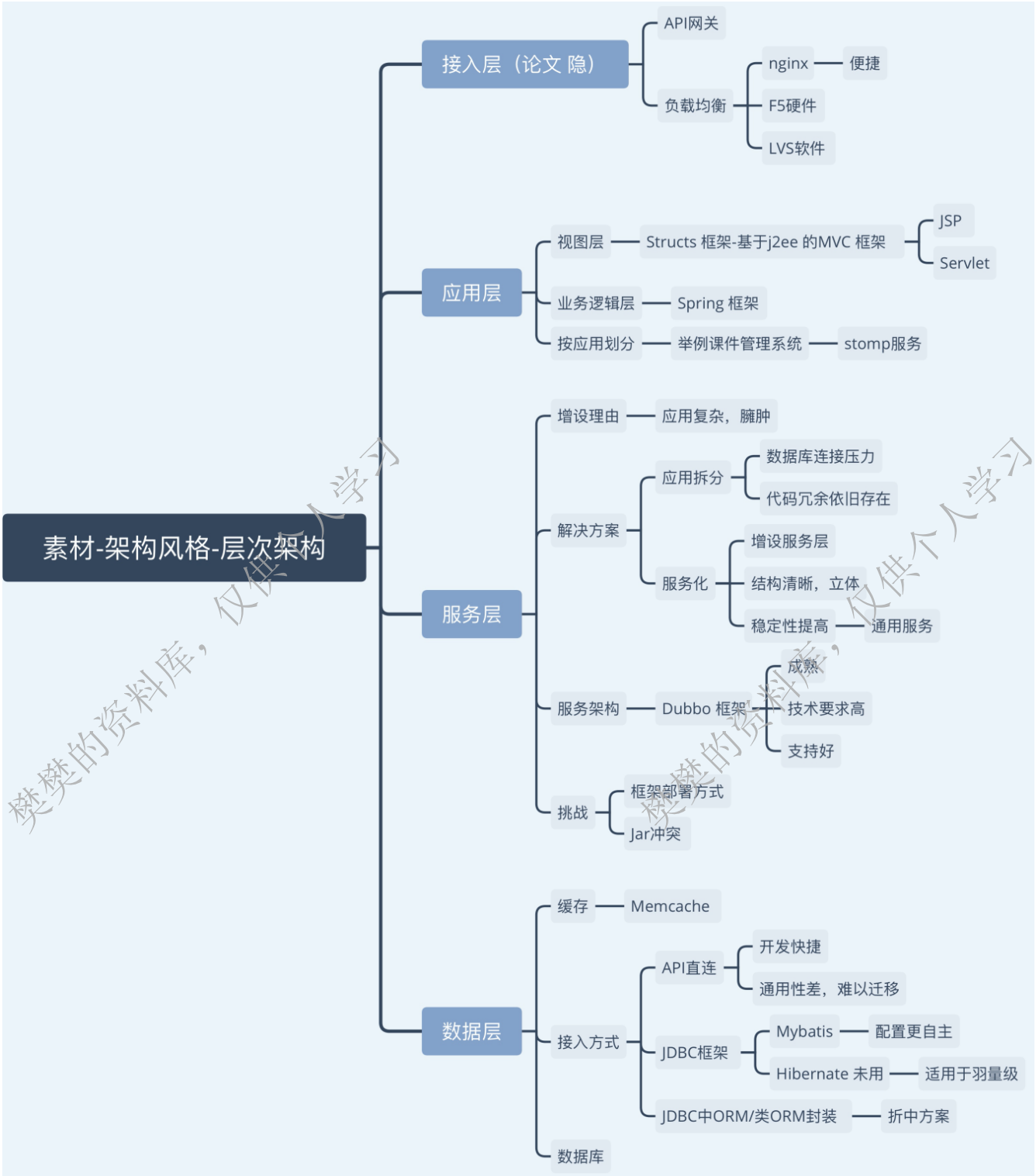
一，理论：



二，论文：

摘要：

本人于2015年11月参与浙江省某在线教育平台“外教一对一在线教育”项目，该项目为客户提供了一对一欧美外教视频教学，社交圈，公众直播等功能提供全方位的软件支撑，在该项目组中我担任系统架构师岗位，主要负责整体架构设计与中间件选型。本文以该教育平台为例，主要讨论了软件架构风格在该项目中的具体应用。整个系统采用具有三层的层次式软件架构的设计思想，分别是应用层，服务层，数据层。在应用层中的业务逻辑层的设计中，将整个业务系统划分为十余个子系统。服务层以 dubbo 服务框架为核心，数据采用了 Mybatis 框架。整个系统开发工作历时18个月。目前，该系统已经稳定运行近一年半的时间。实践证明，这种架构设计有效地降低了维护成本，提高了系统的开放性，可拓展性，可复用性和可移植性。

正文：

随着国家对教育对越发重视，英语教育的市场份额逐步上升，其中用户口语提升的需求越来越大。为此，一些公司开始提供与外国人聊天的平台。我公司决定从国际通讯领域进军口语教育领域。为了这项战略转变，公司于2015年11月设计在线教育平台系统（以下简称为“系统“）。该系统帮助人们与欧美外教进行面对面的口语交流与教学。其中随意聊提供了一个类似 QQ 视频通话，而正式课程还提供了 H5互动课件以提高教学质量。与此同时，还有公众直播用于拉新，AI 测试用于评测学员能力，降低成本。我参与了该项目的开发工作，担任系统架构设计师职务，主要负责设计系统架构。本项目组全体成员共9人，我主要负责项目计划制定，需求分析，整体架构设计与技术选型，以及底层设计。该项目的架构工作于次年2月完成，整个项目耗时18个月，于2017年5月完成。

在架构工作的开始阶段，我们便意识到，架构风格定义了用于描述系统的术语表和一组指导构建系统的规则，是系统组织方式的惯用模式，可以为我们的项目提供架构级的通用解决方案。这种架构级的软件重用可以极大提高我们的系统建设进程。

软件系统开发中常用的软件架构风格有数据流风格，调用/返回风格，独立构件风格，虚拟机风格，仓库风格。数据流风格包括批处理序列与管道-过滤器，其每一步处理都是独立，顺序执行的，适用于简单的线性流程。调用/返回风格包括主程序/子程序风格，面向对象风格，层次结构风格，其进一步降低系统耦合度，明确系统层次。独立构件风格包括进程通信，事件驱动系统（隐式调用），其构件特点为软件重用提供了支持。虚拟机风格包括解释器风格，基于规则的系统风格，其具有良好灵活性，可自定义规则。仓库风格包括数据库系统风格，超文本系统风格，黑板系统风格，其以数据为中心。除此之外，还有 dssa，soa 等架构风格。

在了解需求后，我们决定在公司技术顾问的建议下，采用层次架构风格。为了解决公司原通讯系统代码冗余问题，我们进行了系统服务化，中间层不同的服务中心提供不同的业务服务。最终，我们将系统分为应用层，服务层，数据层。应用层负责具体业务和视图展示，如网站首页，app 内搜索输入与结果展示等，其又分为视图层与业务逻辑层。服务层负责为应用层提供通用服务支持，如账户管理服务，session 管理服务等，其又细分为逻辑处理层与数据接口层。而数据层负责提供数据存储访问服务，如数据库服务，缓存服务，文件服务，搜索服务等。接下来，我将分层次详细介绍三层层次体系结构的设计过程。

首先是应用层。在应用层中，我们将系统根据应用进行水平划分，这有助于代码管理与维护。我们将系统分为课件管理系统，公众直播系统，学员测试系统，课程管理系统等十余个子系统，这里以课件管理系统为例。课件管理系统负责学员上课所用的课件，有课件编辑，课件预览，课件交互，课件展示等多个功能模块。功能模块调用服务层的服务支撑，如课件交互需要调用服务层由 ActiveMQ 实现的 stomp 通信服务，通过该通信服务，实现教师端与学生端的课件的同步，从而使得课件交互变得有趣，生动，具有互动性。另一方面为了区别教师端与学生端的交互权限，课件模块还需要调用服务层的账户服务，确定用户身份，从而明确用户在课件交互中的交互权限。与此同时，为提高课件的可修改性，可互动性等，课件采用 H5编

写。应用层主要采用 structs 这一基于 J2EE 平台的 MVC 框架，主要通过 Servlet 与 JSP 技术实现。另外还有动静分离，动态资源静态化等，这里不再赘述。

其次是服务层。服务层采用了 dubbo 服务框架等技术实现。随着服务器规模的扩大，开发人员的增多，每个应用都变得复杂，臃肿，存在大量代码重复。为解决这个问题，提出了两个方案。一个是将应用拆分得更小，确保每个应用都保持在一个合适的大小。好处是设计能够较快地完成，代码也较容易实现。这个方案存在一些问题，一方面数据库的连接数压力依旧存在，另一方面，代码的冗余依旧存在。所以我们采用了第二个方案-服务化方案。服务化方案，即应用层和数据层中增加一个服务层。首先从结构上来看，系统结构更为清晰明了，更为立体。稳定性上来看，许多散落的代码成为了通用服务，并交付专门的团队负责运维。出于对成本与技术成熟度的考虑，我们采用了阿里研发的 dubbo 服务框架。在服务框架实际操作时有两个问题：服务框架自身的部署方式问题与实现服务框架所依赖的一些外部 jar 包与应用自身依赖的 jar 包之间的冲突。前者，我们通过 Tomcat 作为 Web 容器，而服务框架作为容器的一部分。后者，我们通过 Java 的 ClassLoader 将服务框架自身用的类与应用用的类进行隔离。除此之外，还有服务线程池隔离，分布请求合并，服务调用端的流控处理，异步服务调用，通过 Future 方式对远程服务调用的优化等问题，限于篇幅，不再赘述。

最后是数据层。数据层涉及缓存，文件系统，数据库，数据通知服务，搜索系统等模块。由于用户对数据的访问具有集中性，所以我们基于 SpringCache 与 Redis 实现了缓存机制。由于系统的业务特性，数据库往往是读操作远多于写操作，所以我们对数据库进行了读写分离。数据访问方面，Java 已经有很多成熟技术，大致分为三种。第一种是为用户提供专有 API，这种方法便于实现功能，但是通用性较差。第二种是通过 JDBC 方式访问数据库，数据层本身作为一个 JDBC 的实现，也就是暴露出 JDBC 的接口给服务层。该方法成本很低，迁移成本也非常低，但开发成本相对高一些。第三种方式是基于 ORM 或类 ORM 接口的方式。我们采用的就是这种方

式，使用数据库时使用 ORM 框架-Mybatis 框架，再将框架包装一层，用于实现数据层功能，对外暴露的仍然是 Mybatis 的接口。该方法开发高效，敏捷，成本较低，而且兼容性不错。此外，我们采用 solr 作为数据层搜索引擎，数据访问层物理部署采用 Proxy 方式等。限于篇幅，不在赘述。

最终项目成功上线，正常运行了近一年半，收到各方好评。尤其是 H5课件的良好互动性，使得大量业界同行争相模仿，改用 H5制作课件。还有我们的服务化方案架构被作为许多传统互联网企业系统重构的经典方案。在系统的架构设计中，我们引入了层次架构的设计思想，有效地降低了维护成本，提高了系统的开放性，可扩展性，可重用性以及可移植性。当然还是存在一些问题的。如 H5课件采用 http 协议，易被非法劫持，嵌入广告，可以将协议修改为 https 来解决。还有我们采用的负载均衡算法是加权轮转算法，过于简单，常常出现资源分配不合理的现象，可以将算法改为加权最小连接数算法来解决。这些都是我在今后的系统设计和开发中需要注意与改进的地方，也是日后我应该努力的方向