毕业设计开题报告

林子牛

导师: 甘永梅

目录

1 题目选定

本次毕业设计的题目为: **基于微信小程序及 SpringBoot 的短视频应用开发**。近年来随着诸如 4G、高速宽带网络以及智能手机技术的飞速进步。人们的休闲娱乐重心逐渐从文字与图片相关内容转移到了短视频与直播应用上。市场上也涌现出了一系列的短视频应用软件。本毕业设计课题将完成短视频应用开发的整个流程。

目前短视频应用已经成为互联网产业链中一个重要的流量入口。著名短视频应用抖音在各大手机应用市场中获得了累计 74.61 亿次下载，在 App Store 摄影与录像应用分类中排行第一。截止 2018 年 12 月，在线视频行业月独立设备数达 10.17 亿，同比增长 1.7%，其中短视频行业月独立设备数达 7.34 亿台，同比增长率为 58.7%[1]。在智能手机已经普及的今天，短视频相关应用已经爆发出了强大的活力，具有相当高的市场价值。所以进行短视频应用开发对于我们认识短时频应用相关技术以及商业模式具有非常大的帮助作用。

本设计中，短视频应用软件主要分为两个部分: 前端部分与后端部分。前端部分主要负责与用户的交互。后端部分负责应用程序逻辑功能的实现。本设计要求前端使用微信小程序实现，后端搭建于 SpringBoot 技术。

2 文献综述

本设计使用 Java 语言以及 SpringBoot 框架开发应用后端，使用以微信小程序为代表的混合应用开发框架构建应用前端。

为了研究服务端应用开发演进过程以及其中每个阶段的突出问题与后续解决以方便本次的设计开发，我研读了十几篇相关论文与著作，了解了 J2EE 开发的各个阶段以及开发流程、Spring 框架实现原理与开发过程、微信小程序的开发过程。

2.1 Java Web 开发的发展

Java 服务端开发可分为如下几个阶段:

- applet 阶段: 此阶段严格意义上说不在 J2EE 发展阶段之内。applet 即运行在网络浏览器中的 Java。由于 applet 可能具有的安全漏洞以及使用起来过于麻烦，现在几乎不再将 applet 用于 web 开发之中[2]。

- Servlet 阶段: Servlet 就是一个运行在 J2EE 容器中的 Java 程序，Servlet 接收前端发送过来的请求，待其处理完成后，返回一个由 HTML 组成的响应[3]。它的缺点是只能在 Servlet 内部拼接 HTML，非常不方便。

- Servlet 和 Jsp 阶段: 为了克服 Servlet 的缺点，开发者们引入了 Jsp 技术，即把 Java 代码嵌入 HTML 中[3]，用户使用时由 J2EE 容器解析 Jsp，大大提高了开发效率。但

仍存在前端与后端开发混乱的问题。

- MVC 阶段: MVC[1] 将应用整体分为三个部分: 模型，视图与控制。模型负责处理业务逻辑，视图负责将结果展现给用户，控制负责接收并处理请求，是模型与视图之间的桥梁。MVC 结构将前端与后端分离开来，减少了系统的耦合性[4]。

2.2 Java Web 开发的框架阶段

在 J2EE 开发进入 MVC 阶段后，web 开发又产生了许多新的需求，如: 提高开发效率、增强系统稳定性、进一步降低耦合性以及实现分布式开发等。为此，产生了 SUN 公司大力支持的 EJB[2] 和开源社区驱动的诸多开发框架。

- EJB: EJB 将服务端组件与分布式对象技术、Web Service、进程间异步通信以及对象的持久化管理结合了起来[5]。EJB 提高了企业级应用的开发效率，为实现应用的分布式提供了方便的环境并且极大地改变了人们对于企业及应用的思维方式。然而，EJB 也有许多的缺点，如: 框架体量过重，在编写比较小的服务时，开发的复杂性与业务逻辑的复杂性不成正比以及 EJB 中一些设计已经有些过时等。EJB 并没有很好的降低开发地复杂度并且 EJB 中的持久化方案 Entity Beans 作为一种持久化方式是比较失败的[6]。

- Spring Framework: 为了解决上述 EJB 的一些缺陷，开发者们提出了一些新的技术，如：控制反转[3]、面向切面编程[4]等。并且提出了一种轻量化的新型架构来取代 EJB。Spring 框架是一个轻量级开源框架，它通过依赖注入[5]来实现控制反转，并且使用面向切面编程来解决企业级应用中经常遇见的问题。Spring 框架实现了轻量级编程、系统的松耦合减少了样板式代码提高了开发效率[7]，因此此次设计采用 Spring 框架进行开发。

2.3 Spring 框架简介

Spring 向用户提供的功能都依赖于它的两个核心特性—控制反转容器与面向切面编程框架。

---

[1]Model, View, Controller

[2]Enterprise JavaBeans

[3]Inversion of Controll(IoC)

[4]Aspect Oriented Programing(AOP)

[5]Dependency Injection

2.3.1 Spring 总览

Spring 框架倡导基于 POJO[6] 进行开发，并且提供了许多基于 POJO 构建应用的工具，极大简化了开发工作。Spring 框架的基础是核心模块，其中包含了 Core、IoC 容器以及 Spring 框架基础工具类。核心模块之上是 Spring 框架提供的一些功能，如：AOP 支持、ORM[7]、JavaEE 模块的集成服务以及 MVC 支持等[8]。

2.3.2 控制反转容器

控制反转来自于设计模式中的依赖倒置原则。所谓依赖倒置原则即一种特定的解耦模式，它使高层级的模块不直接依赖于低层级的模块，而是两者同时依赖于某个接口[9]。

依赖注入是控制反转的一种实现方式，Spring 框架使用依赖注入来实现控制反转。Spring IoC 容器就是对依赖注入的具体实现。当应用中需要使用一个对象的实例时不需要该应用自己创建该实例，而是向 IoC 容器申请一个实例。实例的创建以及生命周期均由容器进行控制。此时，高层次模块与低层次模块均依赖于抽象，实现了模块之间的低耦合[10]。在 Spring 框架中，用户首先在配置文件[8]中将需要使用到的对象注册为 bean，随后 IoC 容器便可以自动将 bean 注入到使用实例的地方。

2.3.3 Spring AOP 框架

AOP 即面向切面编程是对面向对象编程的扩展。应用系统中有着许多的横切关注点，如应用某种情况下需要进行日志记录且系统中所有的模块都必须记录，在这种情况下日志记录点就是一个横切关注点。要对横切关注点进行操作时 AOP 模块会将相应的功能织入到 OOP 的功能模块中。由此我们可以通过 AOP 来对横切关注点进行组织，降低系统的耦合性以及提高系统的可维护性。

Spring AOP 是 Spring 框架的重要组成部分。Spring AOP 并没有将所有的 AOP 功能都进行实现而是只使用 AOP 20% 的支持实现 80% 的功能[8]。Spring AOP 的实现基于设计模式中的代理模式即访问者与被访问者不直接进行通信而是通过一个代理对象进行通信[9]。这样有助于方便对象之间通信以及有助于数据安全的作用。Spring 框架就是通过使用动态代理机制在运行时生成代理接口来实现 AOP。

Spring AOP 最为广泛的用途是管理日志操作和数据库事务管理。一个系统中通常有许多模块，没个模块都需要在某些位置进行日志记录，如果不使用 AOP ，则需在每个日志记录位置都放置同样功能的模块，代码利用率低修改复杂。引入 AOP 技术之后，便可以将其作为横切关注点，通过 AOP 进行统一操作，效率高。数据库事务的声明与运行结果的判定同样是一些固定操作，也可以用 AOP 来进行组织以降低系统耦合性、提高系统可维护性。

[6]Plain Old Java Object

[7]Object Relation Mapping 对象关系映射

[8]xml 配置或 Java 注解

2.3.4 Spring 数据访问

应用通常都会涉及到对于数据的访问，为了统一访问数据的方式，J2EE 制定了 DAO[9] 模式，通过在系统中设立数据访问层来讲数据的读取从应用中独立出来。通过 DAO 结合各种数据库的驱动在一定程度上实现了数据的逻辑独立性[11]。此外 Spring 还对 JDBC 以及各种 ORM[10] 框架提供了支持。

通常我们会在 Spring 应用中将 Hibernate 或者 MyBatis 作为 ORM 框架。ORM 框架最基本的作用就是将 Java Object 与 Database Relation 进行映射，方便数据的访问。SpringBoot 出现后，Spring 开源社区更是基于 Hibernate 实现了 Java 的 JPA 规范，供用户直接使用。此外 Spring 还提供了诸如声明式事务管理等一系列功能，方便用户使用 ORM 框架。

2.3.5 MVC 支持

Spring 应用中可以使用许多 MVC 框架如: Spring MVC、structs 等。由于 Spring MVC 与 Spring 框架都是由 Pivotal 开发，所以 Spring MVC 天然地支持 Spring 框架的所有功能，而想使用其他框架则需相应的中间层支持。Spring MVC 具有良好的开放性，从表现层来看，Spring MVC 通过视图解析器寻找并渲染相应的视图，其中视图可以是 JSP、FreeMarker 这样的模板引擎甚至是 PDF 这样的文件。Spring MVC 还具有非常强大的扩展功能，可以轻松实现其他的功能[4]。

2.4 SpringBoot 简介

Spring 框架为我们构建 Web 应用提供了想对简单的方法，然而 Spring 框架本身是轻量级的，它的配置方式确实重量级的。Spring 默认通过 XML 文件进行配置，如配置 IoC 容器，MVC 中配置控制器等。Spring 框架资深也通过支持 Java 注解来消除 XML 配置。但使用某些功能或使用第三方库时仍需使用 XML 进行配置。

为了解决 XML 配置繁杂的问题，Spring 开源社区引入了 SpringBoot 来帮助我们减少配置。SpringBoot 是一个约定大于配置的 Spring 的集成框架。SpringBoot 为我们的开发工作带来了便利，这些便利集中体现在:

- 自动配置功能: SpringBoot 可以对一些常见库进行自动化的配置，无需用户手动操作。
- starter: SpringBoot 提供了许多 starter，只要告诉 SpringBoot 需要还送了个么样的 starter，SpringBoot 就可以自动将它们引入

引入了 SpringBoot 后，我们就可以再也不用配置 XML，配置数据库事物时也可以直接使用注解来创建声明式事物。

[9]Data Access Object

[10]Object Relation Mapping

2.5 数据库简介

在本次设计的应用中，数据库也是非常重要的一部分。数据库是永久储存的、有组织的大量数据的集合，这些数据按照某种模型妥善的加以组织和储存，具有数据冗余小、数据独立性高、易共享、易扩充的特性[12]。

现代数据库所使用的数据模型一般为关系数据模型。关系数据模型非常简单，就是一张表。关系数据模型具有以下优点:

- 关系模型基于严格的数学概念
- 关系模型概念单一，简单易懂
- 具有较高的数据独立性[12]

数据库的开发首先要进行数据库的设计，待设计完成并且通过验证后即可开始运行。常用的数据库设计步骤如下:

- 需求分析
- 概念结构设计
- 逻辑结构设计
- 物理结构设计
- 数据库实施
- 数据库运行与维护

2.6 微信小程序简介

微信小程序开发是一种混合应用[11]开发模式，即 HTML5 与 Native 应用混合。随着 HTML5 的出现，单纯 HTML 可以实现的功能越来越多，并且诸多 Native 应用中都有着 WebView 的功能。因此一个移动应用的主要逻辑部分可以用 HTML5 来实现，在手机上使用 WebView 来显示。此类型应用被称为混合应用。混合应用具有以下优点:

- 混合应用底层由相关容器提供，逻辑部分由 HTML5 页面完成，开发效率高
- 或者应用模式 HTML5 代码可以在不同的平台上复用，可移植性好。

[11]Hybrid APP

2.7 应用设计经验

为了提高应用开发效率与可维护性、降低系统耦合度，在开发 Spring 应用时通常会遵循一系列原则，采用一定的架构。目前普遍使用的是分层次架构，即将整个应用分为表现层、控制器层、服务层、数据访问层以及数据库[13]。

- 表现层: 负责将控制器层返回的处理结果渲染给用户，一般采用 JSP 或某种模板引擎或直接发送 JSON 给前端进行渲染。
- 控制器层: 负责处理请求，调用服务层提供的函数与方法并将结果返回给表现层。
- 服务层: 负责向控制器层提供服务，本身是对数据访问层的封装以及对获取数据的处理。
- 数据访问层: 直接与数据库进行交流，直接负责数据的存取，并向服务层提供封装好的数据存取函数、方法，提供一定的数据独立性。
- 数据库: 负责储存数据，一般有关系数据管理系统来担任。

3 设计方案

本设计目标为:

- 设计并完成短视频应用后端全部功能
- 设计并完成短视频应用小程序

3.1 短视频应用后端程序设计

我采用 SpringBoot 框架构建整个应用后端，其中 MVC 框架选择 SpringMVC、ORM 框架使用 MyBatis、视频处理使用 FFmpeg。如前文所述，整个后端程序应分为表现层、控制器层、服务层、数据访问层以及数据库。

- 表现层: 在此次设计中表现成由微信小程序担任，在后端中只需定义与前端通信的标准视图对象即可，将在下文中介绍。
- 控制器层: 基于 SpringMVC 构建控制器层，直接使用声明式控制器构建方式编写。控制其中应包含拦截器与登录用户数据缓存，以防止用户非法访问和降低数据库访问压力。

- 服务层: 在后端应用设计过程中，应抽象出一部分资源的概念。资源即用户可以在服务端进行操作的一些实体。在本应用中，资源主要为用户本身、用户上传的视频等。服务层就是对资源进行操作的方法，如：添加用户、删除视频等操作。本应用使用 FFmpeg 在服务层中处理视频，并通过数据访问层实现对于各种资源的同一控制。

- 数据访问层分为两部分。第一部分为与数据库中的关系相对应的实体类，这一部分对象不应该被声明为 bean ，应该由用户自行管理，是服务层与数据库进行交流的统一单位。第二部分就是数据库中相关数据访问类，本应用中数据访问层由 MyBatis 搭建，数据访问类将数据从数据库中提取出来并放入对应的实体对象当中，供服务层使用。

- 数据库: 本应用选择 MySQL 作为主数据库，负责存储应用中一切资源的信息，如: 用户信息、视频信息等。此外，本应用还使用 Redis 作为缓存数据库，存储所有的热数据，如活跃用户信息、活跃视频信息等。缓存可以显著提高用户访问速度以及降低数据库压力。

3.2 短视频应用前端程序设计

短时频应用的前端使用微信小程序搭建，前后端之间通过使用 JSON[12] 传递的视图对象来进行通信。

- 视图对象与统一接口: 视图对象一般基于实体对象构造。视图对象可以扩充一些实体中没有的信息也可以屏蔽敏感数据，避免敏感数据传输至前端，造成用户数据泄露。前后端应用之间的通信使用统一接口以方便应用的移植以及错误处理。接口中主要包含本次传递的信息、传输状态以及传输数据量等。

- 微信小程序: 微信小程序在开发上与前端类似，其为一种 MVVM[13] 框架，并且提供了许多常用的组件。在开发时，只需一步一步开发相应的小程序页面，并与后端进行联合调试即可。

3.3 应用开发细节

本应用开发时可能会有一些性能瓶颈与安全漏洞，需要及时处理这些性能瓶颈与漏洞以提高系统效率与稳定性。

- **问题** 高并发时，若数据库事务设置不当会引起数据不一致以及数据库压力过大。

  解决方案 可以通过根据数据库事务的读写情况设置相应的隔离级别与传播行为杜绝数据不一致现象[12]。通过合理设置数据缓存以及数据库的存取结构，以加快读取速度和减小数据库系统压力。

[12] JavaScript Object Notation

[13] 数据与视图双向绑定

- **问题** 多个用户同时观看视频时，服务器压力可能过大。由于本应用中的视频属于静态资源，若由服务器直接处理会增加服务器压力，降低服务器处理其他请求时的能力。

  解决方案 采用 CDN[14] 技术，让 CDN 服务商托管所有的静态资源，来加速静态资源访问并且减小服务器压力。

- **问题** 很多用户同时观看视频时产生大量互联网连接，拥塞网络，引起访问速度下降。

  解决方案 服务端与客户端连接采用 UDP[15] 协议。UDP 协议是面向无连接的协议。发送数据时不需要进行连接，简单高效。在多个用户同时访问时可以采用组播技术。可以显著的降低网络拥塞[14]。

- **问题** 服务器可能被恶意入侵，此时用户储存的密码将被入侵者获取，造成用户数据泄露。

  解决方案 采用密码加密储存方式，即用户密码不直接储存在数据库中，而是先将用户密码添加一随机字符串，再将这一串字符进行 SHA 哈希运算，最后将结果存入数据库，密码验证时执行同一操作即可。

- **问题** 用户上传视频可能比较大，占用网络带宽、服务器储存空间较大，FFmpeg 处理视频时时间也会增加。

  解决方案 限制用户上传视频的时长并对用户视频进行压缩储存。

3.4 具体进度计划

2019 年 1 月 1 日 – 2019 年 1 月 31 日 初步拟定整个应用的架构，确定各方面使用的技术，确定使用个框架版本，学习各个框架的使用方法。

2019 年 2 月 1 日 – 2019 年 2 月 15 日 完成数据库的设计、实施、试运行，确定数据库可用。完成后端应用中与用户相关的接口[16]的实现。

2019 年 2 月 16 日 – 2019 年 2 月 28 日 完成微信小程序用户登录、注册、个人信息页面的编写并与之前完成的后端接口进行联调。后端中完成视频上传、处理、播放相关接口。

2019 年 3 月 1 日 – 2019 年 3 月 8 日 完成微信小程序前端视频上传、播放页面，并与后端接口进行联调。

[14]Content Delivery Network

[15]User Datagram Protocol

[16]用户注册、登录等相关操作接口

2019 年 3 月 9 日 – 2019 年 3 月 16 日 设计用户与用户之间的关注关系、用户与视频之间的点赞关系、用户与视频之间的评论关系。并在数据库中进行实现。

2019 年 3 月 17 日 – 2019 年 3 月 24 日 完成后端点赞、关注、评论接口，并做相应测试。

2019 年 3 月 25 日 – 2019 年 4 月 2 日 完成前端点赞、关注、评论页面，并与后端进行联调。

至此相关开发工作完成，开始测试、微调与论文撰写工作

2019 年 4 月 3 日 – 2019 年 4 月 6 日 完成测试工作，并作出相应的微调。

2019 年 4 月 7 日 – 2019 年 5 月 30 日 完成论文撰写工作，完成答辩准备。

参考文献

[1] 2018 中国互联网流量年度数据报告. [C]. [S.l.]: 上海艾瑞市场咨询有限公司, 2019.

[2] HORSTMANN C S. Java 核心技术[M]. 北京: 机械工业出版社, 2016: 598.

[3] JavaEE8 Documentation. [A]. Oracle, 2017.

[4] WARIN G. Mastering in Spring MVC 4[M]. 北京: 人民邮电出版社, 2017: 33.

[5] BURKE B, MONSON-HAEFEL R. Enterprise JavaBeans 3.0[M]. 北京: 电子工业出版社, 2007chap. 1. Introduction: 3.

[6] JOHNSON R. Expert One-on-One J2EE Development without EJB[M]. [S.l.]: Wrox, 2004: 3.

[7] WALLS C. Spring in Action[M]. 北京: 人民邮电出版社, 2016: 27.

[8] 王福强. Spring 揭秘[M]. 北京: 人民邮电出版社, 2009.

[9] GAMMA E, HELM R, JOHNSON R, et al. 设计模式：可复用面向对象软件的基础[M]. 北京: 机械工业出版社, 2007.

[10] Spring Framework Documentation. [A]. Pivotal Software, 2018.

[11] WALLS C. SpringBoot in Action[M]. 北京: 人民邮电出版社, 2016.

[12] 王珊, 萨师煊. 数据库系统概论[M]. 北京: 高等教育出版社, 2014.

[13] 杨开振. 深入浅出 SpringBoot 2.X[M]. 北京: 人民邮电出版社, 2018.

[14] 竹下隆史, 村山公保, 田幸雄, 等. 图解 TCP/IP[M]. 北京: 人民邮电出版社, 2013.