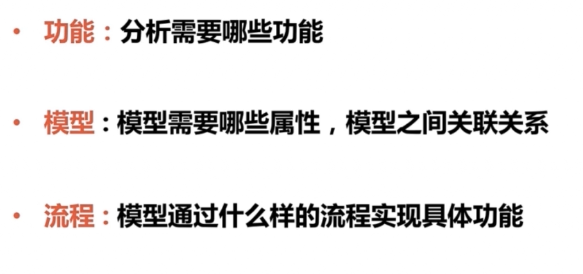
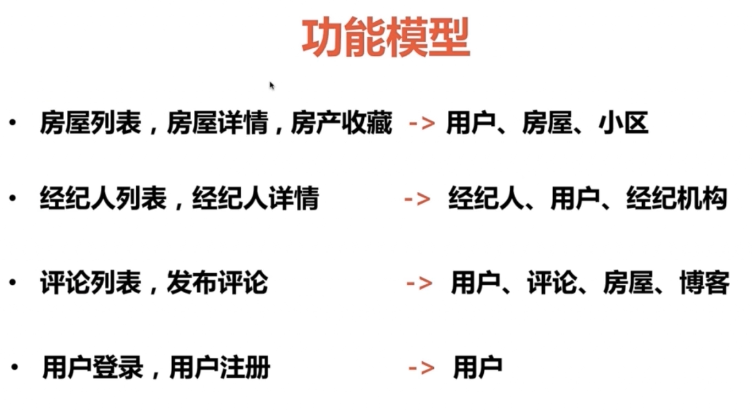
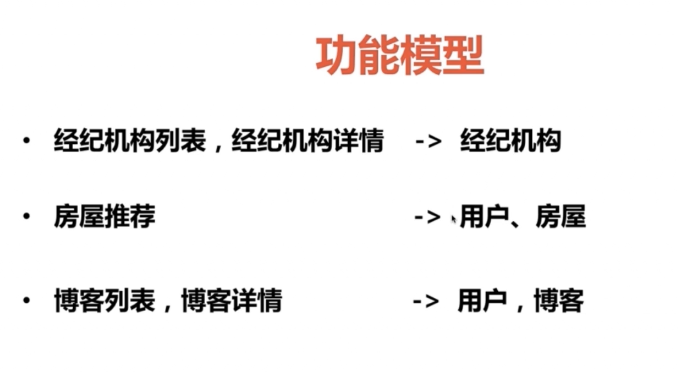
# 需求分析

## 1.1思路

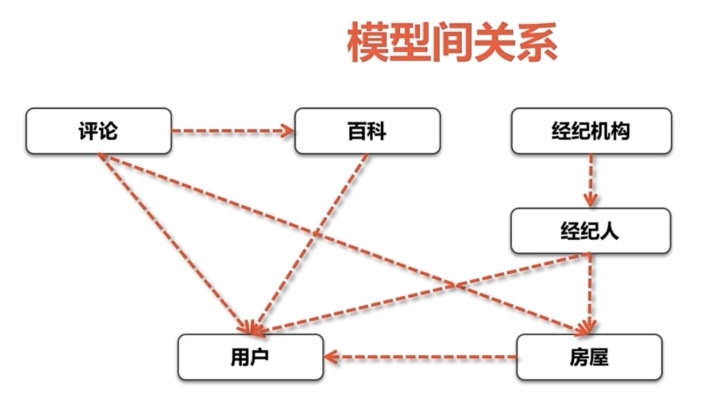


### 1.1.1由功能推算出模型





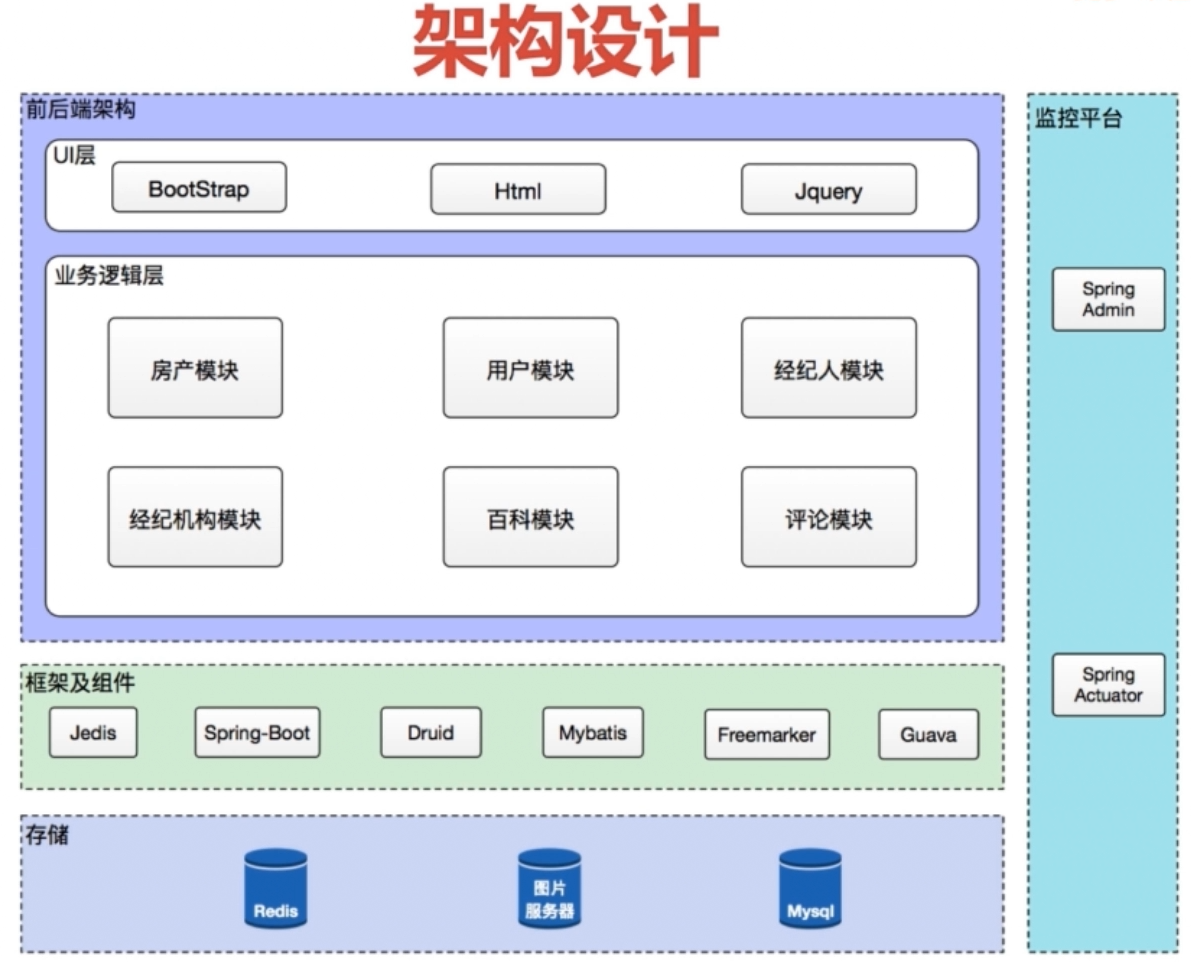
### 1.1.2模型之间的关系



# 技术选型

1. SpringBoot
2. Guava：google的java工具类库
3. Mybatis
4. Druid：自带监控的数据库连接池
5. Freemarker：模板引擎（也可选Thymeleaf）
6. Nginx
7. Redis

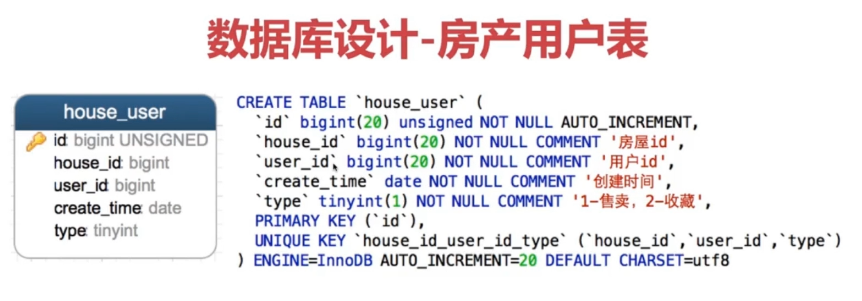
# 架构设计



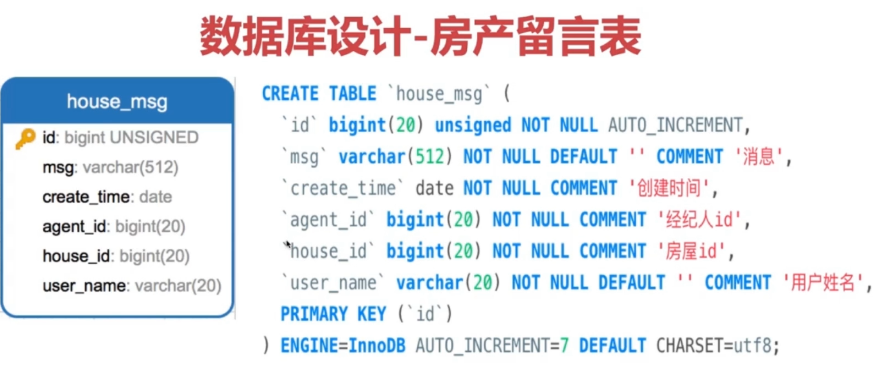
# 4.数据库设计



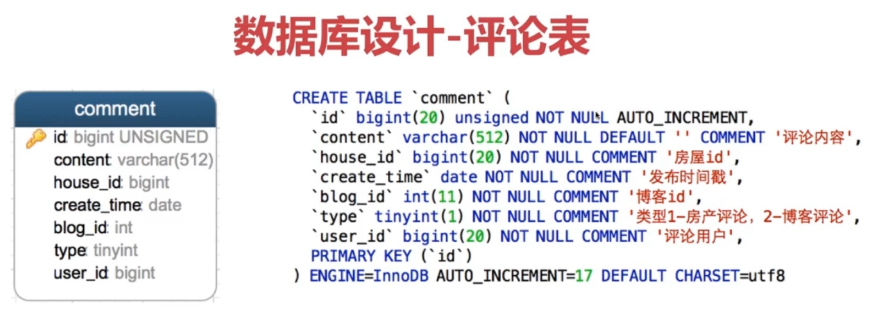




|  |
| --- |
| 房产和用户之间的关系是比较多的，单靠外键+状态字段等等已经无法描述清楚，即便可以，所以可用一张表专门用于描述两者之间的关系。 |

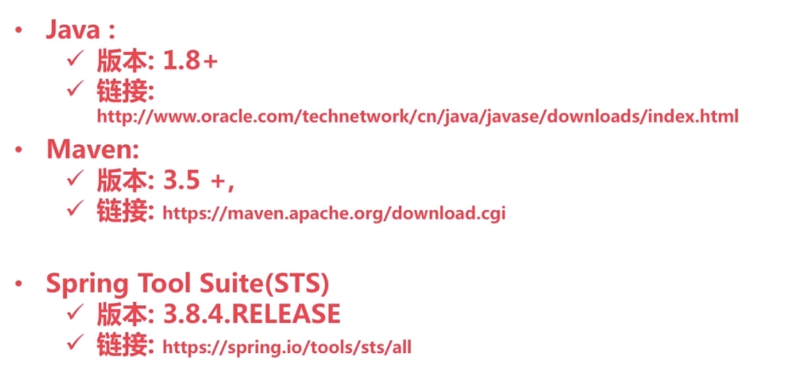








# 5.环境准备



# ========开始开发========

省略整合Mybatis、Druid等等。可以先搭个简单的框架，等需要用到什么框架了，再进行整合。

# 1.多mudule工程模式开发

这里只作为参考，因为这里分的mudule模块并不是很专业，可以参考黑马的宜立方商城搭建SOA架构。但可以先不考虑这部分内容。

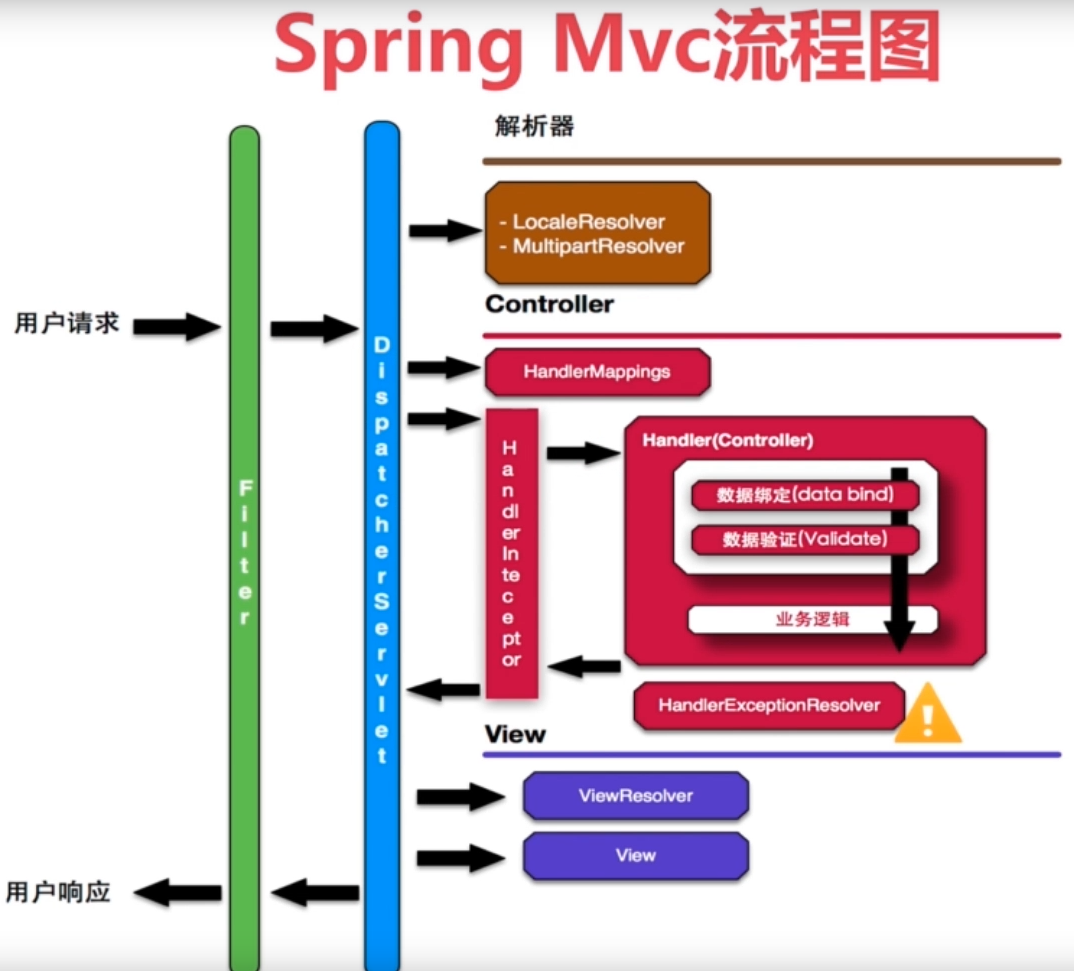


还需要先搭建一个父工程：house，用于版本管理。（父工程的打包方式是pom）。

# 2.错误页面

参考SpringBoot的错误处理机制。

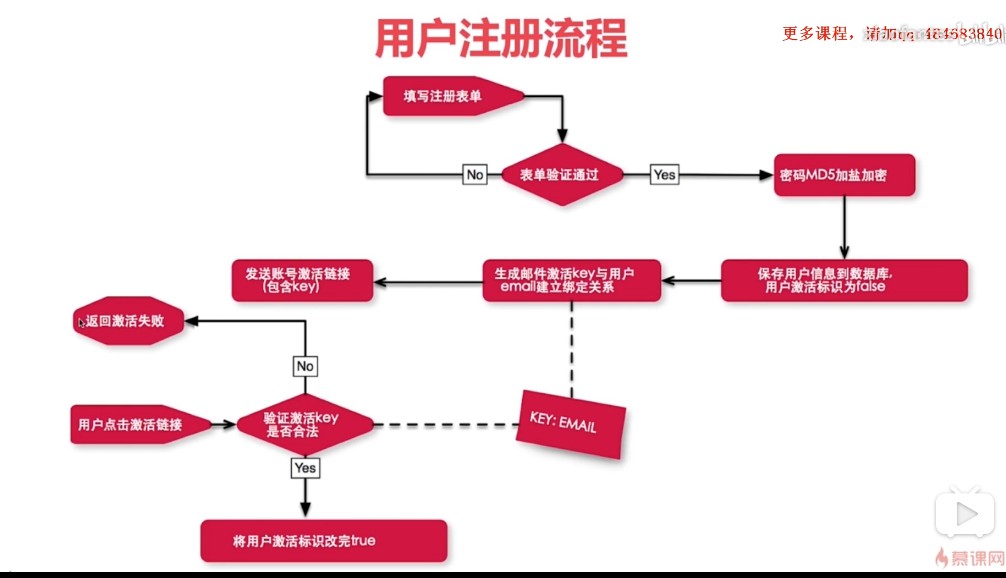
# 3.SpringMvc流程图



**用户请求——>Filter——>DispatchServlet——>解析器——>HandleMapping——>HanderInteceptor（**preHandle方法前**）——>HandAdapter——>Handler（即controller方法）——>HanderInteceptor（**postHandle方法后，视图解析器渲染前**）——>View——>VIewResolver——>DispatchServlet——>Filter——>用户**

# **4.用户注册功能**

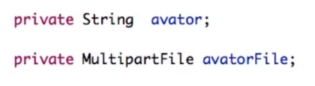
## **4.1 注册流程**



## **4.2 代码细节**

**头像的接收：**

#### 可以在实体类中定义**MultipartFile**属性来接收图像，另外在定义一个String来记录图像的url。



#### **4.3 java类库--guava**

<dependency>  
    <groupId>com.google.guava</groupId>  
    <artifactId>guava</artifactId>  
    <version>27.0.1-jre</version>  
</dependency>

#### **1.4注册流程需要前台表单验证及后台验证，才可与数据库交互**

#### **1.5密码加盐+MD5**

加盐的目的是为了防止一些简单密码被暴力破解。

工具类：



#### 1.6 nginx中root和alias的区别

location /a/image/ {

root /local\_path/image/;

alias /local\_path/image/;

}

比如：当客户端请求 /a/image/cat.jpg 的时候,

root会把请求映射为：/local\_path/image/a/image/cat.jpg

alias会把请求映射为:/local\_path/image/cat.jpg

推荐用alias。

#### 1.7 **@Transactional(rollbackFor=Exception.class)**

Spring的事务默认只回滚runtimeException,可以通过**rollbackFor属性**修改成对运行时和编译时的异常都回滚。

# 5.用户登录功能

## 5.1 登录流程

