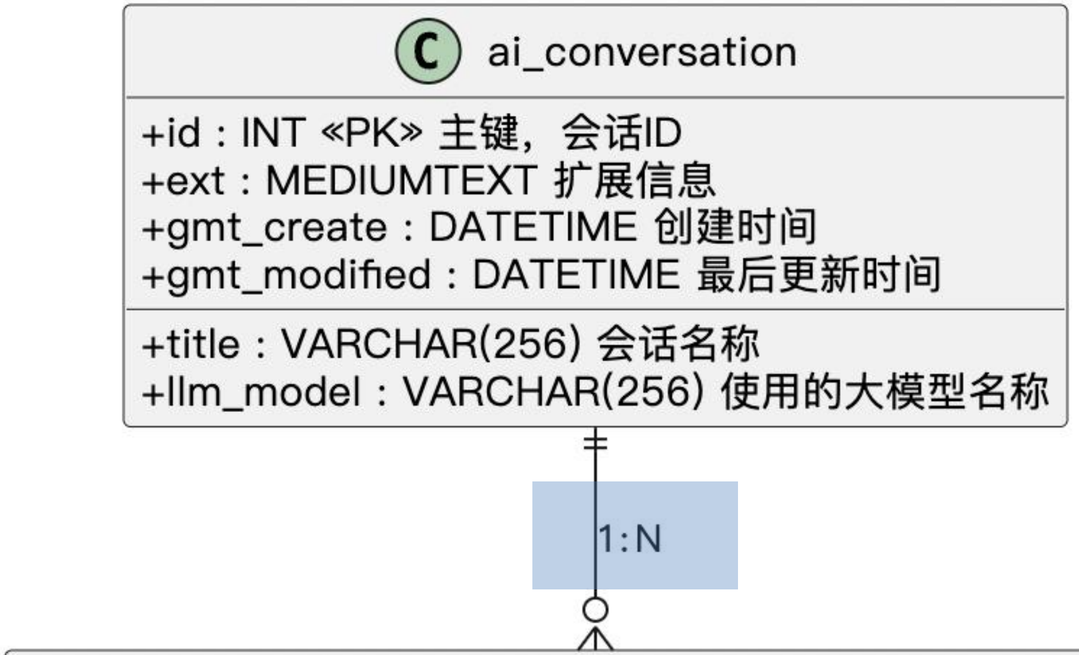
1. 创建会话：用户可以发起新的聊天会话。

2. 聊天：支持在当前会话中进行聊天。

3. 历史记录查询：支持用户查看历史会话记录列表。

4. 数据持久化：会话和消息记录存储在数据库中，关联会话ID，支持后续查询和管理。

Project3--一个简单的会话管理页面-模型设计



会话信息表

聊天记录表

Project3-本周需要实现的接口列表

1. （已实现）新建会话及第一条聊天记录 URL: /conversations/add.json
功能: 创建一个新会话并添加第一条聊天记录。

2. 获取会话列表 URL:/conversations/list.json
功能: 返回所有会话的列表信息。

3. 新增一条聊天记录 URL: /conversations/addChat.json
功能: 向指定会话中添加一条新的聊天记录。

4.（已实现）首页http://127.0.0.1:8081/

Project3-准备工作

demo地址： https://github.com/9cc9/ai_manager_course1

1. 安装java、maven、mysql server
2. 安装依赖的jar包　mvn clean install
3. 参考application.yml文件，按需修改db连接配置（用户名、密码、数据库名）
4. 启动应用

   java -jar target/ai-manager.jar

   启动过程中会自动执行schema.sql,新建数据表
5. 浏览器访问 http://127.0.0.1:8081/

Project3-目标方案（含下周作业）