最终考核

项目开发流程

需求分析–>项目设计（功能设计，前端UI设计，数据库设计，项目总体架构设计）–>项目开发环境搭建–>项目编码开发（前端开发+后端开发）–>项目测试–>项目部署–>撰写项目文档–>项目交付完成

需求书

项目1：小型文章发布管理系统

> 项目名称：小型文章发布管理系统

需求	用户层面需求： Ø 两种用户：普通用户、系统管理员 Ø 普通用户： l 可以按照各种方式查询文章（查询标题、文章内容、作者、标签...需要支持模糊查询），按昵称查询用户 l 可以发布文章 l 可以浏览自己发布过的文章 l 可以对用户加关注 l 可以点开文章标题查看文章的详细内容，可以收藏 l 可以查看自己收藏的文章，如果可以的话能不能做个分类呢? l 可以修改自己的个人信息，个人信息需要有哪些好你寄几想吧 l 支持下载Excel文件（Excel中内容包括标题、作者、类别等） l 注册、登录属于常规操作 Ø 系统管理员 l 一样可以查询文章、查询用户 l 可以删除、修改文章 l 可以对用户进行封号处理（删除用户） l 可以统计文章的数目 l 支持上传Excel ，由Excel读取数据
	系统层面的需求： Ø 登录、注册、注销 Ø 文章信息要合理全面 Ø 用户信息要合理全面 Ø 普通用户随便注册，系统管理员由开发者指定

	技术层面要求： Ø 前端使用html+css+ajax/axios，vue/react也可以（不要求界面美观，能展示数据即可） Ø后端使用JDBC+servelet+tomcat等完成。（不可以使用spring和mybatis系列框架） Ø 使用json 作为传输媒介 Ø 连接池使用dbcp或druid（推荐druid） 注：实在弄不会ajax/axios的，前端可以使用jsp ，尽量用ajax/axios
其他要求： 1. 合理 2. 可运行 3. 代码风格简单易懂 4. 尽你所能消除bug 5. 必须有设计文件（思维导图、viso流程图、文档….形式不限，在编写代码之前） 6. 还有什么疑问可以戳我	

项目2：小型文件管理系统

项目名称：小型文件管理系统

需求	用户层面需求： Ø 两种用户：普通用户、系统管理员 Ø 普通用户： • 可以搜索文件树某个文件 • 可以上传文件 • 可以删除文件 • 可以重命名文件 • 可以将自己上传的文件以目录树结构展示 • 可以下载任意文件 • 可以预览文本文件（txt,markdown,如果能将markdown渲染出来，能够预览pdf，jpg等图片就更好了） • 可以创建文件夹 • 可以修改自己的个人信息，个人信息需要有哪些好你寄几想吧 • 注册、登录属于常规操作 • 可以修改文本文件内容 • 文件的多人协作权限管理，可以授权其他用户对自己的指定文件夹进行修改或者浏览（可选） Ø 系统管理员 • 可以查看多个用户的所有文件，具备普通用户的所有功能，但是能操作所有用户
	系统层面的需求： Ø 登录、注册、注销 Ø 文件信息要合理全面 Ø 用户信息要合理全面 Ø 普通用户随便注册，系统管理员由开发者指定

	技术层面要求： Ø 前端使用html+css+ajax/axios，vue3/react也可以（不要求界面美观，能展示数据即可）pinia,vuex,vue-router Ø后端使用JDBC+servelet+tomcat等完成。（不可以使用spring和mybatis系列框架） Ø 使用json 作为传输媒介 Ø 连接池使用dbcp或druid（推荐druid） 注：实在弄不会ajax/axios的，前端可以使用jsp ，尽量用ajax/axios

其他要求：

1. 合理
2. 可运行
3. 代码风格简单易懂
4. 尽你所能消除bug
5. 必须有设计文件（思维导图、viso流程图、文档....形式不限，在编写代码之前）
6. 还有什么疑问可以戳我

项目设计参考

数据库设计

- 表名全小写，列名全小写，多个单词以下划线分隔
- 每一张表有一个Id字段（主键），update_time以及create_time字段
- 表中单行记录不允许出现一对多关系，如果出现了一对多的需求，应该单独建立一张表存储

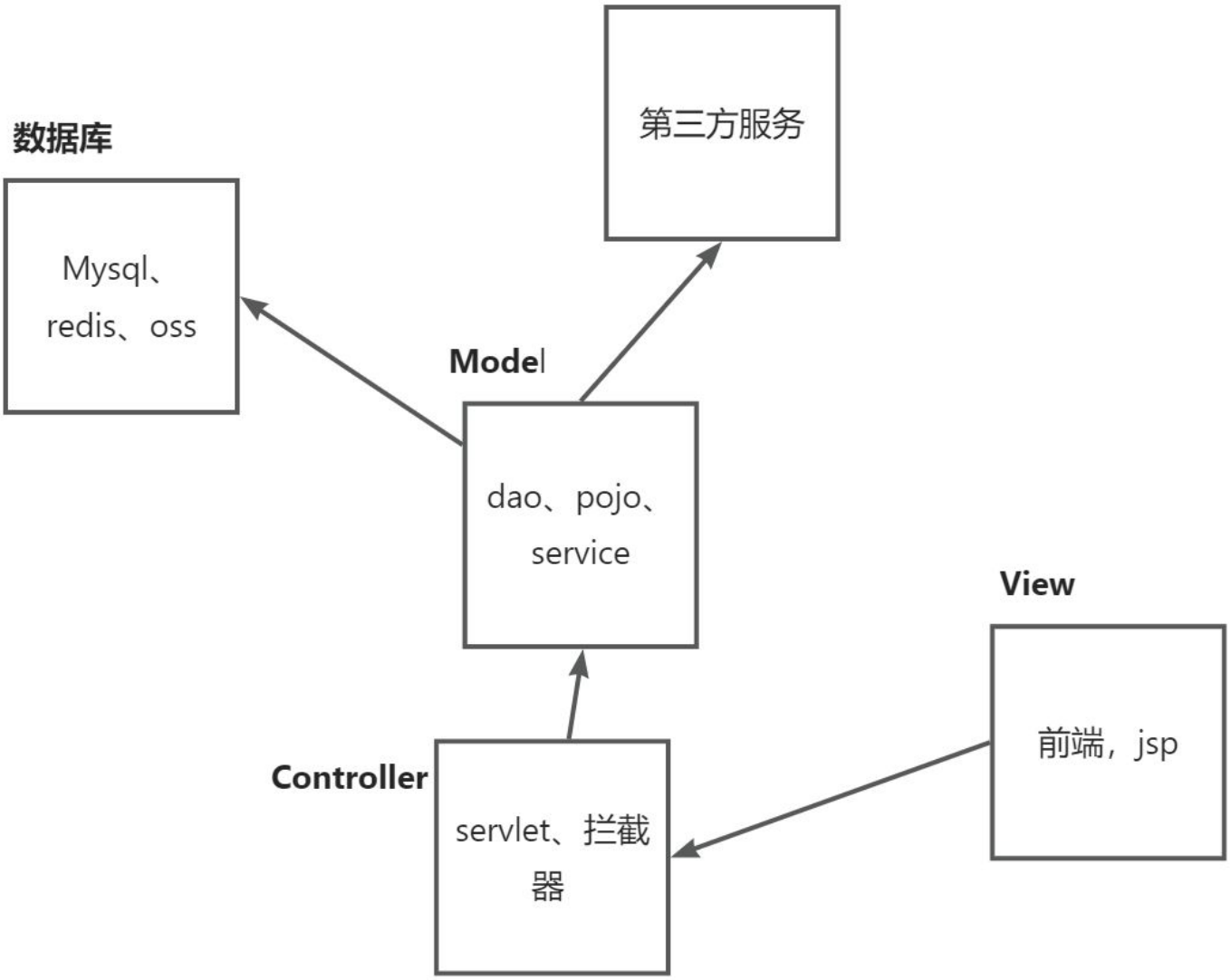
项目架构设计

1.前后端不（半）分离项目（MVC）

jsp或者模板引擎前后端一体项目，或者是前端+ajax+json的前后端半分离项目

MVC模式：

- Model:业务模型，包括你的业务数据以及对业务模型的更新和响应外部对业务模型的数据请求
- View:视图，主要就是前端的模板部分（html,css）,由于jsp的缘故，view没有实现与model的解耦
- Controller:控制器，主要用于处理web请求



应用分层：

- DAO层（Data Access Object）:数据访问层，提供业务数据的处理（增删改查），持久化数据操作，其实就是jdbc操作数据库
- pojo层:存放一些实体类，主要是数据库–Java的映射对象
- service层：业务层，编写具体业务逻辑。
- servlet层：处理servlet业务请求

2.前后端分离项目（MVVM–MC）

2.1.后端

Model:业务模型+公有缓存（二级缓存）

Controller:控制器

2.2.前端

Model:视图模型，存储视图状态+本地私有缓存(一级缓存)

数据分为两种，一种是可以实现对视图组件进行控制的state变量，比如控制一个dom是否可以显示，一个dom显示的样式，这些数据不需要从服务器获取，本地就有，但是会根据不同时间进行不同状态的响应变换，多是bool类型，二是组件用于显示的数据，这些数据来自于服务器，被缓存在本地model里面，是实际的展示数据,且这些数据有可能是被单一组件私有或多个组件共享。

两类数据在命名上最好做一定的区分，好快速定位，state变量可以这样命名：xxxState(xxx表示元素的id名或者类名)，data可以命名为xxxData,对应的方法名可以是xxxStateUpdate(int mode),根据传入的mode做不同的操作，也可以直接命名xxxStateXxx(后面的Xxx直接表示操作)，也可以将xxxState和xxxStateXxx直接封装成对象，方法的命名就可以变成Xxx

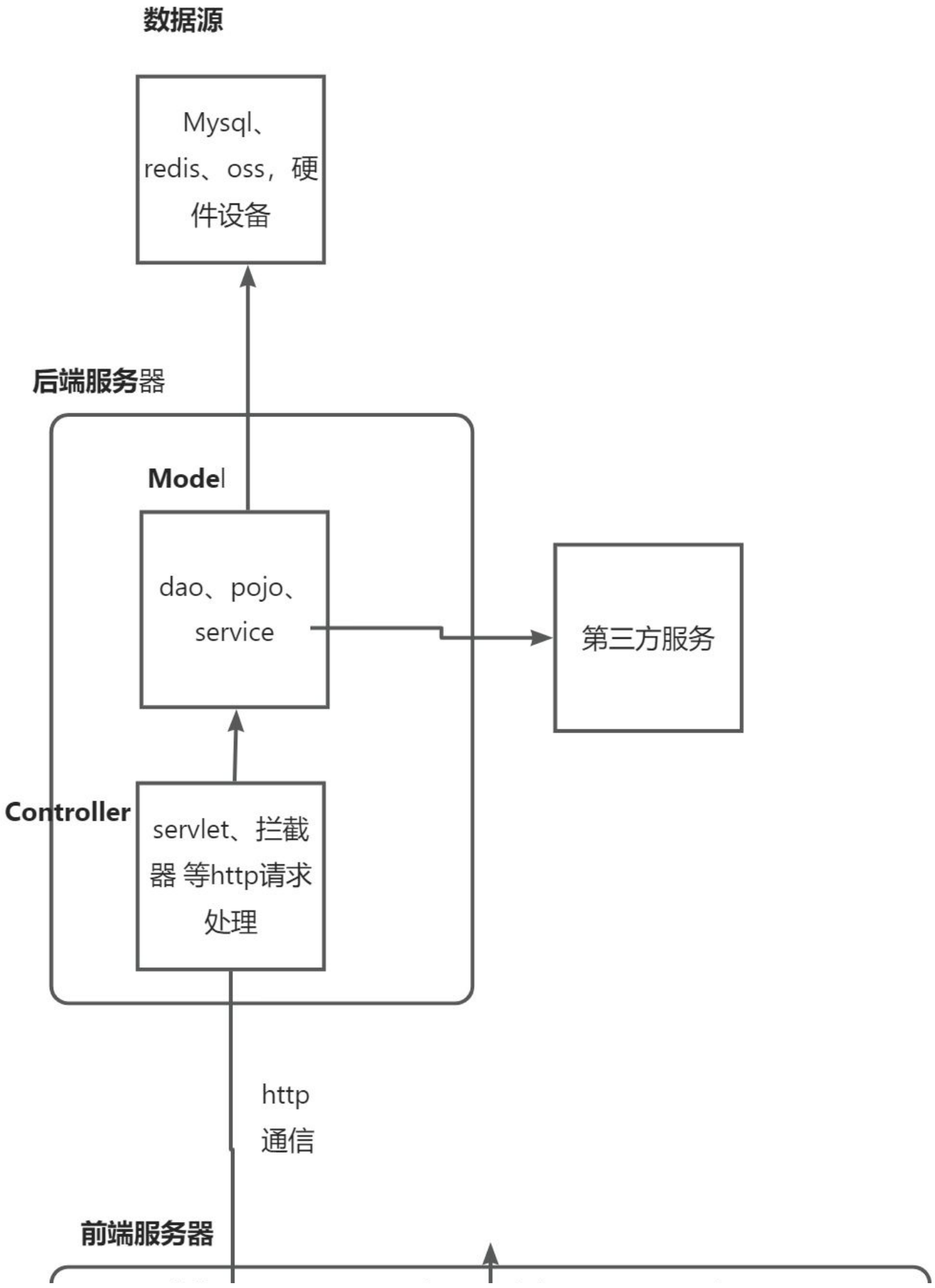
```
interface UserFormState{
    visiable:bool
    userName:string
    pwd:string
}
const userFormState=reactive<UserFormState>({
    visiable=false,
    pwd="",
})
const login=function(){

}
```

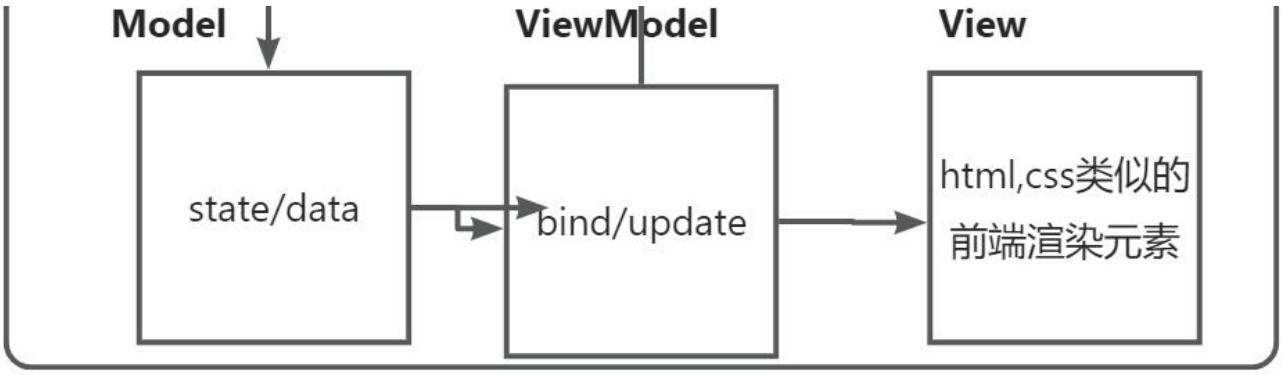
ViewModel:数据绑定层：一要将视图与视图状态和视图数据进行绑定，二要根据视图模型中数据和状态的改变去修改视图的内容。

View：纯视图层，只做数据展示，实际上就是html中一个个DOM。

2.3.两级缓存设计

- 本地私有缓存缓存当前用户的私有数据，这些数据只可能被当前用户更改，也就是说除了当前用户自己，其他用户不可能对该用户的数据进行修改，比如说用户的个人信息，token等。
- 服务端的二级缓存，缓存用户的公有数据，特点是可以被多用户访问，多用户修改。


数据源
Mysql、redis、oss，硬件设备
后端服务器
Model
dao、pojo、service
第三方服务
Controller
servlet、拦截器 等http请求处理
http
通信
前端服务器


Model
ViewModel
View
state/data
bind/update
html,css类似的
前端渲染元素