# 苹果商城

## 软件工程

软件工程是以工程学的方式管理软件产生的过程, 保证生出高质量软件

软件工程的生命周期

人: 出生->成长->工作->死

软件: 无->工作->死

1. 定义: 明确要做一个什么玩意儿
2. 需求分析: 需要实现哪些功能, 市场调研, 可行性分析(技术, 经济, 市场)
3. 系统设计: 概要设计, 详细设计
4. 编码实现:
5. 软件测试:
6. 运营和维护:

## 定义

做一个和苹果商城一样的网站

Apple商城

实现产品的在线购买

## 分析

### 功能

前台:

主页

上部菜单

中间产品列表

底部网站信息

分类页

上部菜单(当前分类的产品)

中间产品列表

底部网站信息

购买页

上部菜单

中间是产品信息(产品的详情, 购买功能)

底部网站信息

购物袋

上部菜单

中间是购物列表

底部网站信息

订单

查看

用户

登录

注册

退出

后台:

产品管理

### 技术

语言: Java, HTML, CSS, JS

开发环境: JDK1.8, HTML5, CSS3

数据库: MySQL

开发工具: STS, Navicat, SpringBoot

主要技术: Mybatis, Spring, SpringMVC, FreeMarker, jQuery

### 经济

### 市场

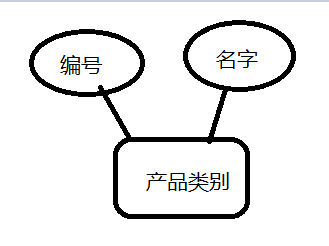
## 设计

### 概要设计

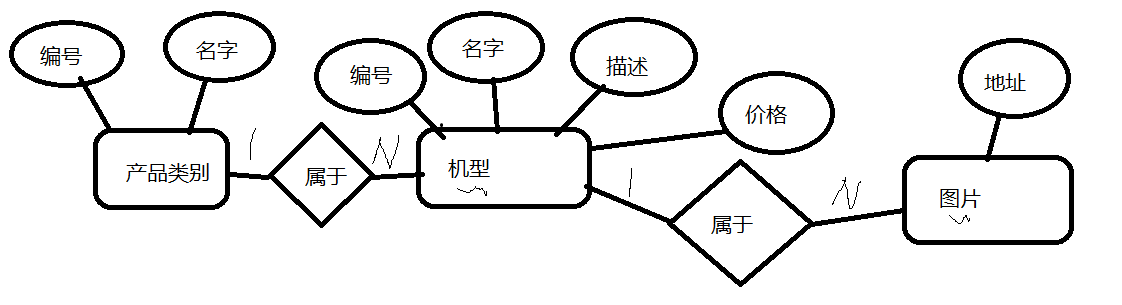
数据库的设计

#### E-R图

1. 产品类别



1. 机型

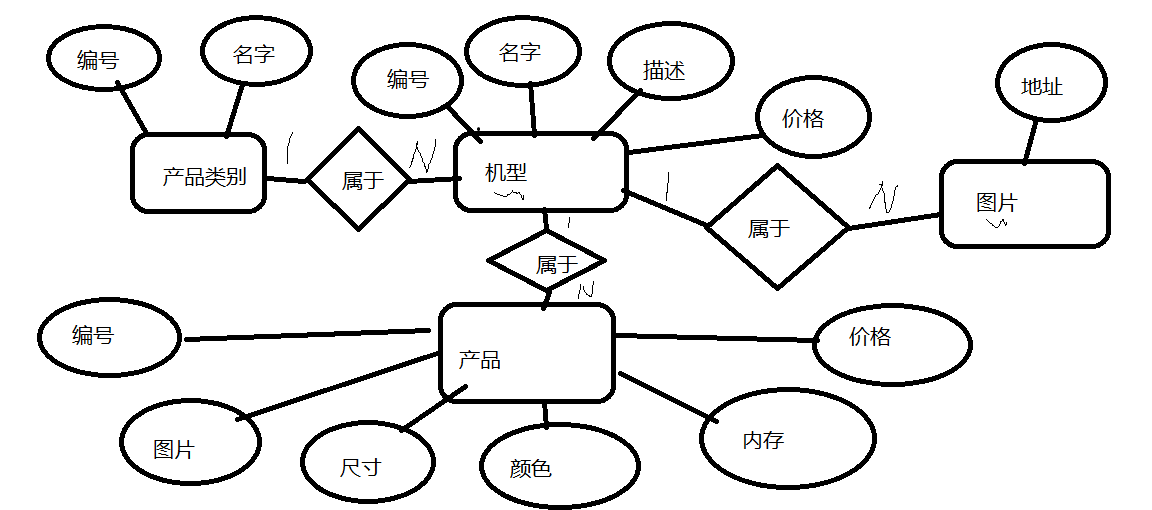


1. 商品

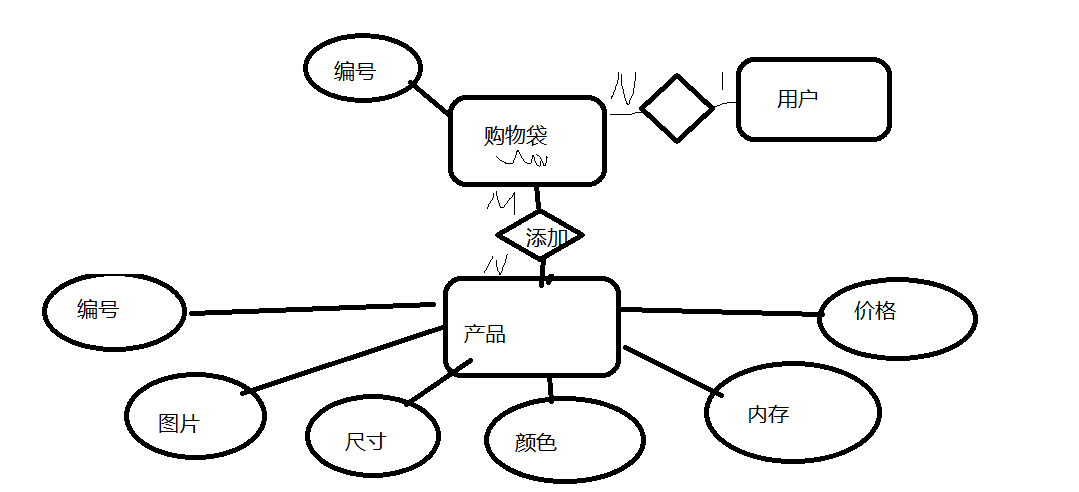
手机: 屏幕大小, 颜色, 内存

电脑: 内存, 硬盘, 描述

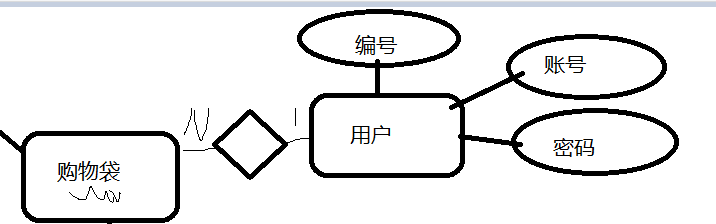
手表: 颜色, 尺寸



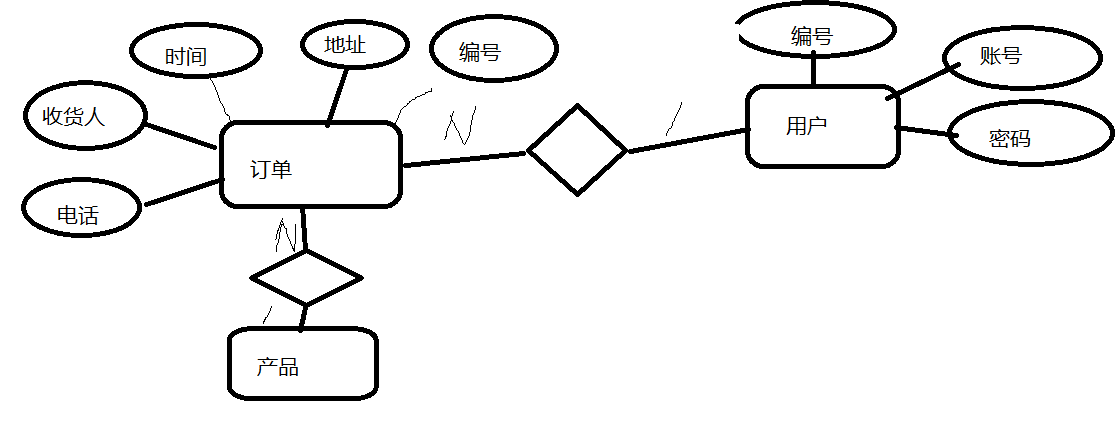
1. 购物袋



1. 用户

­

1. 订单



#### 数据库的逻辑模型

根据E-R图设计数据库的逻辑模型

1. 每一个实体, 都做成一张表, 每一个属性作为表中的一个字段
2. 一对一: 在从表中添加一个外键与主表的主键进行关联

人, 身份证

人作为主表

身份证作为从表

1. 一对多: 在从表中添加一个外键与主表的主键进行关联

多的一方作为从表, 一的一方是主表

1. 多对多

需要一个单独的关系表, 来保存两个表之间的对应关系

表的逻辑结构

1. 产品类别

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | type | | | | | | | | |
| 属性 | 字段名 | 类型 | 长度 | 小数 | 为空 | 主键 | 默认值 | 约束 | 描述 |
| 编号 | id | int |  |  | No | PK |  | 自增 |  |
| 名字 | name | varchar | 16 |  | No |  |  |  | 产品类别名 |

1. 机型

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | model | | | | | | | | |
| 属性 | 字段名 | 类型 | 长度 | 小数 | 为空 | 主键 | 默认值 | 约束 | 描述 |
| 编号 | id | int |  |  | No | PK |  | 自增 |  |
| 名字 | name | varchar | 32 |  | No |  |  |  |  |
| 描述 | describe | text |  |  | No |  |  |  |  |
| 价格 | price | int |  |  | No |  |  |  |  |
| 类别 | type | int |  |  | No | FK |  |  |  |

1. 图片

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | image | | | | | | | | |
| 属性 | 字段名 | 类型 | 长度 | 小数 | 为空 | 主键 | 默认值 | 约束 | 描述 |
| 编号 | id | int |  |  | No | PK |  | 自增 |  |
| 地址 | url | varchar | 128 |  | No |  |  |  |  |
| 机型 | Model | Int |  |  | No | FK |  |  |  |

数据库的物理模型

在Mysql中创建相应的表

### 详细设计

#### 使用MVC三层架构

M：模型

数据模型：

数据的呈现方式：**实体类**/List/Map/

com.st.apple.pojo

功能模型:

持久层：数据库的访问操作

com.st.apple.mapper

业务层：对系统中的业务逻辑进行处理

com.st.apple.service

V：视图

网页：**模板**代替jsp，静态资源：静态CSS样式，JS，image

动态页面：数据的渲染

FreeMarker

JSON/XML

C：控制

SpringMVC的Handler作为控制层

controller中的

com.st.apple.controller

#### 功能的详细设计：

1. 主页功能

|  |  |
| --- | --- |
| 功能名 | 主页 |
| 角色 | 游客，登录的用户 |
| 启动 | 输入网址 |
| 页面 | FreeMarker模板 |
| 输入 | 无 |
| 输出 | 显示分类菜单，显示产品信息 |
| 数据库 | 查询分类表(type表)，查询model表 |

MVC三层设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| View层 | Controller层 | Model层 | | |
| Service层 | Mapper层 | 实体类 |
| index.html | IndexController.java | ItypeService.java  TypeServiceImpl.java | TypeMapper.java  TypeMapper.xml | Type.java |

View层

|  |  |  |  |
| --- | --- | --- | --- |
| url | 请求方式 | 响应方式 | 功能的描述 |
| /index | Get | index.html | 当请求url时，把查询到的数据显示在index.html中 |

Controller层

|  |  |  |  |
| --- | --- | --- | --- |
| 类 | url | 方法 | 描述 |
| IndexController.java | /index | public ModelAndView index() |  |

Model层

|  |  |  |  |
| --- | --- | --- | --- |
| 模型 | 类 | 方法 | 描述 |
| Service层 | ItypeService.java  TypeServiceImpl.java | List<Type> getAll() | 获取所有的产品分类 |
| Mapper | TypeMapper.java  TypeMapper.xml | List<Type> selectAll() | 查询所有的产品分类 |
| 实体类 | Type.java | 属性及get(),set()方法 | 对应数据表中的属性 |