长 沙 学 院

课程设计说明书

|  |  |
| --- | --- |
| 题目 | 网上商店前台功能的设计与实现 |
| 学院 | 计算机工程与应用数学 |
| 专业(班级) | 软件开发（2015级03班） |
| 姓名 | 晏司举 |
| 学号 | B20150304508 |
| 指导教师 | 马丽、段晓娟 |
| 起止日期 | 2018.06.04—2018.06.14 |

课程设计任务书

**课程名称：**Web应用程序课程设计

**设计题目：**网上商店前台功能的设计与实现

# 已知技术参数和设计要求：

1. 问题描述（功能要求）：

分析设计一个网上商店的前台功能部分。完成本系统的如下功能：

* + 用户注册和登录
  + 展示特定的某类商品：展示的方式有多种，可以按类别、按销售排行榜、按上架时间等，有层次的展示商品，提高用户的购买兴趣。对于某种商品，除显示商品信息以外，还可以附加评论、推荐相关类型商品。
  + 商品查询：根据多种条件来查询，比如商品名称、价格、颜色等特征信息。能进行模糊匹配，包含关键字的商品都能查到。
  + 购物车：只有注册用户才能购买。为用户提供购物车，购物车内同类商品的数量可以修改，也可以删除购物车里已有的商品。购物车信息保存，便于下次购物时继续选购商品。

**2. 运行环境要求：**

正确安装、配置和运行JDK。

正确安装、配置和运行Tomcat。

正确安装、配置和运行Eclipse或MyEclipse。

正确安装、配置和运行数据库和JDBC驱动程序。

3. 技术要求：

要求使用Java语言，利用面向对象的方法、编程思想来完成系统的设计；在设计的过程中，建立清晰的类层次；在系统设计中要分析和定义各个类，每个类中要有各自的属性和方法。写出需求分析，UML设计图，数据库设计图等。

长沙学院课程设计鉴定表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | | 晏司举 | | 学号 | B20150304508 | 专业 | 软件工程 | | 班级 | 15级03班 |
| 设计题目 | | | 网上商店前台功能的设计与实现 | | | | | 指导教师 | | 马丽、段晓娟 |
| 指导教师意见：  评定等级： 教师签名： 日期： | | | | | | | | | | |
| 答辩小组意见：  评定等级：　　　　　答辩小组长签名：　　　　　日期： | | | | | | | | | | |
| 教研室意见：  教研室主任签名： 日期： | | | | | | | | | | |
| 系（部）意见：  系主任签名：　　　　　　　　日期： | | | | | | | | | | |
| 说明 | 课程设计成绩分“优秀”、“良好”、“中等”、“及格”、“不及格”四类； | | | | | | | | | |

# 摘 要

本系统为华为手机商城前台设计系统，用户可以进行注册、登录，登录成功后跳转至首页，可以查看商城所有商品，亦或分类查询或按照价格销量高低查询商城商品，可以在右上角搜索框输入商品关键字进行模糊查询，可以点击商品图片进入单个商品信息展示界面，每个登录用户均可以在该页面下方评论一次。可选择商品颜色、配置及数量将商品添加至购物车,并在购物车页面进行结算，还可以点击导航栏的“我的订单”查看用户的所有订单详情。

关键词：首页，查询，评论，购物车，订单详情

目 录

[已知技术参数和设计要求： 2](#_Toc517080632)

[摘 要 4](#_Toc517080633)

[第1章 设计内容与要求 6](#_Toc517080634)

[1.1课程名称：Java Web 课程设计 6](#_Toc517080635)

[1.2设计要求 6](#_Toc517080636)

[第2章 需求分析 7](#_Toc517080637)

[2.1需求描述 7](#_Toc517080638)

[2.2设计环境 9](#_Toc517080639)

[2.3用例规约 9](#_Toc517080640)

[2.4系统用例图 13](#_Toc517080641)

[第3章 概要设计 14](#_Toc517080642)

[3.1功能结构 14](#_Toc517080643)

[3.2系统流程图 15](#_Toc517080644)

[3.3数据库设计 15](#_Toc517080645)

[第4章 详细设计 27](#_Toc517080646)

[4.1 首页模块 27](#_Toc517080647)

[4.2 商品详情模块 28](#_Toc517080648)

[4.3 购物车模块 30](#_Toc517080649)

[1.将商品添加到购物车 30](#_Toc517080650)

[2.修改购物车内商品数量 31](#_Toc517080651)

[3.删除购物车内商品 32](#_Toc517080652)

[4.4 订单模块 33](#_Toc517080653)

[4.5 评论模块 34](#_Toc517080654)

[第5章 测试 37](#_Toc517080655)

[5.1 登录模块测试 37](#_Toc517080656)

[5.2 注册模块测试 38](#_Toc517080657)

[总 结 42](#_Toc517080658)

[参考文献 43](#_Toc517080659)

[附 录 主要源代码 43](#_Toc517080660)

第1章 设计内容与要求

# 1.1课程名称：Java Web 课程设计

设计题目：网上商店前台功能的设计与实现

# 1.2设计要求

具体要求如下：

1. 问题描述（功能要求）：

（1）分析设计一个网上商店的前台功能部分。完成本系统的如下功能：

用户注册和登陆

展示特定的某类商品：展示的方式有多种，可以按类别、按销售排行榜、按上架时间等，有层次的展示商品，提高用户的购买兴趣。对于某种商品，除显示商品信息以外，还可以附加评论、推荐相关类型商品。

商品查询：根据多种条件来查询，比如商品名称、价格、颜色等特征信息。能进行模糊匹配，包含关键字的商品都能查到。

购物车：只有注册用户才能购买。为用户提供购物车，购物车内同类商品的数量可以修改，也可以删除购物车里已有的商品。购物车信息保存，便于下次购物时继续选购商品；实现每样商品库存的修改。

2. 运行环境要求：

正确安装、配置和运行Tomcat。

正确安装、配置和运行Eclipse或MyEclipse。

正确安装、配置和运行MySQL数据库和JDBC驱动程序。

3. 技术要求：

要求使用Java语言、JSP、Servlet等技术，利用面向对象的方法和编程思想来完成系统的设计；在设计的过程中，建立清晰的类层次；在系统设计中要分析和定义各个类，每个类中要有各自的属性和方法。写出需求分析，UML设计图，数据库设计图。

备注：网上商店的商品不能和书本案例的一样

**http://hao.uisdc.com/ 模板库**

1. 评分方法 : 问题分析和功能定义准确，清晰了解整个系统中定义的类和数据库的关系，文档齐全，符合内容规范，实现功能要求，计“中”；能扩充功能，则相应提高等级；问题理解不正确，设计混乱，文档不全，内容不完整，代码不理解的，答辩不通过的，计“不及格”。

第2章 需求分析

# 2.1需求描述

2.1.1用户模块

2.1.1.1注册

需求编号：01-01

需求描述：游客进入该系统首先进入登录界面，可点击右上角的注册超链接跳转至注册界面填写手机号、用户名、密码、姓名、收货地址信息进行注册，若有一项为空或者不符合指定的格式给出警告提示，若各项数据均合法则注册成功，数据库中的user表新增加一条用户记录。

2.1.1.2登录

需求编号：01-02

需求描述：注册用户进入登录界面输入用户名和密码进行登录，若用户名或密码有一项为空或者是用户名和密码不匹配则给出提示，用户名和密码均正确则登录成功，跳转至商城首页。

2.1.2首页模块

2.1.2.1 显示所有商品信息

需求编号：02-01

需求描述：用户登录成功后跳转至首页，该页面所有商品信息主要通过从数据库动态获取手机的图片配置等信息来展示。

2.1.2.2 分类排序

需求编号：02-02

需求描述：用户可点击导航栏下方的分类排序选项对商品进行排序，可以按照价格和销量的从高到低、从低到高以及型号和品牌。

2.1.3购物车模块

2.1.3.1 查看购物车信息

需求编号：03-01

需求描述：用户登录成功后跳转至首页，在该页面有“我的购物车” 链接，用户可以通过此链接查看购物车内信息。

2.1.3.2 修改商品数量

需求编号：03-02

需求描述：用户在查看购物车内信息时可以修改商品数量

2.1.3.3 删除商品

需求编号：03-03

需求描述：用户在查看购物车内信息时可以删除某个商品

2.1.4订单模块

2.1.4.1 下订单

需求编号：04-01

需求描述：用户在查看商品详情时可以选择配置、颜色和数量，同时下单购买。

2.1.5 评论模块

2.1.5.1 查看评论

需求编号：05-01

需求描述：用户在查看商品详情时可以查看商品的评论

2.1.5.2 评论商品

需求编号：05-02

需求描述：每个登录用户可以对每个商品评论一次

# 2.2设计环境

软硬件开发环境:windows 10系统、 使用Eclipse开发

运行环境:windows 10

# 2.3用例规约

表2.1 注册用户用例规约

|  |  |
| --- | --- |
| 用例名称： | 注册用户 |
| 用例ID： | 01 |
| 角色： | 游客 |
| 用例说明： | 游客注册用户。 |
| 前置条件： | 无 |
| 基本事件流： | 1.游客进入注册用户页面  2.游客输入基本账户信息  3.游客点击“注册”按钮。  4.系统弹出“注册成功”信息页面。  5. 5秒后页面跳转到登录页面，游客也可以点击“登录”链接直接进入登录页面 |
| 其他事件流： | 第2步：游客输入的用户名、姓名、密码、收货地址、手机号为必选项，如果某一项没有填写，要给出提示信息。  第2步：游客输入的系统用户信息不符合格式要求，系统给出提示信息，要求重新输入。  第2步：如果有重名的用户名，则注册失败，给出提示信息：“此用户名已被注册”。 |
| 异常事件流： | 第3步，系统保存用户信息时出现系统故障，例如网络故障，数据库服务器故障，系统弹出系统异常页面，提示User：注册失败。 |
| 后置条件： |  |

表2.2 登录用例规约

|  |  |
| --- | --- |
| 用例名称： | 登录 |
| 用例ID： | 02 |
| 角色： | 用户 |
| 用例说明： | 用户登录到系统 |
| 前置条件： | 注册 |
| 基本事件流： | 1.用户进入登录页面  2.用户输入账号和密码  3.用户点击“登录”按钮  4.登录成功，系统跳转到首页 |
| 其他事件流： | 第2步：用户输入的用户名或密码错误，登录失败，并提示用户“帐号或密码输入错误，请重新输入” |
| 异常事件流： |  |
| 后置条件： |  |

表2.3 查询商品规约

|  |  |
| --- | --- |
| 用例名称： | 查询商品 |
| 用例ID： | 03 |
| 角色： | 游客 |
| 用例说明： | 用户查询商城内的商品 |
| 前置条件： | 无 |
| 基本事件流： | 1.用户进入系统首页  2.用户输入关键字，点击“搜索”按钮  3.页面显示含有关键字的商品 |
| 其他事件流： | 第3步： 没有含有此关键字的商品，提示用户“您搜索的商品不存在” |
| 异常事件流： |  |
| 后置条件： |  |

表2.4 下订单用例规约

|  |  |
| --- | --- |
| 用例名称： | 下订单 |
| 用例ID： | 04 |
| 角色： | 用户 |
| 用例说明： | 用户购买商城内的商品 |
| 前置条件： | 登录 |
| 基本事件流： | 1.用户进入系统首页  2.用户选中某一商品，点击其图片，进入商品详情页面  3.用户选择商品配置、颜色、数量后，点击“立即购买”，进入到订单页面  4.用户确认订单信息后，确认下单，订单创建成功 |
| 其他事件流： | 第3步：用户没有选择颜色或者配置即点击“立即购买”，系统提示用户“请选择颜色或配置”  第3步：用户没有登录，点击“立即购买”后进入登录页面 |
| 异常事件流： |  |
| 后置条件： | 用户创建订单后，可以查看订单详情 |

表2.5 加入购物车用例规约

|  |  |
| --- | --- |
| 用例名称： | 加入购物车 |
| 用例ID： | 05 |
| 角色： | 用户 |
| 用例说明： | 用户将商品暂时加入购物车 |
| 前置条件： | 登录 |
| 基本事件流： | 1.用户进入系统首页  2.用户选中某一商品，点击其图片，进入商品详情页面  3.用户选择商品配置、颜色、数量后，点击“加入购物车”，系统提示用户“商品加入购物车成功”。用户可以重新查看商品详情，也可以查看购物车内商品 |
| 其他事件流： | 第3步：用户没有选择颜色或者配置即点击“加入购物车”，系统提示用户“请选择颜色或配置”  第3步：用户没有登录，点击“加入购物车”后进入登录页面 |
| 异常事件流： |  |
| 后置条件： |  |

# 2.4系统用例图

图片包含 文字, 地图

已生成极高可信度的说明

图2.1 系统用例图

第3章 概要设计

# 3.1功能结构

本系统主要分为注册登录模块、首页模块、商品信息展示模块、购物车模块、订单模块,具体的模块图如下:

**图3.1 系统功能结构图**

# 3.2系统流程图

# 3.3数据库设计

表0 [用户] 表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | User | | | | | |
| 数据库用户 | |  | | | | | |
| 主键 | | Id | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | | Id | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| *0* | id | int(11) | *N* | Y | 高 | 无 | *主键（用户id）* |
| 1 | tel | varchar(11) | *N* | *Y* | 高 | 无 | *手机号* |
| 2 | userName | *Varchar(18)* | *N* | *Y* |  |  | *用户名* |
| 3 | password | *Varchar(255)* | *N* |  |  |  | *密码* |
| 4 | address | *Varchar(100)* | *N* |  |  |  | *收货地址* |
| 5 | name | *Varchar(18)* | *N* |  |  |  | *收货人* |
| sql脚本 | | DROP TABLE IF EXISTS `user`;  CREATE TABLE `user` (  `id` int(11) NOT NULL AUTO\_INCREMENT,  `tel` varchar(11) NOT NULL,  `userName` varchar(18) NOT NULL,  `password` varchar(255) NOT NULL,  `address` varchar(100) NOT NULL,  `name` varchar(18) NOT NULL,  PRIMARY KEY (`id`),  UNIQUE KEY `userName\_unique` (`userName`)  ) ENGINE=InnoDB AUTO\_INCREMENT=110017 DEFAULT CHARSET=utf8; | | | | | |
| 备注 | | *无* | | | | | |
|  | |  | | | | | |

表1:[购物车]表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | ShoppingTrolley | | | | | |
| 数据库用户 | |  | | | | | |
| 主键 | |  | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| *0* | id | int(11) | *N* | Y |  | 无 | *主键（购物车id）* |
| 1 | userId | *int(11)* | *N* | *Y* |  |  | *用户id(*外键*)* |
| sql脚本 | | DROP TABLE IF EXISTS `shoppingtrolley`;  CREATE TABLE `shoppingtrolley` (  `id` int(11) NOT NULL AUTO\_INCREMENT,  `userId` int(11) NOT NULL,  PRIMARY KEY (`id`),  KEY `fk\_shoppingTrolley\_userId` (`userId`),  CONSTRAINT `fk\_shoppingTrolley\_userId` FOREIGN KEY (`userId`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE NO ACTION  ) ENGINE=InnoDB AUTO\_INCREMENT=2215 DEFAULT CHARSET=utf8; | | | | | |
| 备注 | | *无* | | | | | |
|  | |  | | | | | |

表2:[购物车内商品]表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | |  | | | | | |
| 数据库用户 | |  | | | | | |
| 主键 | |  | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| *0* | shoppingTrolleyId | int(11) | *N* |  | 高 | 无 | *组合主键（购物车id）外键* |
| 1 | goodsId | int(11) | *N* |  | 高 | 无 | *组合主键 商品id(外键)* |
| 2 | quantity | int(11) | *Y* |  |  | 1 | *数量* |
| 2 | amount | *double* |  |  |  |  | *金额* |
| 3 | type | *varchar(15)* | *N* |  |  |  | *配置* |
|  | batteryCapacity | *varchar(20)* | *N* |  |  |  | *电池容量* |
|  | frontFacingCamera | *varchar(20)* | *N* |  |  |  | *前置摄像头* |
|  | rearCamera | *varchar(20)* | *N* |  |  |  | *后置摄像头* |
| sql脚本 | | DROP TABLE IF EXISTS `goodsinshoppingtrolley`;  CREATE TABLE `goodsinshoppingtrolley` (  `shoppingTrolleyId` int(11) NOT NULL,  `goodsId` int(11) NOT NULL,  `quantity` int(11) DEFAULT '1',  `amount` double NOT NULL,  `type` varchar(15) DEFAULT NULL,  `batteryCapacity` varchar(20) DEFAULT NULL,  `frontFacingCamera` varchar(20) DEFAULT NULL,  `rearCamera` varchar(20) DEFAULT NULL,  PRIMARY KEY (`shoppingTrolleyId`,`goodsId`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8; | | | | | |
| 备注 | | *无* | | | | | |
|  | |  | | | | | |

表3：[订单]表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | Order | | | | | |
| 数据库用户 | |  | | | | | |
| 主键 | |  | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| *0* | id | int | *N* | Y | 高 | 无 | *主键（订单号）* |
| 1 | time | *dataTime* | *N* |  | 高 | 无 | *订单时间* |
| 2 | address | *Varchar(100)* | *N* | *N* | *低* | 无 | *收货地址* |
| 3. | name | *Varchar(18)* | *N* | *N* |  |  | *收货人* |
| 4 | amount | *double* | *N* |  |  | * 0 | *金额* |
| 5 | userId | *int(11)* | *N* |  |  | 外键 | *对应的用户* |
| sql脚本 | | DROP TABLE IF EXISTS `order`;  CREATE TABLE `order` (  `orderId` int(11) NOT NULL,  `time` datetime NOT NULL,  `address` varchar(100) NOT NULL,  `name` varchar(18) NOT NULL,  `amount` double NOT NULL,  `userId` int(11) NOT NULL,  PRIMARY KEY (`orderId`),  KEY `fk\_order\_userId` (`userId`),  CONSTRAINT `fk\_order\_userId` FOREIGN KEY (`userId`) REFERENCES `user` (`id`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8; | | | | | |
| 备注 | | *无* | | | | | |
|  | |  | | | | | |

表4：[商品]表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | goods | | | | | |
| 数据库用户 | |  | | | | | |
| 主键 | |  | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| *0* | id | int | *N* | Y | 高 | 无 | *主键（商品id）* |
| 1 | name | *Varchar(20)* | *N* |  | 高 | 无 | *商品名称* |
| 2 | `battery Capacity | *Varchar(20)* | *N* |  |  |  | *电池容量* |
| 3 | frontFacingCamera | *Varchar(20)* | *N* |  |  |  | *前置摄像头* |
| 4 | rearCamera | *Varchar(20)* | *N* |  |  |  | *后置摄像头* |
| sql脚本 | | DROP TABLE IF EXISTS `goods`;  CREATE TABLE `goods` (  `id` int(11) NOT NULL AUTO\_INCREMENT,  `name` varchar(20) NOT NULL,  `batteryCapacity` varchar(20) NOT NULL,  `frontFacingCamera` varchar(20) NOT NULL,  `rearCamera` varchar(20) NOT NULL,  PRIMARY KEY (`id`)  ) ENGINE=InnoDB AUTO\_INCREMENT=34 DEFAULT CHARSET=utf8; | | | | | |
| 备注 | |  | | | | | |
|  | |  | | | | | |

表5：[商品价格]表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | goodsprice | | | | | |
| 数据库用户 | |  | | | | | |
| 主键 | |  | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| *0* | goodsId | int | *N* |  | 高 | 无 | *组合主键 外键（商品id）* |
| 1 | Type | *Varchar(15)* | *N* | *N* | 高 | 无 | *组合主键 型号* |
| 2 | Price | *double* | *N* |  | *低* | 无 | *商品价格* |
| 3 | quantity | *int* |  |  |  |  | *库存* |
| 4 | SalesVolume | *int(11)* | *N* |  |  |  | *销量* |
| 5 | state | *int(11)* |  |  |  |  | *是否上下架* |
| sql脚本 | | DROP TABLE IF EXISTS `goodsprice`;  CREATE TABLE `goodsprice` (  `goodsId` int(11) NOT NULL,  `type` varchar(15) NOT NULL,  `price` double NOT NULL,  `quantity` int(11) NOT NULL,  `salesVolume` int(11) NOT NULL,  `state` int(11) NOT NULL,  PRIMARY KEY (`goodsId`,`type`),  CONSTRAINT `fk\_goodsPrice\_goodsId` FOREIGN KEY (`goodsId`) REFERENCES `goods` (`id`) ON DELETE CASCADE ON UPDATE NO ACTION  ) ENGINE=InnoDB DEFAULT CHARSET=utf8; | | | | | |
| 备注 | | *无* | | | | | |
|  | |  | | | | | |

表6：[图片地址]表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | goodsImg | | | | | |
| 数据库用户 | |  | | | | | |
| 主键 | |  | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| *0* | goodsId | int(11) | *N* |  | 高 | 无 | *组合主键 外键(商品id)* |
| 1 | color | *Varchar(25)* | *N* |  | 高 | 无 | *组合主键 颜色* |
| 2 | imgSrc | *Varchar(255)* | *N* |  | *低* |  | *图片地址（相对地址）* |
| sql脚本 | | DROP TABLE IF EXISTS `goodsimg`;  CREATE TABLE `goodsimg` (  `goodsId` int(11) NOT NULL,  `color` varchar(25) NOT NULL,  `imgSrc` varchar(255) NOT NULL,  PRIMARY KEY (`goodsId`,`color`),  CONSTRAINT `fk\_goodsImg\_goodsId` FOREIGN KEY (`goodsId`) REFERENCES `goods` (`id`) ON DELETE CASCADE ON UPDATE NO ACTION  ) ENGINE=InnoDB DEFAULT CHARSET=utf8; | | | | | |
| 备注 | | *无* | | | | | |
|  | |  | | | | | |

表7：[订单---商品]表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | goodsInOrder | | | | | |
| 数据库用户 | |  | | | | | |
| 主键 | |  | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| *0* | OrderId | int(20) | *N* |  | 高 | 无 | *组合主键 外键(订单id)* |
| 1 | goodsId | int(11) | *N* |  | 高 | 无 | *组合主键 外键(商品id)* |
| 2 | quantity | int(11) | *N* |  | *低* |  | *商品数量* |
| 3 | amount | *double* | *N* |  |  |  | *金额* |
| 4 | batteryCapacity | varchar(20) | *Y* |  |  |  | *电池容量* |
| 5 | frontFacingCamera | varchar(20) | *Y* |  |  |  | *前置摄像头* |
| 6 | rearCamera | varchar(20) | *Y* |  |  |  | *后置摄像头* |
| 7 | type | varchar(15) | *Y* |  |  |  | *商品配置* |
| sql脚本 | | DROP TABLE IF EXISTS `goodsinorder`;  CREATE TABLE `goodsinorder` (  `orderId` int(20) NOT NULL,  `goodsId` int(11) NOT NULL,  `quantity` int(11) NOT NULL,  `amount` double NOT NULL,  `batteryCapacity` varchar(20) DEFAULT NULL,  `frontFacingCamera` varchar(20) DEFAULT NULL,  `rearCamera` varchar(20) DEFAULT NULL,  `type` varchar(15) DEFAULT NULL,  PRIMARY KEY (`goodsId`,`orderId`),  KEY `fk\_goodsInOrderform\_orderformId` (`orderId`),  CONSTRAINT `fk\_goodsInOrderform\_goodsId` FOREIGN KEY (`goodsId`) REFERENCES `goods` (`id`) ON DELETE CASCADE ON UPDATE NO ACTION,  CONSTRAINT `fk\_goodsInOrderform\_orderformId` FOREIGN KEY (`orderId`) REFERENCES `order` (`orderId`) ON DELETE CASCADE ON UPDATE NO ACTION  ) ENGINE=InnoDB DEFAULT CHARSET=utf8; | | | | | |
| 备注 | | *无* | | | | | |
|  | |  | | | | | |

表8：[评论]表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | Comment | | | | | |
| 数据库用户 | |  | | | | | |
| 主键 | |  | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| *0* | userId | int | *N* |  |  | 无 | *组合主键 外键(用户id)* |
| 1 | goodsId | *int* | *N* |  |  | 无 | *组合主键 外键(商品id)* |
| 2 | content | *Varchar(800)* | *N* |  |  | 无 | *评论内容* |
| sql脚本 | | DROP TABLE IF EXISTS `comment`;  CREATE TABLE `comment` (  `userId` int(11) NOT NULL,  `goodsId` int(11) NOT NULL,  `content` varchar(800) NOT NULL,  PRIMARY KEY (`goodsId`,`userId`),  KEY `fk\_comment\_userId` (`userId`),  CONSTRAINT `fk\_comment\_goodsId` FOREIGN KEY (`goodsId`) REFERENCES `goods` (`id`) ON DELETE CASCADE ON UPDATE NO ACTION,  CONSTRAINT `fk\_comment\_userId` FOREIGN KEY (`userId`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE NO ACTION  ) ENGINE=InnoDB DEFAULT CHARSET=utf8; | | | | | |
| 备注 | | *无* | | | | | |
|  | |  | | | | | |

第4章 详细设计

# 4.1 首页模块

实现:从数据库中查询出商品信息，传到controller层，controller层将信息整合后存在session中，用户即可以通过session访问到商品信息。另外，为了提高访问速度，在第一次查询商品信息时，将商品信息缓存到内存中，后续访问内存即可得到数据。

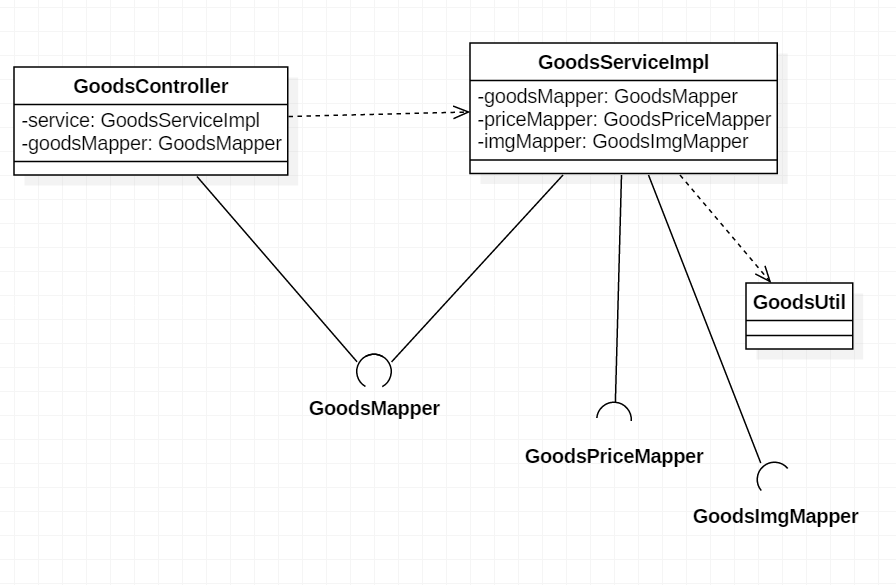


图4.1 首页模块类图

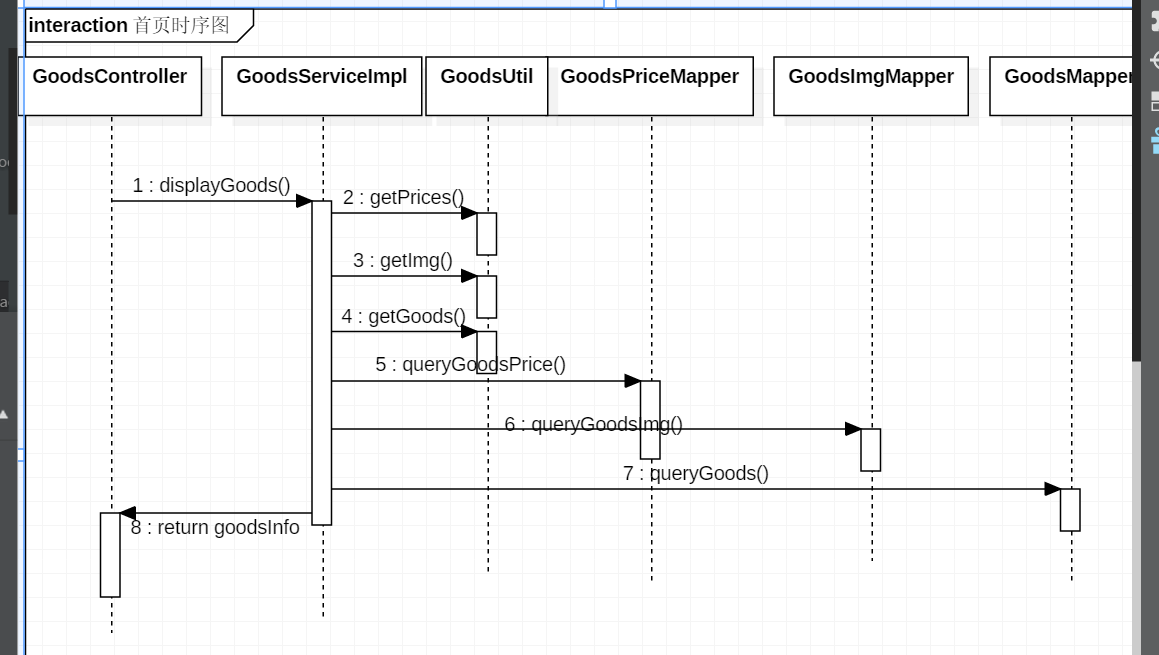


图4.2 首页模块时序图

# 4.2 商品详情模块

实现；根据请求的id查询商品信息、价格、评论，存到session中，用户即可以访问商品信息。

图片包含 文字, 地图

已生成极高可信度的说明

图4.3 商品详情模块类图

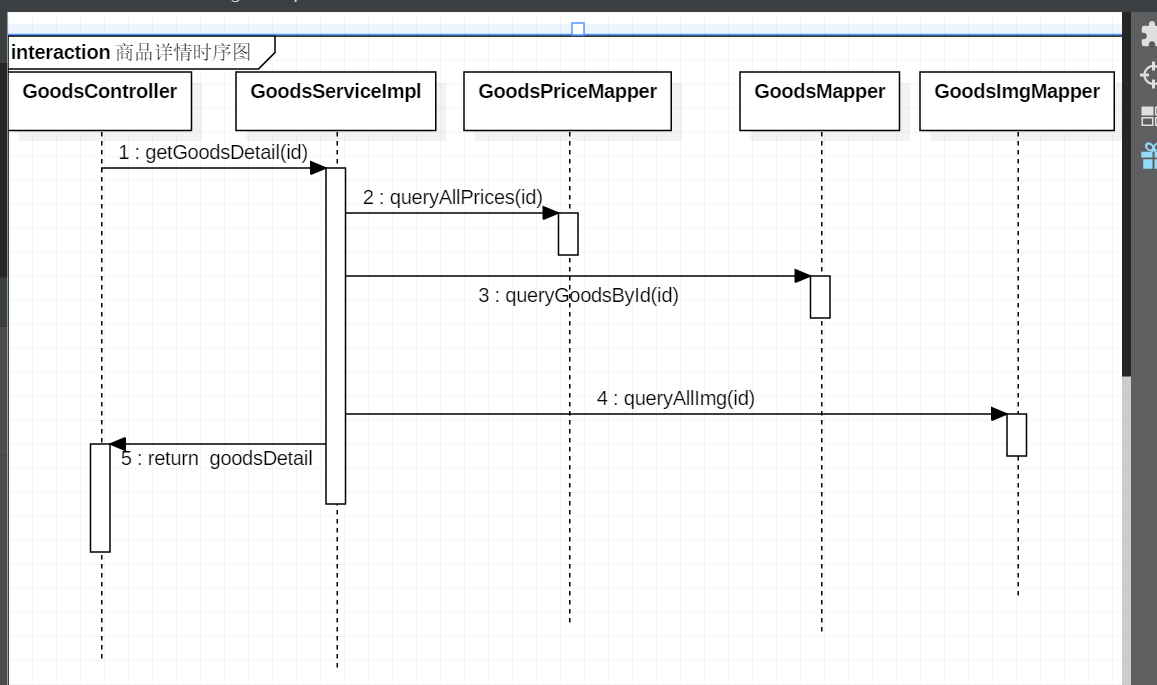


图4.4 商品详情模块时序图

图4.4 商品详情模块时序图

# 4.3 购物车模块

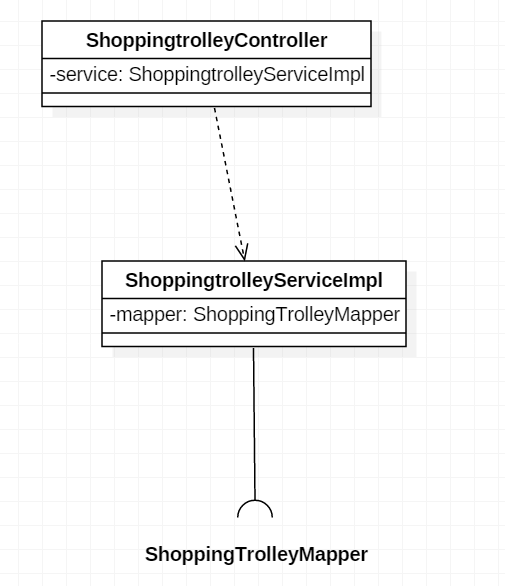


图4.5 购物车模块类图

## 1.将商品添加到购物车

用户选择商品类型和数量后，即可将商品添加到购物车内。若用户未登录，则会被拦截器拦截并跳转到登录页面。后台在session中取出购物车id，将数据传到Service层。Service层根据传入的购物车id和商品id判断购物车内是否已有该商品。若已有该商品，则直接更新数量和金额即可；否则在购物车内添加该商品。

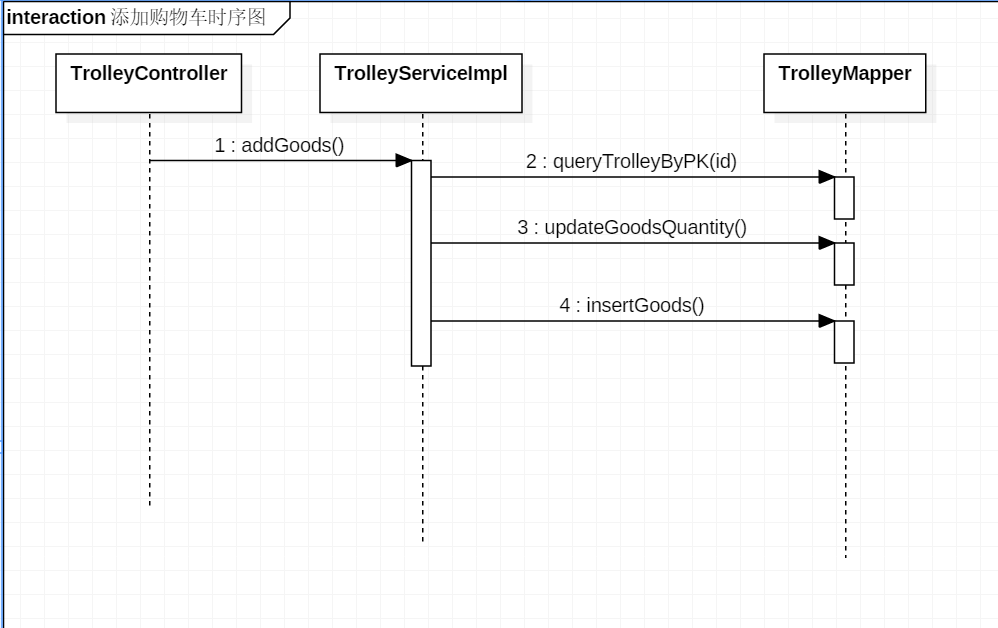


图4.6 添加购物车时序图

## 2.修改购物车内商品数量

实现：用户查看购物车内商品时，可以修改商品数量。增加商品数量时，会判断其数量是否超过库存，超过库存则增加失败。减少商品数量时则保证商品数量不为负。增减商品数量时都发送ajax请求到服务器，更新购物车内数据。

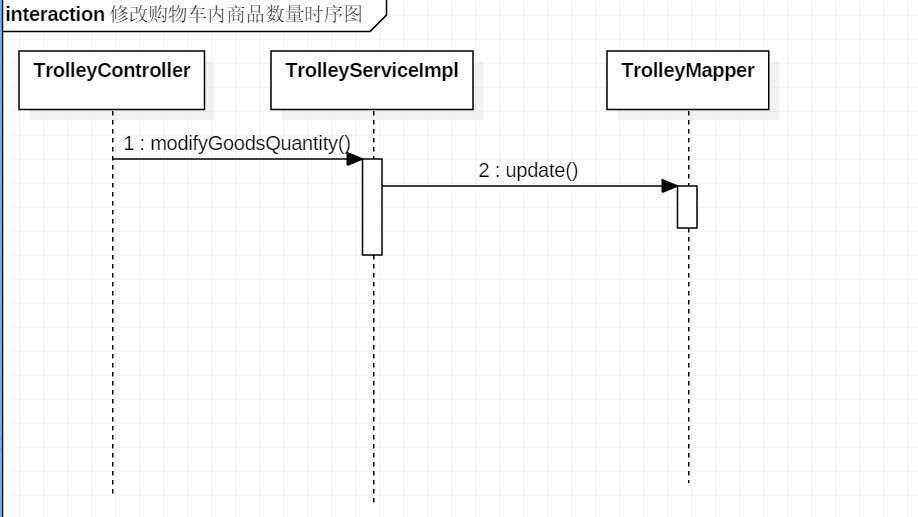


图4.7 修改购物车内商品数量时序图

## 3.删除购物车内商品

实现:用户可以直接删除购物车内某一商品。前台采用JQuery实现在页面动态删除，同时发送ajax请求到服务器端，根据传过来的商品id和存在session中的购物车id删除数据库中的数据。

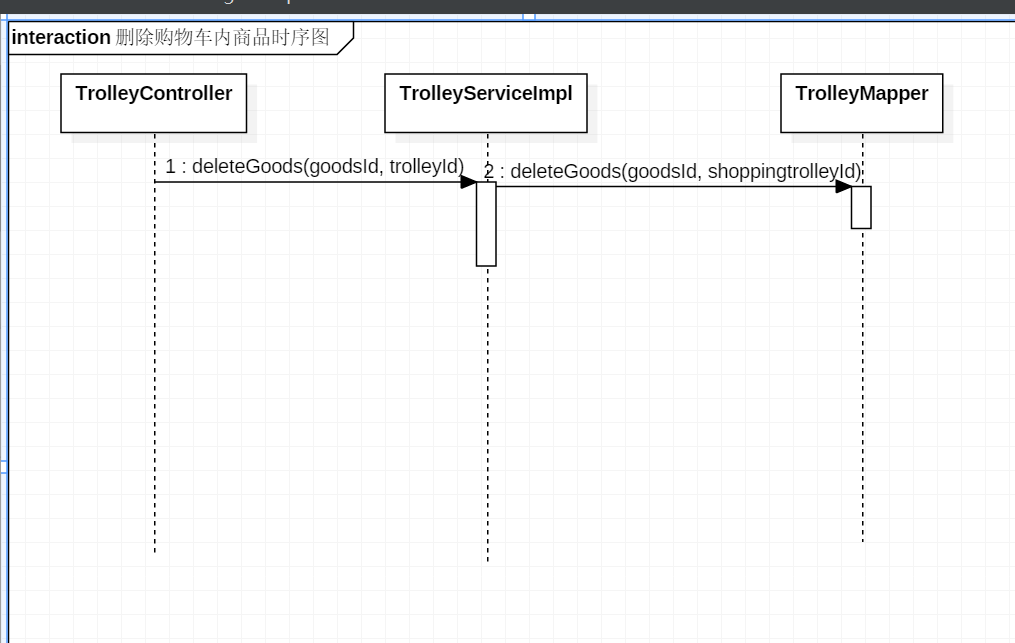


图4.8 删除购物车内商品时序图

# 4.4 订单模块

实现：根据用户从前台传过来的数据生成订单插入到数据库中，同时更新商品的库存。

图片包含 文字, 地图

已生成极高可信度的说明

图4.9 订单模块类图

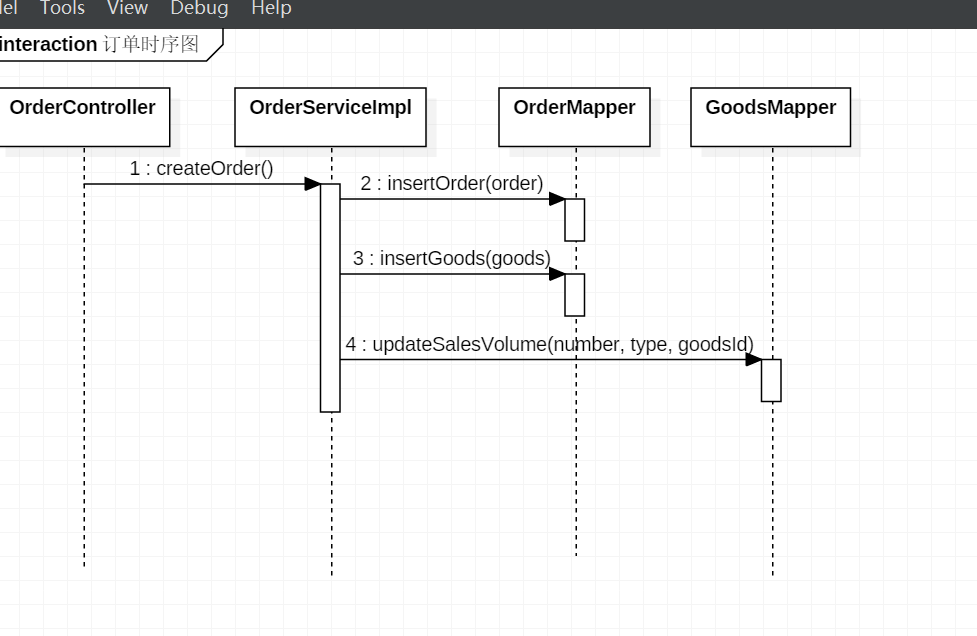


图4.12 订单模块类图

图4.9 订单模块类图

图4.10 订单模块时序图

# 4.5 评论模块

实现：每个注册用户可对每个商品评论一次。为了防止XSS攻击，在前台对用户的评论进行html转义后通过ajax请求发送到后台。后台首先判断该用户是否已经评论，若已经评论则无法再次评论，并发信息到前台提醒用户。接着对用户的数据进行过滤,判断其是否含有敏感词，含有敏感词的评论无法添加到数据库中，也无法显示在页面。过滤的实现如下：首先在数据库中查询出所有的敏感词，然后判断评论内容中是否含有该敏感词。为了提高访问速度，在第一次查询所有的敏感词后，即将其存在内存中，后续访问内存即可得到。前台根据服务器端传过来的数据进行操作。若评论含有敏感词或者已经评论，则无法显示在页面上。另外，未登录用户的评论会被拦截，不作处理，亦不显示在页面上。

图片包含 文字, 地图

已生成极高可信度的说明

图4.13 评论模块类图

图4.11 评论模块类图

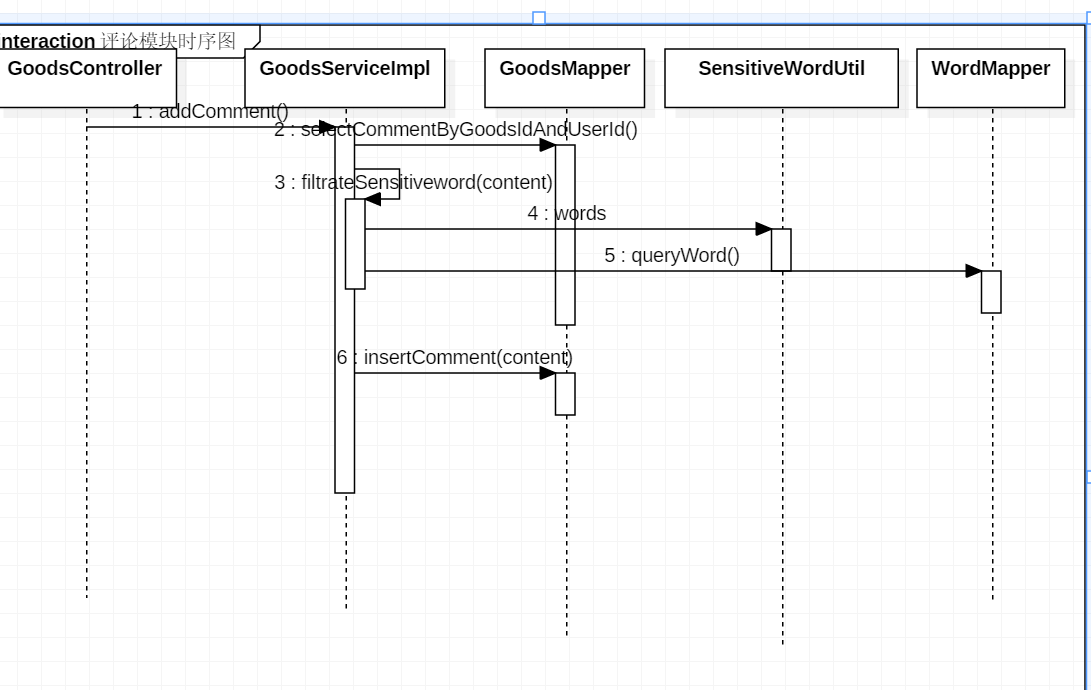


图4.12 评论模块时序图

第5章 测试

# 5.1 登录模块测试

用户登录时，若账号和密码输入错误，则会提示用户，并让用户重新登录。

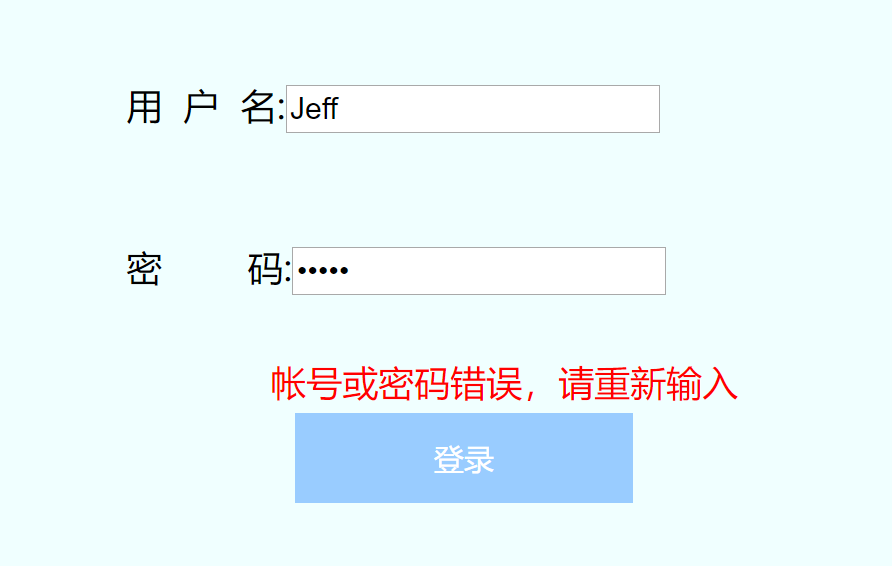


图5.1 帐号或密码输入错误

# 5.2 注册模块测试

注册时会检查用户输入的数据的合法性，如果数据格式不合要求或者为空，就会提醒用户，同时注册会失败。另外，当用户注册时会向服务器端发送Ajax请求验证用户名是否已被注册，保证用户名不重复。



图5.2 用户输入的数据为空，注册失败

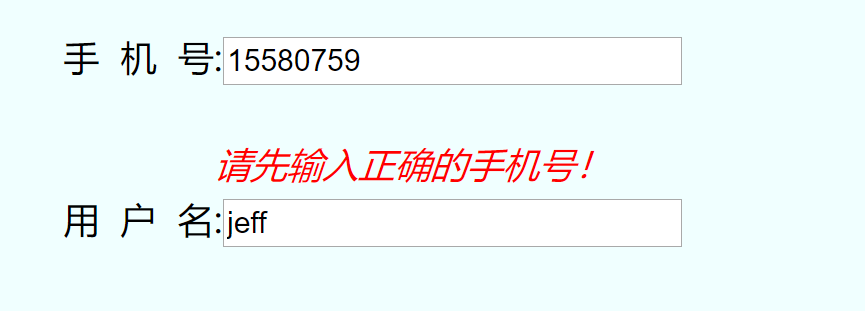


图5.3 用户输入的手机号不合要求

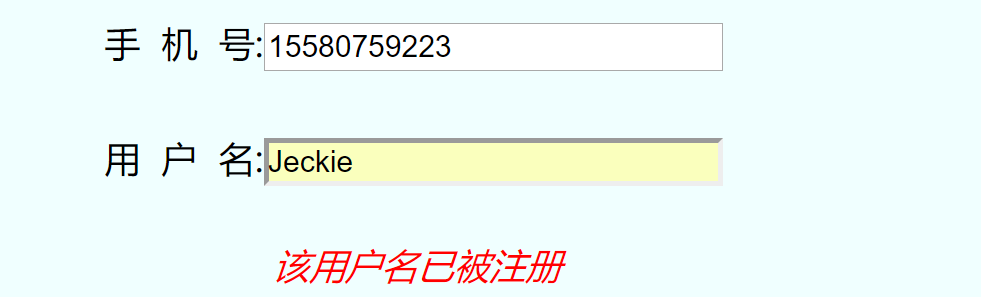


图5.4 用户名重复注册

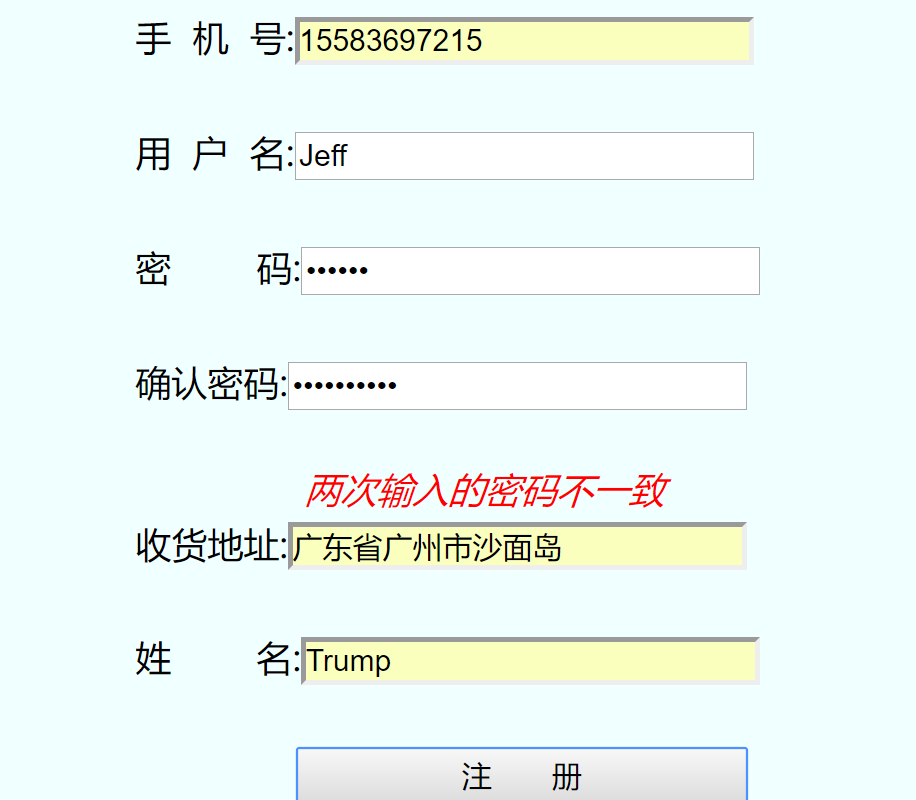


图5.5 两次输入的密码不一致

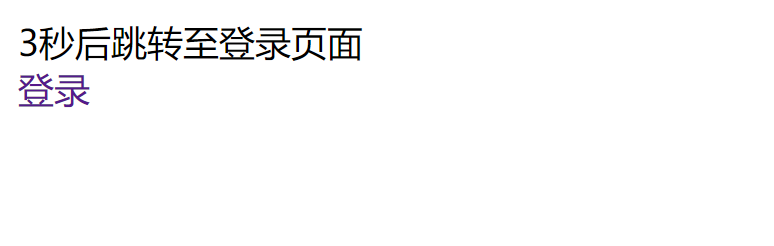


图5.6 注册成功，跳转至登录页面

第6章 系统后台设计

# 1.ssm框架的使用

## A.spring的使用

项目使用了spring框架，采用控制反转的思想，即IOC。让spring控制对象的生成，使用者不用关心对象的创建，用户可以直接从IOC容器中获取所需的对象。容器启动后会自动扫描组件，并会自动装配对象间的依赖关系。其次，使用spring控制Mybatis的sqlsession的生成，简化数据访问。

Spring配置:

<!-- 自动扫描除了Controller以外的其它组件，并進行依賴裝配 -->

<context:component-scan base-package="com.alibaba">

<context:exclude-filter type="annotation"

expression="org.springframework.stereotype.Controller" />

</context:component-scan>

<!-- 数据库访问参数 -->

<context:property-placeholder location="classpath:db.properties" />

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">

<property name="driverClass" value="${jdbc.driver}"></property>

<property name="jdbcUrl" value="${jdbc.url}"></property>

<property name="password" value="${jdbc.password}"></property>

<property name="user" value="${jdbc.username}"></property>

</bean>

<!-- 配置sqlSessionFactoryBean，用于生成sqlSession -->

<bean id="sqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource"></property>

<property name="configLocation" value="classpath:mybatis-config.xml"></property>

<property name="mapperLocations" value="classpath:com/alibaba/mapper/\*.xml"></property>

</bean>

<!--配置一个可以进行批量执行的sqlSession -->

<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">

<constructor-arg name="sqlSessionFactory" ref="sqlSessionFactoryBean"></constructor-arg>

<constructor-arg name="executorType" value="BATCH"></constructor-arg>

</bean>

<!-- 扫描Mybatis的\*mapper.xml文件 -->

<mybatis-spring:scan base-package="com.alibaba.mapper" />

## B.spring mvc的使用

Spring mvc主要处理前后台交互，接收前台发来的请求，同时将处理好的数据发送到前台。spring mvc的DispatcherServlet控制视图的转发，对视图进行渲染。另外，通过实现HandlerInterceptor接口可以自定义拦截器，对某些请求进行拦截。如用户没有权限，则请求不会被转发到controller。

详细配置：

<!-- 扫描控制器，生成对象 -->

<context:component-scan base-package="com.alibaba">

<context:include-filter type="annotation"

expression="org.springframework.stereotype.Controller" />

</context:component-scan>

<!-- 配置视图解析器 -->

<bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/jsp/"></property>

<property name="suffix" value=".jsp"></property>

</bean>

<mvc:annotation-driven></mvc:annotation-driven>

<!-- 处理静态资源 -->

<mvc:default-servlet-handler />

<!-- 配置拦截器，对请求进行拦截 -->

<mvc:interceptors>

<mvc:interceptor>

<mvc:mapping path="/Shoppingtrolley/addGoods" />

<mvc:mapping path="/Shoppingtrolley/queryTrolleyInfo" />

<mvc:mapping path="/goods/commentGoods" />

<mvc:mapping path="/Order/order" />

<bean class="com.alibaba.interceptor.UserInterceptor"></bean>

</mvc:interceptor>

</mvc:interceptors>

## C. spring和spring mvc在web环境下的使用

在web环境下使用spring和spring mvc，需配置spring的Listener和spring mvc的DispatcherServlet（控制视图转发）。

详细配置:

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:applicationContext.xml</param-value>

</context-param>

<!-- 配置ContextLoaderListener -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!--配置DispatcherServlet，处理请求并返回视图 -->

<servlet>

<servlet-name>springDispatcherServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:springmvc.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<!-- 映射所有的请求 -->

<servlet-mapping>

<servlet-name>springDispatcherServlet</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<!-- 字符编码过滤器 -->

<filter>

<filter-name>CharacterEncodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CharacterEncodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

## D. Mybatis的使用

Mybatis用于访问数据库，它封装了访问数据库的实现细节，用户只需要关心sql语句即可，不用管理各种dao 类，简化开发。Mybatis采用接口绑定，无需提供接口的实现，只需绑定接口与mapper.xml文件即可访问数据库。

示例:

<!-- 绑定接口,com.alibaba.mapper.UserMapper即为绑定的接口的全类名 -->

<mapper namespace="com.alibaba.mapper.UserMapper">

<!-- 绑定接口的方法，id即为方法名 -->

<!-- User selectUserByName(String userName) -->

<select id="selectUserByName" resultType="com.alibaba.entity.User">

select \* from user

where userName = #{userName}

</select>

<!-- public void insertUser(User user) -->

<insert id="insertUser">

INSERT INTO `user`(tel,userName,password,address,name)

VALUES (#{tel},#{userName}, #{password}, #{address},#{name})

</insert>

</mapper>

# 2.密码加密处理

为了保证用户的数据被泄漏，系统对用户的密码进行了加密处理。这样，即使数据泄露，用户的密码也不会泄露出去，保证了用户信息的安全。

系统采用Bcrypt加密算法（即加盐加密），对存入数据库的密码进行加密。进行密码比对的时候只需将用户输入的明文与数据库中的密文传给 方法即可，该方法会返回比对结果。

具体实现:

public void register(User user) {

//注册的时候对密码进行加密后添加到数据库

user.setPassword(BCrypt.hashpw(user.getPassword(), BCrypt.gensalt()));

mapper.insertUser(user);

User u = mapper.selectUserByName(user.getUserName());

ShoppingTrolley trolley = new ShoppingTrolley(1, u.getId());

// 创建购物车

trolleyMapper.createShoppingTrolley(trolley);

}

//登录时进行比对

public User login(String userName, String pwd) {

User user = mapper.selectUserByName(userName);

System.out.println("login:" + user);

// 比对用户输入的明文与数据库中的密文，比对成功则返回user信息，否则返回null

if (user != null && BCrypt.checkpw(pwd, user.getPassword())) {

return user;

}

return null;

}

# 3.敏感词

系统会对用户的评论进行过滤，如评论中含有敏感词，则不会被添加到数据库，评论失败。

敏感词库:

习近平

xjp

习明泽

老习

温宝宝

hujintao

温家宝

张培莉

胡boss

朱容基

李洪志

新疆骚乱

法轮功

大陆当局

北京当局

共贪党

公产党

产党共

阿共

供铲党

共x党

工产党

共产主义的幽灵

总 结

首先感谢老师的指导，这次实训老师很负责任，每天都会耐心检查我们的进度，还不时巡视指导我们。

其次，自己在这次实训中也得到了不少锻炼。学会使用Ajax发送异步请求，对于JQuery更加熟悉。同时也提高了处理集合的能力。不足的是对于前端网页设计不够熟练，对于网页设计不够熟悉。另外，对于JQuery的掌握不是很熟练，导致在某些问题上花了不少时间。而且，由于不会redis，无法让数据访问的效率得到提升。

在后面的学习中要锻炼自己设计网页的能力，加强自己对JQuery的熟练程度。同时也要加强自己的团队合作能力，学会用团队的力量解决问题，提升自己的团队协作能力和沟通能力。

参考文献

[1] 温谦编著. CSS网页设计标准教程[M]. 北京：人民邮电出版社，2009.

[2]

[3]

[4]

[5]

上面为参考文献的格式，请自己写参考书籍等。

附 录 主要源代码

# GoodsServiceImpl.java

package com.alibaba.service.impl;

import java.util.Arrays;

import java.util.Collections;

import java.util.Comparator;

import java.util.Iterator;

import java.util.LinkedList;

import java.util.List;

import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.alibaba.entity.Comment;

import com.alibaba.entity.Goods;

import com.alibaba.entity.GoodsDetail;

import com.alibaba.entity.GoodsImg;

import com.alibaba.entity.GoodsInfo;

import com.alibaba.entity.GoodsPrice;

import com.alibaba.entity.SensitiveWord;

import com.alibaba.entity.SortedGoodsInfo;

import com.alibaba.mapper.GoodsImgMapper;

import com.alibaba.mapper.GoodsMapper;

import com.alibaba.mapper.GoodsPriceMapper;

import com.alibaba.mapper.SensitiveWordMapper;

import com.alibaba.service.GoodsService;

import com.alibaba.util.GoodsUtil;

import com.alibaba.util.SensitiveWordUtil;

@Service

public class GoodsServiceImpl implements GoodsService {

@Autowired

private GoodsPriceMapper priceMapper;

@Autowired

private GoodsImgMapper imgMapper;

@Autowired

private SensitiveWordMapper wordMapper;

@Autowired

private GoodsMapper goodsMapper;

public GoodsDetail getGoodsDetail(int id) {

List<Goods> goods = goodsMapper.queryGoodsById(id);

List<GoodsImg> image = imgMapper.queryAllImg(id);

List<GoodsPrice> prcs = priceMapper.queryAllPrices(id);

for (GoodsPrice p : prcs) {

StringBuilder type = new StringBuilder(p.getType());

type.insert(1, "G");

type.insert(type.length(), "G");

p.setType(type.toString());

}

// PrintInfo(goods, image, prcs);

return new GoodsDetail(goods, image, prcs);

}

@Override

public GoodsInfo displayGoods() {

Set<GoodsPrice> prices;

Set<GoodsImg> img;

List<Goods> goods;

GoodsUtil goodsUtils = GoodsUtil.UTIL;

GoodsInfo goodsInfo = null;

prices = goodsUtils.getPrices();

img = goodsUtils.getImg();

goods = goodsUtils.getGoods();

if (goods != null && prices != null && img != null) {

goodsInfo = new GoodsInfo(prices, img, goods);

} else {

prices = priceMapper.queryGoodsPrice();

img = imgMapper.queryGoodsImg();

goods = goodsMapper.queryGoods();

goodsInfo = new GoodsInfo(prices, img, goods);

goodsUtils.setInfo(goodsInfo);

}

return goodsInfo;

}

@Override

public SortedGoodsInfo orderByPrice() {

// 获取商品信息

GoodsInfo info = displayGoods();

Set<GoodsPrice> prices = info.getPrices();

List<Goods> goods = info.getGoods();

Set<GoodsImg> image = info.getImg();

// 将Set转为List，用于排序

List<GoodsPrice> prcs = new LinkedList<GoodsPrice>(prices);

// 按照price升序排序

Collections.sort(prcs, new Comparator<GoodsPrice>() {

@Override

public int compare(GoodsPrice o1, GoodsPrice o2) {

if (o1.getPrice() > o2.getPrice()) {

return 1;

} else if (o1.getPrice() == o2.getPrice()) {

return 0;

}

return -1;

}

});

return moveElementsToNewCollection(goods, image, prcs);

}

public static void PrintInfo(Goods[] goodsArray, GoodsImg[] imgArray, List<GoodsPrice> prcs) {

System.out.println(" \n \n---------------Service PrintInfo begin-------------------------");

System.out.println("goods:");

for (Goods g : goodsArray) {

System.out.println(g);

}

System.out.println("\n\nImg:");

for (GoodsImg i : imgArray) {

System.out.println(i);

}

System.out.println("\n\nprice:");

for (GoodsPrice p : prcs) {

System.out.println(p);

}

System.out.println("--------------PrintInfo end-------------------------");

}

@Override

public SortedGoodsInfo orderBySalesVolume() {

// 获取商品信息

GoodsInfo info = displayGoods();

Set<GoodsPrice> prices = info.getPrices();

List<Goods> goods = info.getGoods();

Set<GoodsImg> image = info.getImg();

// 将Set转为List，用于排序

List<GoodsPrice> prcs = new LinkedList<GoodsPrice>(prices);

// 按照SlaesVolume升序排序

Collections.sort(prcs, new Comparator<GoodsPrice>() {

@Override

public int compare(GoodsPrice o1, GoodsPrice o2) {

if (o1.getSalesVolume() > o2.getSalesVolume()) {

return 1;

} else if (o1.getSalesVolume() == o2.getSalesVolume()) {

return 0;

}

return -1;

}

});

return moveElementsToNewCollection(goods, image, prcs);

}

public SortedGoodsInfo moveElementsToNewCollection(List<Goods> goods, Set<GoodsImg> image, List<GoodsPrice> prcs) {// 根据prcs对goods集合和image集合进行调整，使其goodsId的顺序一致

Goods[] goodsArray = new Goods[goods.size()];

GoodsImg[] imgArray = new GoodsImg[image.size()];

int goodsIndex = 0;

int imgIndex = 0;

for (GoodsPrice p : prcs) {

int id = p.getGoodsId();

// 将goods中的元素转到goodsArray中

for (Goods g : goods) {

if (g.getId() == id) {

goodsArray[goodsIndex++] = g;

}

}

// 将image中的元素转到imgArray中

for (GoodsImg i : image) {

if (i.getGoodsId() == id) {

imgArray[imgIndex++] = i;

}

}

}

image = null;// Let GC do it's work

goods = null;

List<Goods> g = Arrays.asList(goodsArray);

List<GoodsImg> img = Arrays.asList(imgArray);

// System.out.println("\n\nImgIndex:" + imgIndex + " goodsIndex:" + goodsIndex +

// "\n");

// PrintInfo(g, img, prcs);

// System.out.println(" \n \n---------------moveElementsToNewCollection

// end-------------------------");

return new SortedGoodsInfo(g, img, prcs);

}

@Override

public SortedGoodsInfo fuzzyQueryGoods(String param) {

// 根据字符串模糊查询商品

List<Goods> goods = goodsMapper.fuzzyQueryGoods(param);

// GoodsUtil goodsUtil = GoodsUtil.UTIL;

Set<GoodsPrice> prices = priceMapper.queryGoodsPrice();

Set<GoodsImg> img = imgMapper.queryGoodsImg();

// if (img == null) {

// img = imgMapper.queryGoodsImg();

//// goodsUtil.setImg(img);

// }

// if (prices == null) {

// prices = priceMapper.queryGoodsPrice();

//// goodsUtil.setPrices(prices);

// }

// 保留在goods中存在的商品

retain(prices, img, goods);

return moveElementsToNewCollection(goods, img, new LinkedList<GoodsPrice>(prices));

}

private void retain(Set<GoodsPrice> prices, Set<GoodsImg> img, List<Goods> goods) {

// fail-fast,应使用迭代器

// for (GoodsPrice p : prices) {

// int id = p.getGoodsId();

// boolean priceFlag = false;

// for (Goods g : goods) {

// if (g.getId() == id) {

// priceFlag = true;

// break;

// }

// }

// if (!priceFlag) {

// prices.remove(p);

// }

// }

Iterator<GoodsPrice> it = prices.iterator();

while (it.hasNext()) {

GoodsPrice p = it.next();

int id = p.getGoodsId();

boolean flag = false;

for (Goods g : goods) {

if (g.getId() == id) {

flag = true;

break;

}

}

if (!flag) {

it.remove();

}

}

Iterator<GoodsImg> imgIt = img.iterator();

while (imgIt.hasNext()) {

GoodsImg i = imgIt.next();

int id = i.getGoodsId();

boolean flag = false;

for (Goods g : goods) {

if (g.getId() == id) {

flag = true;

break;

}

}

if (!flag) {

imgIt.remove();

}

}

// for (GoodsImg i : img) {

// int id = i.getGoodsId();

// boolean flag = false;

// for (Goods g : goods) {

// if (g.getId() == id) {

// flag = true;

// break;

// }

// }

// if (!flag) {

// img.remove(i);

// }

// }

// PrintInfo(goods, img, prices);

}

@Override

public SortedGoodsInfo orderByPriceDecline() {

// 获取商品信息

GoodsInfo info = displayGoods();

Set<GoodsPrice> prices = info.getPrices();

List<Goods> goods = info.getGoods();

Set<GoodsImg> image = info.getImg();

// 将Set转为List，用于排序

List<GoodsPrice> prcs = new LinkedList<GoodsPrice>(prices);

// 按照price降序排序

Collections.sort(prcs, new Comparator<GoodsPrice>() {

@Override

public int compare(GoodsPrice o1, GoodsPrice o2) {

if (o1.getPrice() > o2.getPrice()) {

return -1;

} else if (o1.getPrice() == o2.getPrice()) {

return 0;

}

return 1;

}

});

return moveElementsToNewCollection(goods, image, prcs);

}

@Override

public SortedGoodsInfo orderBySalesVolumeDecline() {

// 获取商品信息

GoodsInfo info = displayGoods();

Set<GoodsPrice> prices = info.getPrices();

List<Goods> goods = info.getGoods();

Set<GoodsImg> image = info.getImg();

// 将Set转为List，用于排序

List<GoodsPrice> prcs = new LinkedList<GoodsPrice>(prices);

// 按照SlaesVolume降序排序

Collections.sort(prcs, new Comparator<GoodsPrice>() {

@Override

public int compare(GoodsPrice o1, GoodsPrice o2) {

if (o1.getSalesVolume() > o2.getSalesVolume()) {

return -1;

} else if (o1.getSalesVolume() == o2.getSalesVolume()) {

return 0;

}

return 1;

}

});

return moveElementsToNewCollection(goods, image, prcs);

}

// 1:包含敏感词

// -1:已经评论

// 0:OK

@Override

public int addComment(int goodsId, int userId, String content) {

// 判断是否已经评论

List<Comment> comments = goodsMapper.selectCommentByGoodsIdAndUserId(goodsId, userId);

System.out.println("\n\n\ncomments:" + comments);

if (null != comments && comments.size() > 0) {

return -1;

} else if (filtrateSensitiveword(content)) {

// 包含敏感词

return 1;

}

Comment c = new Comment(userId, goodsId, content);

if (content != null) {

goodsMapper.insertComment(c);

}

// 评论成功

return 0;

}

boolean filtrateSensitiveword(String content) {

List<SensitiveWord> words = SensitiveWordUtil.words;

if (words == null) {

words = wordMapper.queryWord();

SensitiveWordUtil.setWords(words);

}

if (content != null) {

for (SensitiveWord w : words) {

if (content.indexOf(w.getWord()) != -1) {

// 包含敏感词

return true;

}

}

}

// 不包含敏感词

return false;

}

}

# GoodsController.java

package com.alibaba.controller;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import java.util.Set;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.json.JSONException;

import org.json.JSONObject;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

import com.alibaba.entity.Comment;

import com.alibaba.entity.Goods;

import com.alibaba.entity.GoodsDetail;

import com.alibaba.entity.GoodsImg;

import com.alibaba.entity.GoodsInfo;

import com.alibaba.entity.GoodsPrice;

import com.alibaba.entity.SortedGoodsInfo;

import com.alibaba.entity.User;

import com.alibaba.mapper.GoodsMapper;

import com.alibaba.service.impl.GoodsServiceImpl;

@Controller

@RequestMapping("goods")

public class GoodsController {

@Autowired

private GoodsServiceImpl service;

@Autowired

private GoodsMapper mapper;

@RequestMapping("displayGoods")

public String displayGoods(HttpServletRequest request) {

HttpSession session = request.getSession();

GoodsInfo info = service.displayGoods();

List<GoodsPrice> prices = new LinkedList<GoodsPrice>(info.getPrices());

Set<GoodsImg> image = info.getImg();

List<Goods> goods = info.getGoods();

SortedGoodsInfo newInfo = service.moveElementsToNewCollection(goods, image, prices);

List<GoodsImg> img = newInfo.getImg();

prices = newInfo.getPrices();

goods = newInfo.getG();

session.setAttribute("prices", prices);

session.setAttribute("img", img);

session.setAttribute("goods", goods);

return "goods";

}

@RequestMapping("commentGoods")

@ResponseBody

public Map<String, String> commentGoods(@RequestBody String data, HttpServletRequest request) {

JSONObject jsonObj;

Map<String, String> map = new HashMap<>();

HttpSession session = request.getSession();

User user = (User) session.getAttribute("user");

try {

jsonObj = new JSONObject(data);

int goodsId = (int) jsonObj.get("goodsId");

String content = (String) jsonObj.get("content");

System.out.println("\n\n\ncontent:" + content);

System.out.println("\ngoodsId:" + goodsId);

if (null != user) {

int result = service.addComment(goodsId, user.getId(), content);

if (result == -1) {

map.put("msg", "您已经评论过了!!");

} else if (result == 1) {

map.put("msg", "Sensitiveword");

} else {

map.put("msg", "OK");

}

}

} catch (JSONException e) {

e.printStackTrace();

map.put("msg", "error");

}

return map;

}

@RequestMapping("goodsDetail")

public String goodsDetail(@RequestParam(value = "goodsId", required = false,defaultValue="1") int id, HttpServletRequest request,

HttpServletResponse response) {

service.getGoodsDetail(id);

HttpSession session = request.getSession();

GoodsDetail detail = service.getGoodsDetail(id);

List<GoodsPrice> price = detail.getPrices();

List<Integer> record = new ArrayList<>(price.size());

for (int i = 0; i < price.size(); i++) {

record.add(i);

}

List<GoodsImg> image = detail.getImage();

List<Integer> imgrecd = new ArrayList<>(image.size());

for (int i = 0; i < image.size(); i++) {

imgrecd.add(i);

}

session.setAttribute("goods", detail.getGoods());

session.setAttribute("image", image);

session.setAttribute("prices", price);

session.setAttribute("record", record);

session.setAttribute("recd", imgrecd);

List<Comment> comments = mapper.selectCommentByGoodsId(id);

session.setAttribute("comments", comments);

// browsHistory(request, response, id);

return "showGoods";

}

public void browsHistory(HttpServletRequest request, HttpServletResponse response, int id) {

System.out.println("\n\n\n-------browsHistory---------");

Cookie[] cookies = request.getCookies();

for (Cookie c : cookies) {

if (c != null && "history".equals(c.getName())) {

String val = c.getValue();

// System.out.println(val);

StringBuilder builder = new StringBuilder(val);

System.out.println("old builder:" + builder);

builder.append(",").append(String.valueOf(id));

System.out.println("new builder:" + builder);

Cookie cookie = new Cookie("history", builder.toString());

response.addCookie(cookie);

} else {

Cookie cookie = new Cookie("history", "" + id);

response.addCookie(cookie);

}

/\*

\* if (c != null) { System.out.println(c.getValue()); }

\*/

}

}

@RequestMapping("queryGoods")

public String queryGoods(HttpServletRequest request,

@RequestParam(value = "param", required = false) String param) {

SortedGoodsInfo info = service.fuzzyQueryGoods(param);

HttpSession session = request.getSession();

// List list = new ArrayList(new HashSet());

List<GoodsPrice> prices = new LinkedList<GoodsPrice>(info.getPrices());

session.setAttribute("prices", prices);

session.setAttribute("img", info.getImg());

session.setAttribute("goods", info.getG());

return "goods";

}

@RequestMapping("displayGoodsByPrices")

public String displayGoodsByPrices(HttpServletRequest request) {

SortedGoodsInfo info = service.orderByPrice();

HttpSession session = request.getSession();

saveInfoToSession(session, info);

return "goods";

}

@RequestMapping("displayGoodsBySalesVolume")

public String displayGoodsBySalesVolume(HttpServletRequest request) {

SortedGoodsInfo info = service.orderBySalesVolume();

HttpSession session = request.getSession();

saveInfoToSession(session, info);

return "goods";

}

@RequestMapping("displayGoodsByPricesDecline")

public String displayGoodsByPricesDecline(HttpServletRequest request) {

SortedGoodsInfo info = service.orderByPriceDecline();

HttpSession session = request.getSession();

saveInfoToSession(session, info);

return "goods";

}

@RequestMapping("displayGoodsBySalesVolumeDecline")

public String displayGoodsBySalesVolumeDecline(HttpServletRequest request) {

SortedGoodsInfo info = service.orderBySalesVolumeDecline();

HttpSession session = request.getSession();

saveInfoToSession(session, info);

return "goods";

}

private void saveInfoToSession(HttpSession session, SortedGoodsInfo info) {

session.setAttribute("prices", info.getPrices());

session.setAttribute("img", info.getImg());

session.setAttribute("goods", info.getG());

}

}

# UserController.java

package com.alibaba.controller;

import java.util.HashMap;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import java.util.Set;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.json.JSONException;

import org.json.JSONObject;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

import com.alibaba.entity.Goods;

import com.alibaba.entity.GoodsImg;

import com.alibaba.entity.GoodsInfo;

import com.alibaba.entity.GoodsPrice;

import com.alibaba.entity.ShoppingTrolley;

import com.alibaba.entity.SortedGoodsInfo;

import com.alibaba.entity.User;

import com.alibaba.mapper.ShoppingTrolleyMapper;

import com.alibaba.service.impl.GoodsServiceImpl;

import com.alibaba.service.impl.UserServiceImpl;

@Controller

@RequestMapping("User")

public class UserController {

@Autowired

private UserServiceImpl service;

@Autowired

private GoodsServiceImpl goodsService;

@Autowired

private ShoppingTrolleyMapper trolleyMapper;

@RequestMapping("register")

public String register(@RequestParam(value = "address", required = false) String address,

@RequestParam(value = "userName", required = false) String userName,

@RequestParam(value = "password", required = false) String password,

@RequestParam(value = "tel", required = false) String tel,

@RequestParam(value = "name", required = false) String name, HttpServletRequest request) {

int id = 0;

User user = new User(name, address, userName, password, tel, id);

service.register(user);

request.setAttribute("userName", userName);

return "registerSuccess";

}

@RequestMapping("regist")

public String register() {

// return "sendCode";

return "register";

}

@RequestMapping("loginSkip")

public String login() {

return "login";

}

@RequestMapping("sendCode")

@ResponseBody

public Map<String, String> sendValidateCode(@RequestBody String mobile) {

Map<String, String> result = new HashMap<String, String>();

result.put("code", "code");

System.out.println(result);

return result;

}

@RequestMapping("logout")

public String logout(HttpServletRequest request) {

request.getSession().invalidate();

return "goods";

}

@RequestMapping("validateUserName")

@ResponseBody

public Map<String, String> ifUserNameDuplicate(@RequestBody String name) {

Map<String, String> result = new HashMap<String, String>();

try {

JSONObject jsonobj = new JSONObject(name);

System.out.println(jsonobj.get("name").toString());

if (service.validateUserName(jsonobj.get("name").toString())) {

result.put("msg", "该用户名已被注册");

} else {

result.put("msg", "OK");

}

} catch (JSONException e) {

e.printStackTrace();

}

System.out.println("ifUserNameDuplicate\n name---->" + name + "\nresult:" + result);

return result;

}

@RequestMapping("login")

public String login(@RequestParam(value = "userName", required = false) String name,

@RequestParam(value = "password", required = false) String pwd, HttpServletRequest request) {

User user = service.login(name, pwd);

if (user != null) {

HttpSession session = request.getSession();

// 根据userId查询出购物车

List<ShoppingTrolley> trolley = trolleyMapper.queryShoppingTrolley(user.getId());

// System.out.println("\n\n\n----->查询出的购物车!!" + trolley + "\n\n\n");

if (trolley != null && trolley.size() > 0) {

// 将购物车id保存在session里

session.setAttribute("shoppingtrolleyId", trolley.get(0).getId());

// System.out.print("\n\n\nLogining... 购物车id:" +

// session.getAttribute("shoppingtrolleyId")+"\n\n");

}

// 保存用户信息

session.setAttribute("user", user);

GoodsInfo info = goodsService.displayGoods();

List<GoodsPrice> prices = new LinkedList<GoodsPrice>(info.getPrices());

Set<GoodsImg> image = info.getImg();

List<Goods> goods = info.getGoods();

SortedGoodsInfo newInfo = goodsService.moveElementsToNewCollection(goods, image, prices);

List<GoodsImg> img = newInfo.getImg();

prices = newInfo.getPrices();

goods = newInfo.getG();

// PrintInfo(goods, img, prices);

session.setAttribute("prices", prices);

session.setAttribute("img", img);

session.setAttribute("goods", goods);

// System.out.println("----->登录成功!!" + trolley.get(0).getId() + "\n\n\n");

return "goods";

}

request.setAttribute("errorMsg", "帐号或密码错误，请重新输入");

request.setAttribute("userName", name);

request.setAttribute("pwd", pwd);

return "login";

/\* return "forward:/index.jsp"; \*/

}

}

# UserServiceImpl.java

package com.alibaba.service.impl;

import org.mindrot.jbcrypt.BCrypt;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.alibaba.entity.ShoppingTrolley;

import com.alibaba.entity.User;

import com.alibaba.mapper.ShoppingTrolleyMapper;

import com.alibaba.mapper.UserMapper;

import com.alibaba.service.UserService;

@Service

public class UserServiceImpl implements UserService {

@Autowired

private UserMapper mapper;

@Autowired

private ShoppingTrolleyMapper trolleyMapper;

public User login(String userName, String pwd) {

User user = mapper.selectUserByName(userName);

System.out.println("login:" + user);

if (user != null && BCrypt.checkpw(pwd, user.getPassword())) {

return user;

}

return null;

}

public boolean validateUserName(String userName) {

if (mapper.selectUserByName(userName) != null) {

return true;// 用户名已被注册

}

return false;// 用户名未被注册

}

public void register(User user) {

System.out.println(user);

user.setPassword(BCrypt.hashpw(user.getPassword(), BCrypt.gensalt()));

mapper.insertUser(user);

User u = mapper.selectUserByName(user.getUserName());

ShoppingTrolley trolley = new ShoppingTrolley(1, u.getId());

// 创建购物车

trolleyMapper.createShoppingTrolley(trolley);

}

}

# OrderController.java

package com.alibaba.controller;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import com.alibaba.entity.User;

import com.alibaba.service.impl.OrderServiceImpl;

@Controller

@RequestMapping("Order")

public class OrderController {

@Autowired

private OrderServiceImpl service;

@RequestMapping("order")

public String order(HttpServletRequest request, @RequestParam(value = "price", required = false) double price,

@RequestParam(value = "type", required = false) String type,

@RequestParam(value = "color", required = false) String color,

@RequestParam(value = "number", required = false) int number,

@RequestParam(value = "rear", required = false) String rear,

@RequestParam(value = "front", required = false) String front,

@RequestParam(value = "capacity", required = false) String capacity,

@RequestParam(value = "goodsId", required = false) int goodsId) {

HttpSession session = request.getSession();

@SuppressWarnings("unused")

User user = (User) session.getAttribute("user");

session.setAttribute("number", number);

session.setAttribute("color", color);

session.setAttribute("type", type);

session.setAttribute("price", price);

session.setAttribute("front", front);

session.setAttribute("rear", rear);

session.setAttribute("capacity", capacity);

session.setAttribute("goodsId", goodsId);

return "order";

}

@RequestMapping("dealOrder")

public String dealOrder(HttpServletRequest request) {

HttpSession session = request.getSession();

User user = (User) session.getAttribute("user");

int number = (int) session.getAttribute("number");

String color = (String) session.getAttribute("color");

String type = (String) session.getAttribute("type");

double price = (double) session.getAttribute("price");

String front = (String) session.getAttribute("front");

String rear = (String) session.getAttribute("rear");

String capacity = (String) session.getAttribute("capacity");

int goodsId = (int) session.getAttribute("goodsId");

service.createOrder(price, type, color, number, rear, front, capacity, user, goodsId);

request.setAttribute("type", type);

request.setAttribute("color", color);

request.setAttribute("rear", rear);

request.setAttribute("front", front);

request.setAttribute("capacity", capacity);

request.setAttribute("number", number);

request.setAttribute("goodsId", goodsId);

return "orderCreated";

}

}

# OrderServiceImpl.java

package com.alibaba.service.impl;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.alibaba.entity.GoodsInOrder;

import com.alibaba.entity.Order;

import com.alibaba.entity.User;

import com.alibaba.mapper.GoodsMapper;

import com.alibaba.mapper.OrderMapper;

import com.alibaba.service.OrderService;

@Service

public class OrderServiceImpl implements OrderService {

@Autowired

private OrderMapper mapper;

@Autowired

private GoodsMapper goodsMapper;

@Override

public void createOrder(double price, String type, String color, int number, String rear, String front,

String capacity, User user, int goodsId) {

String name = null;

String address = null;

int userId = 0;

if (user != null) {

name = user.getName();

address = user.getAddress();

userId = user.getId();

}

DateTimeFormatter dateTimeformatter = DateTimeFormatter.ofPattern("YYYYMMddHHmmss");

LocalDateTime dateTime = LocalDateTime.now();

String time = dateTime.format(dateTimeformatter);

String id = "" + userId;

String str = "" + time.substring(time.length() - 4, time.length()) + id.substring(id.length() - 2, id.length());

int orderId = Integer.valueOf(str);

Order o = new Order(orderId, dateTime, address, name, number \* price, user.getId());

GoodsInOrder goods = new GoodsInOrder(orderId, goodsId, number, number \* price, capacity, front, rear, type);

mapper.insertOrder(o);

mapper.insertGoods(goods);

// 更新库存

goodsMapper.updateSalesVolume(number, type.replace("G", ""), goodsId);

}

}

# BCrypt算法

// Copyright (c) 2006 Damien Miller <djm@mindrot.org>

//

// Permission to use, copy, modify, and distribute this software for any

// purpose with or without fee is hereby granted, provided that the above

// copyright notice and this permission notice appear in all copies.

//

// THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES

// WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF

// MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR

// ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES

// WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN

// ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF

// OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

package org.mindrot.jbcrypt;

import java.io.UnsupportedEncodingException;

import java.security.SecureRandom;

/\*\*

\* BCrypt implements OpenBSD-style Blowfish password hashing using

\* the scheme described in "A Future-Adaptable Password Scheme" by

\* Niels Provos and David Mazieres.

\* <p>

\* This password hashing system tries to thwart off-line password

\* cracking using a computationally-intensive hashing algorithm,

\* based on Bruce Schneier's Blowfish cipher. The work factor of

\* the algorithm is parameterised, so it can be increased as

\* computers get faster.

\* <p>

\* Usage is really simple. To hash a password for the first time,

\* call the hashpw method with a random salt, like this:

\* <p>

\* <code>

\* String pw\_hash = BCrypt.hashpw(plain\_password, BCrypt.gensalt()); <br />

\* </code>

\* <p>

\* To check whether a plaintext password matches one that has been

\* hashed previously, use the checkpw method:

\* <p>

\* <code>

\* if (BCrypt.checkpw(candidate\_password, stored\_hash))<br />

\* &nbsp;&nbsp;&nbsp;&nbsp;System.out.println("It matches");<br />

\* else<br />

\* &nbsp;&nbsp;&nbsp;&nbsp;System.out.println("It does not match");<br />

\* </code>

\* <p>

\* The gensalt() method takes an optional parameter (log\_rounds)

\* that determines the computational complexity of the hashing:

\* <p>

\* <code>

\* String strong\_salt = BCrypt.gensalt(10)<br />

\* String stronger\_salt = BCrypt.gensalt(12)<br />

\* </code>

\* <p>

\* The amount of work increases exponentially (2\*\*log\_rounds), so

\* each increment is twice as much work. The default log\_rounds is

\* 10, and the valid range is 4 to 30.

\*

\* @author Damien Miller

\* @version 0.2

\*/

public class BCrypt {

// BCrypt parameters

private static final int GENSALT\_DEFAULT\_LOG2\_ROUNDS = 10;

private static final int BCRYPT\_SALT\_LEN = 16;

// Blowfish parameters

private static final int BLOWFISH\_NUM\_ROUNDS = 16;

// Initial contents of key schedule

private static final int P\_orig[] = {

0x243f6a88, 0x85a308d3, 0x13198a2e, 0x03707344,

0xa4093822, 0x299f31d0, 0x082efa98, 0xec4e6c89,

0x452821e6, 0x38d01377, 0xbe5466cf, 0x34e90c6c,

0xc0ac29b7, 0xc97c50dd, 0x3f84d5b5, 0xb5470917,

0x9216d5d9, 0x8979fb1b

};

private static final int S\_orig[] = {

0xd1310ba6, 0x98dfb5ac, 0x2ffd72db, 0xd01adfb7,

0xb8e1afed, 0x6a267e96, 0xba7c9045, 0xf12c7f99,

0x24a19947, 0xb3916cf7, 0x0801f2e2, 0x858efc16,

0x636920d8, 0x71574e69, 0xa458fea3, 0xf4933d7e,

0x0d95748f, 0x728eb658, 0x718bcd58, 0x82154aee,

0x7b54a41d, 0xc25a59b5, 0x9c30d539, 0x2af26013,

0xc5d1b023, 0x286085f0, 0xca417918, 0xb8db38ef,

0x8e79dcb0, 0x603a180e, 0x6c9e0e8b, 0xb01e8a3e,

0xd71577c1, 0xbd314b27, 0x78af2fda, 0x55605c60,

0xe65525f3, 0xaa55ab94, 0x57489862, 0x63e81440,

0x55ca396a, 0x2aab10b6, 0xb4cc5c34, 0x1141e8ce,

0xa15486af, 0x7c72e993, 0xb3ee1411, 0x636fbc2a,

0x2ba9c55d, 0x741831f6, 0xce5c3e16, 0x9b87931e,

0xafd6ba33, 0x6c24cf5c, 0x7a325381, 0x28958677,

0x3b8f4898, 0x6b4bb9af, 0xc4bfe81b, 0x66282193,

0x61d809cc, 0xfb21a991, 0x487cac60, 0x5dec8032,

0xef845d5d, 0xe98575b1, 0xdc262302, 0xeb651b88,

0x23893e81, 0xd396acc5, 0x0f6d6ff3, 0x83f44239,

0x2e0b4482, 0xa4842004, 0x69c8f04a, 0x9e1f9b5e,

0x21c66842, 0xf6e96c9a, 0x670c9c61, 0xabd388f0,

0x6a51a0d2, 0xd8542f68, 0x960fa728, 0xab5133a3,

0x6eef0b6c, 0x137a3be4, 0xba3bf050, 0x7efb2a98,

0xa1f1651d, 0x39af0176, 0x66ca593e, 0x82430e88,

0x8cee8619, 0x456f9fb4, 0x7d84a5c3, 0x3b8b5ebe,

0xe06f75d8, 0x85c12073, 0x401a449f, 0x56c16aa6,

0x4ed3aa62, 0x363f7706, 0x1bfedf72, 0x429b023d,

0x37d0d724, 0xd00a1248, 0xdb0fead3, 0x49f1c09b,

0x075372c9, 0x80991b7b, 0x25d479d8, 0xf6e8def7,

0xe3fe501a, 0xb6794c3b, 0x976ce0bd, 0x04c006ba,

0xc1a94fb6, 0x409f60c4, 0x5e5c9ec2, 0x196a2463,

0x68fb6faf, 0x3e6c53b5, 0x1339b2eb, 0x3b52ec6f,

0x6dfc511f, 0x9b30952c, 0xcc814544, 0xaf5ebd09,

0xbee3d004, 0xde334afd, 0x660f2807, 0x192e4bb3,

0xc0cba857, 0x45c8740f, 0xd20b5f39, 0xb9d3fbdb,

0x5579c0bd, 0x1a60320a, 0xd6a100c6, 0x402c7279,

0x679f25fe, 0xfb1fa3cc, 0x8ea5e9f8, 0xdb3222f8,

0x3c7516df, 0xfd616b15, 0x2f501ec8, 0xad0552ab,

0x323db5fa, 0xfd238760, 0x53317b48, 0x3e00df82,

0x9e5c57bb, 0xca6f8ca0, 0x1a87562e, 0xdf1769db,

0xd542a8f6, 0x287effc3, 0xac6732c6, 0x8c4f5573,

0x695b27b0, 0xbbca58c8, 0xe1ffa35d, 0xb8f011a0,

0x10fa3d98, 0xfd2183b8, 0x4afcb56c, 0x2dd1d35b,

0x9a53e479, 0xb6f84565, 0xd28e49bc, 0x4bfb9790,

0xe1ddf2da, 0xa4cb7e33, 0x62fb1341, 0xcee4c6e8,

0xef20cada, 0x36774c01, 0xd07e9efe, 0x2bf11fb4,

0x95dbda4d, 0xae909198, 0xeaad8e71, 0x6b93d5a0,

0xd08ed1d0, 0xafc725e0, 0x8e3c5b2f, 0x8e7594b7,

0x8ff6e2fb, 0xf2122b64, 0x8888b812, 0x900df01c,

0x4fad5ea0, 0x688fc31c, 0xd1cff191, 0xb3a8c1ad,

0x2f2f2218, 0xbe0e1777, 0xea752dfe, 0x8b021fa1,

0xe5a0cc0f, 0xb56f74e8, 0x18acf3d6, 0xce89e299,

0xb4a84fe0, 0xfd13e0b7, 0x7cc43b81, 0xd2ada8d9,

0x165fa266, 0x80957705, 0x93cc7314, 0x211a1477,

0xe6ad2065, 0x77b5fa86, 0xc75442f5, 0xfb9d35cf,

0xebcdaf0c, 0x7b3e89a0, 0xd6411bd3, 0xae1e7e49,

0x00250e2d, 0x2071b35e, 0x226800bb, 0x57b8e0af,

0x2464369b, 0xf009b91e, 0x5563911d, 0x59dfa6aa,

0x78c14389, 0xd95a537f, 0x207d5ba2, 0x02e5b9c5,

0x83260376, 0x6295cfa9, 0x11c81968, 0x4e734a41,

0xb3472dca, 0x7b14a94a, 0x1b510052, 0x9a532915,

0xd60f573f, 0xbc9bc6e4, 0x2b60a476, 0x81e67400,

0x08ba6fb5, 0x571be91f, 0xf296ec6b, 0x2a0dd915,

0xb6636521, 0xe7b9f9b6, 0xff34052e, 0xc5855664,

0x53b02d5d, 0xa99f8fa1, 0x08ba4799, 0x6e85076a,

0x4b7a70e9, 0xb5b32944, 0xdb75092e, 0xc4192623,

0xad6ea6b0, 0x49a7df7d, 0x9cee60b8, 0x8fedb266,

0xecaa8c71, 0x699a17ff, 0x5664526c, 0xc2b19ee1,

0x193602a5, 0x75094c29, 0xa0591340, 0xe4183a3e,

0x3f54989a, 0x5b429d65, 0x6b8fe4d6, 0x99f73fd6,

0xa1d29c07, 0xefe830f5, 0x4d2d38e6, 0xf0255dc1,

0x4cdd2086, 0x8470eb26, 0x6382e9c6, 0x021ecc5e,

0x09686b3f, 0x3ebaefc9, 0x3c971814, 0x6b6a70a1,

0x687f3584, 0x52a0e286, 0xb79c5305, 0xaa500737,

0x3e07841c, 0x7fdeae5c, 0x8e7d44ec, 0x5716f2b8,

0xb03ada37, 0xf0500c0d, 0xf01c1f04, 0x0200b3ff,

0xae0cf51a, 0x3cb574b2, 0x25837a58, 0xdc0921bd,

0xd19113f9, 0x7ca92ff6, 0x94324773, 0x22f54701,

0x3ae5e581, 0x37c2dadc, 0xc8b57634, 0x9af3dda7,

0xa9446146, 0x0fd0030e, 0xecc8c73e, 0xa4751e41,

0xe238cd99, 0x3bea0e2f, 0x3280bba1, 0x183eb331,

0x4e548b38, 0x4f6db908, 0x6f420d03, 0xf60a04bf,

0x2cb81290, 0x24977c79, 0x5679b072, 0xbcaf89af,

0xde9a771f, 0xd9930810, 0xb38bae12, 0xdccf3f2e,

0x5512721f, 0x2e6b7124, 0x501adde6, 0x9f84cd87,

0x7a584718, 0x7408da17, 0xbc9f9abc, 0xe94b7d8c,

0xec7aec3a, 0xdb851dfa, 0x63094366, 0xc464c3d2,

0xef1c1847, 0x3215d908, 0xdd433b37, 0x24c2ba16,

0x12a14d43, 0x2a65c451, 0x50940002, 0x133ae4dd,

0x71dff89e, 0x10314e55, 0x81ac77d6, 0x5f11199b,

0x043556f1, 0xd7a3c76b, 0x3c11183b, 0x5924a509,

0xf28fe6ed, 0x97f1fbfa, 0x9ebabf2c, 0x1e153c6e,

0x86e34570, 0xeae96fb1, 0x860e5e0a, 0x5a3e2ab3,

0x771fe71c, 0x4e3d06fa, 0x2965dcb9, 0x99e71d0f,

0x803e89d6, 0x5266c825, 0x2e4cc978, 0x9c10b36a,

0xc6150eba, 0x94e2ea78, 0xa5fc3c53, 0x1e0a2df4,

0xf2f74ea7, 0x361d2b3d, 0x1939260f, 0x19c27960,

0x5223a708, 0xf71312b6, 0xebadfe6e, 0xeac31f66,

0xe3bc4595, 0xa67bc883, 0xb17f37d1, 0x018cff28,

0xc332ddef, 0xbe6c5aa5, 0x65582185, 0x68ab9802,

0xeecea50f, 0xdb2f953b, 0x2aef7dad, 0x5b6e2f84,

0x1521b628, 0x29076170, 0xecdd4775, 0x619f1510,

0x13cca830, 0xeb61bd96, 0x0334fe1e, 0xaa0363cf,

0xb5735c90, 0x4c70a239, 0xd59e9e0b, 0xcbaade14,

0xeecc86bc, 0x60622ca7, 0x9cab5cab, 0xb2f3846e,

0x648b1eaf, 0x19bdf0ca, 0xa02369b9, 0x655abb50,

0x40685a32, 0x3c2ab4b3, 0x319ee9d5, 0xc021b8f7,

0x9b540b19, 0x875fa099, 0x95f7997e, 0x623d7da8,

0xf837889a, 0x97e32d77, 0x11ed935f, 0x16681281,

0x0e358829, 0xc7e61fd6, 0x96dedfa1, 0x7858ba99,

0x57f584a5, 0x1b227263, 0x9b83c3ff, 0x1ac24696,

0xcdb30aeb, 0x532e3054, 0x8fd948e4, 0x6dbc3128,

0x58ebf2ef, 0x34c6ffea, 0xfe28ed61, 0xee7c3c73,

0x5d4a14d9, 0xe864b7e3, 0x42105d14, 0x203e13e0,

0x45eee2b6, 0xa3aaabea, 0xdb6c4f15, 0xfacb4fd0,

0xc742f442, 0xef6abbb5, 0x654f3b1d, 0x41cd2105,

0xd81e799e, 0x86854dc7, 0xe44b476a, 0x3d816250,

0xcf62a1f2, 0x5b8d2646, 0xfc8883a0, 0xc1c7b6a3,

0x7f1524c3, 0x69cb7492, 0x47848a0b, 0x5692b285,

0x095bbf00, 0xad19489d, 0x1462b174, 0x23820e00,

0x58428d2a, 0x0c55f5ea, 0x1dadf43e, 0x233f7061,

0x3372f092, 0x8d937e41, 0xd65fecf1, 0x6c223bdb,

0x7cde3759, 0xcbee7460, 0x4085f2a7, 0xce77326e,

0xa6078084, 0x19f8509e, 0xe8efd855, 0x61d99735,

0xa969a7aa, 0xc50c06c2, 0x5a04abfc, 0x800bcadc,

0x9e447a2e, 0xc3453484, 0xfdd56705, 0x0e1e9ec9,

0xdb73dbd3, 0x105588cd, 0x675fda79, 0xe3674340,

0xc5c43465, 0x713e38d8, 0x3d28f89e, 0xf16dff20,

0x153e21e7, 0x8fb03d4a, 0xe6e39f2b, 0xdb83adf7,

0xe93d5a68, 0x948140f7, 0xf64c261c, 0x94692934,

0x411520f7, 0x7602d4f7, 0xbcf46b2e, 0xd4a20068,

0xd4082471, 0x3320f46a, 0x43b7d4b7, 0x500061af,

0x1e39f62e, 0x97244546, 0x14214f74, 0xbf8b8840,

0x4d95fc1d, 0x96b591af, 0x70f4ddd3, 0x66a02f45,

0xbfbc09ec, 0x03bd9785, 0x7fac6dd0, 0x31cb8504,

0x96eb27b3, 0x55fd3941, 0xda2547e6, 0xabca0a9a,

0x28507825, 0x530429f4, 0x0a2c86da, 0xe9b66dfb,

0x68dc1462, 0xd7486900, 0x680ec0a4, 0x27a18dee,

0x4f3ffea2, 0xe887ad8c, 0xb58ce006, 0x7af4d6b6,

0xaace1e7c, 0xd3375fec, 0xce78a399, 0x406b2a42,

0x20fe9e35, 0xd9f385b9, 0xee39d7ab, 0x3b124e8b,

0x1dc9faf7, 0x4b6d1856, 0x26a36631, 0xeae397b2,

0x3a6efa74, 0xdd5b4332, 0x6841e7f7, 0xca7820fb,

0xfb0af54e, 0xd8feb397, 0x454056ac, 0xba489527,

0x55533a3a, 0x20838d87, 0xfe6ba9b7, 0xd096954b,

0x55a867bc, 0xa1159a58, 0xcca92963, 0x99e1db33,

0xa62a4a56, 0x3f3125f9, 0x5ef47e1c, 0x9029317c,

0xfdf8e802, 0x04272f70, 0x80bb155c, 0x05282ce3,

0x95c11548, 0xe4c66d22, 0x48c1133f, 0xc70f86dc,

0x07f9c9ee, 0x41041f0f, 0x404779a4, 0x5d886e17,

0x325f51eb, 0xd59bc0d1, 0xf2bcc18f, 0x41113564,

0x257b7834, 0x602a9c60, 0xdff8e8a3, 0x1f636c1b,

0x0e12b4c2, 0x02e1329e, 0xaf664fd1, 0xcad18115,

0x6b2395e0, 0x333e92e1, 0x3b240b62, 0xeebeb922,

0x85b2a20e, 0xe6ba0d99, 0xde720c8c, 0x2da2f728,

0xd0127845, 0x95b794fd, 0x647d0862, 0xe7ccf5f0,

0x5449a36f, 0x877d48fa, 0xc39dfd27, 0xf33e8d1e,

0x0a476341, 0x992eff74, 0x3a6f6eab, 0xf4f8fd37,

0xa812dc60, 0xa1ebddf8, 0x991be14c, 0xdb6e6b0d,

0xc67b5510, 0x6d672c37, 0x2765d43b, 0xdcd0e804,

0xf1290dc7, 0xcc00ffa3, 0xb5390f92, 0x690fed0b,

0x667b9ffb, 0xcedb7d9c, 0xa091cf0b, 0xd9155ea3,

0xbb132f88, 0x515bad24, 0x7b9479bf, 0x763bd6eb,

0x37392eb3, 0xcc115979, 0x8026e297, 0xf42e312d,

0x6842ada7, 0xc66a2b3b, 0x12754ccc, 0x782ef11c,

0x6a124237, 0xb79251e7, 0x06a1bbe6, 0x4bfb6350,

0x1a6b1018, 0x11caedfa, 0x3d25bdd8, 0xe2e1c3c9,

0x44421659, 0x0a121386, 0xd90cec6e, 0xd5abea2a,

0x64af674e, 0xda86a85f, 0xbebfe988, 0x64e4c3fe,

0x9dbc8057, 0xf0f7c086, 0x60787bf8, 0x6003604d,

0xd1fd8346, 0xf6381fb0, 0x7745ae04, 0xd736fccc,

0x83426b33, 0xf01eab71, 0xb0804187, 0x3c005e5f,

0x77a057be, 0xbde8ae24, 0x55464299, 0xbf582e61,

0x4e58f48f, 0xf2ddfda2, 0xf474ef38, 0x8789bdc2,

0x5366f9c3, 0xc8b38e74, 0xb475f255, 0x46fcd9b9,

0x7aeb2661, 0x8b1ddf84, 0x846a0e79, 0x915f95e2,

0x466e598e, 0x20b45770, 0x8cd55591, 0xc902de4c,

0xb90bace1, 0xbb8205d0, 0x11a86248, 0x7574a99e,

0xb77f19b6, 0xe0a9dc09, 0x662d09a1, 0xc4324633,

0xe85a1f02, 0x09f0be8c, 0x4a99a025, 0x1d6efe10,

0x1ab93d1d, 0x0ba5a4df, 0xa186f20f, 0x2868f169,

0xdcb7da83, 0x573906fe, 0xa1e2ce9b, 0x4fcd7f52,

0x50115e01, 0xa70683fa, 0xa002b5c4, 0x0de6d027,

0x9af88c27, 0x773f8641, 0xc3604c06, 0x61a806b5,

0xf0177a28, 0xc0f586e0, 0x006058aa, 0x30dc7d62,

0x11e69ed7, 0x2338ea63, 0x53c2dd94, 0xc2c21634,

0xbbcbee56, 0x90bcb6de, 0xebfc7da1, 0xce591d76,

0x6f05e409, 0x4b7c0188, 0x39720a3d, 0x7c927c24,

0x86e3725f, 0x724d9db9, 0x1ac15bb4, 0xd39eb8fc,

0xed545578, 0x08fca5b5, 0xd83d7cd3, 0x4dad0fc4,

0x1e50ef5e, 0xb161e6f8, 0xa28514d9, 0x6c51133c,

0x6fd5c7e7, 0x56e14ec4, 0x362abfce, 0xddc6c837,

0xd79a3234, 0x92638212, 0x670efa8e, 0x406000e0,

0x3a39ce37, 0xd3faf5cf, 0xabc27737, 0x5ac52d1b,

0x5cb0679e, 0x4fa33742, 0xd3822740, 0x99bc9bbe,

0xd5118e9d, 0xbf0f7315, 0xd62d1c7e, 0xc700c47b,

0xb78c1b6b, 0x21a19045, 0xb26eb1be, 0x6a366eb4,

0x5748ab2f, 0xbc946e79, 0xc6a376d2, 0x6549c2c8,

0x530ff8ee, 0x468dde7d, 0xd5730a1d, 0x4cd04dc6,

0x2939bbdb, 0xa9ba4650, 0xac9526e8, 0xbe5ee304,

0xa1fad5f0, 0x6a2d519a, 0x63ef8ce2, 0x9a86ee22,

0xc089c2b8, 0x43242ef6, 0xa51e03aa, 0x9cf2d0a4,

0x83c061ba, 0x9be96a4d, 0x8fe51550, 0xba645bd6,

0x2826a2f9, 0xa73a3ae1, 0x4ba99586, 0xef5562e9,

0xc72fefd3, 0xf752f7da, 0x3f046f69, 0x77fa0a59,

0x80e4a915, 0x87b08601, 0x9b09e6ad, 0x3b3ee593,

0xe990fd5a, 0x9e34d797, 0x2cf0b7d9, 0x022b8b51,

0x96d5ac3a, 0x017da67d, 0xd1cf3ed6, 0x7c7d2d28,

0x1f9f25cf, 0xadf2b89b, 0x5ad6b472, 0x5a88f54c,

0xe029ac71, 0xe019a5e6, 0x47b0acfd, 0xed93fa9b,

0xe8d3c48d, 0x283b57cc, 0xf8d56629, 0x79132e28,

0x785f0191, 0xed756055, 0xf7960e44, 0xe3d35e8c,

0x15056dd4, 0x88f46dba, 0x03a16125, 0x0564f0bd,

0xc3eb9e15, 0x3c9057a2, 0x97271aec, 0xa93a072a,

0x1b3f6d9b, 0x1e6321f5, 0xf59c66fb, 0x26dcf319,

0x7533d928, 0xb155fdf5, 0x03563482, 0x8aba3cbb,

0x28517711, 0xc20ad9f8, 0xabcc5167, 0xccad925f,

0x4de81751, 0x3830dc8e, 0x379d5862, 0x9320f991,

0xea7a90c2, 0xfb3e7bce, 0x5121ce64, 0x774fbe32,

0xa8b6e37e, 0xc3293d46, 0x48de5369, 0x6413e680,

0xa2ae0810, 0xdd6db224, 0x69852dfd, 0x09072166,

0xb39a460a, 0x6445c0dd, 0x586cdecf, 0x1c20c8ae,

0x5bbef7dd, 0x1b588d40, 0xccd2017f, 0x6bb4e3bb,

0xdda26a7e, 0x3a59ff45, 0x3e350a44, 0xbcb4cdd5,

0x72eacea8, 0xfa6484bb, 0x8d6612ae, 0xbf3c6f47,

0xd29be463, 0x542f5d9e, 0xaec2771b, 0xf64e6370,

0x740e0d8d, 0xe75b1357, 0xf8721671, 0xaf537d5d,

0x4040cb08, 0x4eb4e2cc, 0x34d2466a, 0x0115af84,

0xe1b00428, 0x95983a1d, 0x06b89fb4, 0xce6ea048,

0x6f3f3b82, 0x3520ab82, 0x011a1d4b, 0x277227f8,

0x611560b1, 0xe7933fdc, 0xbb3a792b, 0x344525bd,

0xa08839e1, 0x51ce794b, 0x2f32c9b7, 0xa01fbac9,

0xe01cc87e, 0xbcc7d1f6, 0xcf0111c3, 0xa1e8aac7,

0x1a908749, 0xd44fbd9a, 0xd0dadecb, 0xd50ada38,

0x0339c32a, 0xc6913667, 0x8df9317c, 0xe0b12b4f,

0xf79e59b7, 0x43f5bb3a, 0xf2d519ff, 0x27d9459c,

0xbf97222c, 0x15e6fc2a, 0x0f91fc71, 0x9b941525,

0xfae59361, 0xceb69ceb, 0xc2a86459, 0x12baa8d1,

0xb6c1075e, 0xe3056a0c, 0x10d25065, 0xcb03a442,

0xe0ec6e0e, 0x1698db3b, 0x4c98a0be, 0x3278e964,

0x9f1f9532, 0xe0d392df, 0xd3a0342b, 0x8971f21e,

0x1b0a7441, 0x4ba3348c, 0xc5be7120, 0xc37632d8,

0xdf359f8d, 0x9b992f2e, 0xe60b6f47, 0x0fe3f11d,

0xe54cda54, 0x1edad891, 0xce6279cf, 0xcd3e7e6f,

0x1618b166, 0xfd2c1d05, 0x848fd2c5, 0xf6fb2299,

0xf523f357, 0xa6327623, 0x93a83531, 0x56cccd02,

0xacf08162, 0x5a75ebb5, 0x6e163697, 0x88d273cc,

0xde966292, 0x81b949d0, 0x4c50901b, 0x71c65614,

0xe6c6c7bd, 0x327a140a, 0x45e1d006, 0xc3f27b9a,

0xc9aa53fd, 0x62a80f00, 0xbb25bfe2, 0x35bdd2f6,

0x71126905, 0xb2040222, 0xb6cbcf7c, 0xcd769c2b,

0x53113ec0, 0x1640e3d3, 0x38abbd60, 0x2547adf0,

0xba38209c, 0xf746ce76, 0x77afa1c5, 0x20756060,

0x85cbfe4e, 0x8ae88dd8, 0x7aaaf9b0, 0x4cf9aa7e,

0x1948c25c, 0x02fb8a8c, 0x01c36ae4, 0xd6ebe1f9,

0x90d4f869, 0xa65cdea0, 0x3f09252d, 0xc208e69f,

0xb74e6132, 0xce77e25b, 0x578fdfe3, 0x3ac372e6

};

// bcrypt IV: "OrpheanBeholderScryDoubt". The C implementation calls

// this "ciphertext", but it is really plaintext or an IV. We keep

// the name to make code comparison easier.

static private final int bf\_crypt\_ciphertext[] = {

0x4f727068, 0x65616e42, 0x65686f6c,

0x64657253, 0x63727944, 0x6f756274

};

// Table for Base64 encoding

static private final char base64\_code[] = {

'.', '/', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',

'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',

'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',

'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',

'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5',

'6', '7', '8', '9'

};

// Table for Base64 decoding

static private final byte index\_64[] = {

-1, -1, -1, -1, -1, -1, -1, -1, -1, -1,

-1, -1, -1, -1, -1, -1, -1, -1, -1, -1,

-1, -1, -1, -1, -1, -1, -1, -1, -1, -1,

-1, -1, -1, -1, -1, -1, -1, -1, -1, -1,

-1, -1, -1, -1, -1, -1, 0, 1, 54, 55,

56, 57, 58, 59, 60, 61, 62, 63, -1, -1,

-1, -1, -1, -1, -1, 2, 3, 4, 5, 6,

7, 8, 9, 10, 11, 12, 13, 14, 15, 16,

17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,

-1, -1, -1, -1, -1, -1, 28, 29, 30,

31, 32, 33, 34, 35, 36, 37, 38, 39, 40,

41, 42, 43, 44, 45, 46, 47, 48, 49, 50,

51, 52, 53, -1, -1, -1, -1, -1

};

// Expanded Blowfish key

private int P[];

private int S[];

/\*\*

\* Encode a byte array using bcrypt's slightly-modified base64

\* encoding scheme. Note that this is \*not\* compatible with

\* the standard MIME-base64 encoding.

\*

\* @param d the byte array to encode

\* @param len the number of bytes to encode

\* @return base64-encoded string

\* @exception IllegalArgumentException if the length is invalid

\*/

private static String encode\_base64(byte d[], int len)

throws IllegalArgumentException {

int off = 0;

StringBuffer rs = new StringBuffer();

int c1, c2;

if (len <= 0 || len > d.length)

throw new IllegalArgumentException ("Invalid len");

while (off < len) {

c1 = d[off++] & 0xff;

rs.append(base64\_code[(c1 >> 2) & 0x3f]);

c1 = (c1 & 0x03) << 4;

if (off >= len) {

rs.append(base64\_code[c1 & 0x3f]);

break;

}

c2 = d[off++] & 0xff;

c1 |= (c2 >> 4) & 0x0f;

rs.append(base64\_code[c1 & 0x3f]);

c1 = (c2 & 0x0f) << 2;

if (off >= len) {

rs.append(base64\_code[c1 & 0x3f]);

break;

}

c2 = d[off++] & 0xff;

c1 |= (c2 >> 6) & 0x03;

rs.append(base64\_code[c1 & 0x3f]);

rs.append(base64\_code[c2 & 0x3f]);

}

return rs.toString();

}

/\*\*

\* Look up the 3 bits base64-encoded by the specified character,

\* range-checking againt conversion table

\* @param x the base64-encoded value

\* @return the decoded value of x

\*/

private static byte char64(char x) {

if ((int)x < 0 || (int)x > index\_64.length)

return -1;

return index\_64[(int)x];

}

/\*\*

\* Decode a string encoded using bcrypt's base64 scheme to a

\* byte array. Note that this is \*not\* compatible with

\* the standard MIME-base64 encoding.

\* @param s the string to decode

\* @param maxolen the maximum number of bytes to decode

\* @return an array containing the decoded bytes

\* @throws IllegalArgumentException if maxolen is invalid

\*/

private static byte[] decode\_base64(String s, int maxolen)

throws IllegalArgumentException {

StringBuffer rs = new StringBuffer();

int off = 0, slen = s.length(), olen = 0;

byte ret[];

byte c1, c2, c3, c4, o;

if (maxolen <= 0)

throw new IllegalArgumentException ("Invalid maxolen");

while (off < slen - 1 && olen < maxolen) {

c1 = char64(s.charAt(off++));

c2 = char64(s.charAt(off++));

if (c1 == -1 || c2 == -1)

break;

o = (byte)(c1 << 2);

o |= (c2 & 0x30) >> 4;

rs.append((char)o);

if (++olen >= maxolen || off >= slen)

break;

c3 = char64(s.charAt(off++));

if (c3 == -1)

break;

o = (byte)((c2 & 0x0f) << 4);

o |= (c3 & 0x3c) >> 2;

rs.append((char)o);

if (++olen >= maxolen || off >= slen)

break;

c4 = char64(s.charAt(off++));

o = (byte)((c3 & 0x03) << 6);

o |= c4;

rs.append((char)o);

++olen;

}

ret = new byte[olen];

for (off = 0; off < olen; off++)

ret[off] = (byte)rs.charAt(off);

return ret;

}

/\*\*

\* Blowfish encipher a single 64-bit block encoded as

\* two 32-bit halves

\* @param lr an array containing the two 32-bit half blocks

\* @param off the position in the array of the blocks

\*/

private final void encipher(int lr[], int off) {

int i, n, l = lr[off], r = lr[off + 1];

l ^= P[0];

for (i = 0; i <= BLOWFISH\_NUM\_ROUNDS - 2;) {

// Feistel substitution on left word

n = S[(l >> 24) & 0xff];

n += S[0x100 | ((l >> 16) & 0xff)];

n ^= S[0x200 | ((l >> 8) & 0xff)];

n += S[0x300 | (l & 0xff)];

r ^= n ^ P[++i];

// Feistel substitution on right word

n = S[(r >> 24) & 0xff];

n += S[0x100 | ((r >> 16) & 0xff)];

n ^= S[0x200 | ((r >> 8) & 0xff)];

n += S[0x300 | (r & 0xff)];

l ^= n ^ P[++i];

}

lr[off] = r ^ P[BLOWFISH\_NUM\_ROUNDS + 1];

lr[off + 1] = l;

}

/\*\*

\* Cycically extract a word of key material

\* @param data the string to extract the data from

\* @param offp a "pointer" (as a one-entry array) to the

\* current offset into data

\* @return the next word of material from data

\*/

private static int streamtoword(byte data[], int offp[]) {

int i;

int word = 0;

int off = offp[0];

for (i = 0; i < 4; i++) {

word = (word << 8) | (data[off] & 0xff);

off = (off + 1) % data.length;

}

offp[0] = off;

return word;

}

/\*\*

\* Initialise the Blowfish key schedule

\*/

private void init\_key() {

P = (int[])P\_orig.clone();

S = (int[])S\_orig.clone();

}

/\*\*

\* Key the Blowfish cipher

\* @param key an array containing the key

\*/

private void key(byte key[]) {

int i;

int koffp[] = { 0 };

int lr[] = { 0, 0 };

int plen = P.length, slen = S.length;

for (i = 0; i < plen; i++)

P[i] = P[i] ^ streamtoword(key, koffp);

for (i = 0; i < plen; i += 2) {

encipher(lr, 0);

P[i] = lr[0];

P[i + 1] = lr[1];

}

for (i = 0; i < slen; i += 2) {

encipher(lr, 0);

S[i] = lr[0];

S[i + 1] = lr[1];

}

}

/\*\*

\* Perform the "enhanced key schedule" step described by

\* Provos and Mazieres in "A Future-Adaptable Password Scheme"

\* http://www.openbsd.org/papers/bcrypt-paper.ps

\* @param data salt information

\* @param key password information

\*/

private void ekskey(byte data[], byte key[]) {

int i;

int koffp[] = { 0 }, doffp[] = { 0 };

int lr[] = { 0, 0 };

int plen = P.length, slen = S.length;

for (i = 0; i < plen; i++)

P[i] = P[i] ^ streamtoword(key, koffp);

for (i = 0; i < plen; i += 2) {

lr[0] ^= streamtoword(data, doffp);

lr[1] ^= streamtoword(data, doffp);

encipher(lr, 0);

P[i] = lr[0];

P[i + 1] = lr[1];

}

for (i = 0; i < slen; i += 2) {

lr[0] ^= streamtoword(data, doffp);

lr[1] ^= streamtoword(data, doffp);

encipher(lr, 0);

S[i] = lr[0];

S[i + 1] = lr[1];

}

}

/\*\*

\* Perform the central password hashing step in the

\* bcrypt scheme

\* @param password the password to hash

\* @param salt the binary salt to hash with the password

\* @param log\_rounds the binary logarithm of the number

\* of rounds of hashing to apply

\* @param cdata the plaintext to encrypt

\* @return an array containing the binary hashed password

\*/

public byte[] crypt\_raw(byte password[], byte salt[], int log\_rounds,

int cdata[]) {

int rounds, i, j;

int clen = cdata.length;

byte ret[];

if (log\_rounds < 4 || log\_rounds > 30)

throw new IllegalArgumentException ("Bad number of rounds");

rounds = 1 << log\_rounds;

if (salt.length != BCRYPT\_SALT\_LEN)

throw new IllegalArgumentException ("Bad salt length");

init\_key();

ekskey(salt, password);

for (i = 0; i != rounds; i++) {

key(password);

key(salt);

}

for (i = 0; i < 64; i++) {

for (j = 0; j < (clen >> 1); j++)

encipher(cdata, j << 1);

}

ret = new byte[clen \* 4];

for (i = 0, j = 0; i < clen; i++) {

ret[j++] = (byte)((cdata[i] >> 24) & 0xff);

ret[j++] = (byte)((cdata[i] >> 16) & 0xff);

ret[j++] = (byte)((cdata[i] >> 8) & 0xff);

ret[j++] = (byte)(cdata[i] & 0xff);

}

return ret;

}

/\*\*

\* Hash a password using the OpenBSD bcrypt scheme

\* @param password the password to hash

\* @param salt the salt to hash with (perhaps generated

\* using BCrypt.gensalt)

\* @return the hashed password

\*/

public static String hashpw(String password, String salt) {

BCrypt B;

String real\_salt;

byte passwordb[], saltb[], hashed[];

char minor = (char)0;

int rounds, off = 0;

StringBuffer rs = new StringBuffer();

if (salt.charAt(0) != '$' || salt.charAt(1) != '2')

throw new IllegalArgumentException ("Invalid salt version");

if (salt.charAt(2) == '$')

off = 3;

else {

minor = salt.charAt(2);

if (minor != 'a' || salt.charAt(3) != '$')

throw new IllegalArgumentException ("Invalid salt revision");

off = 4;

}

// Extract number of rounds

if (salt.charAt(off + 2) > '$')

throw new IllegalArgumentException ("Missing salt rounds");

rounds = Integer.parseInt(salt.substring(off, off + 2));

real\_salt = salt.substring(off + 3, off + 25);

try {

passwordb = (password + (minor >= 'a' ? "\000" : "")).getBytes("UTF-8");

} catch (UnsupportedEncodingException uee) {

throw new AssertionError("UTF-8 is not supported");

}

saltb = decode\_base64(real\_salt, BCRYPT\_SALT\_LEN);

B = new BCrypt();

hashed = B.crypt\_raw(passwordb, saltb, rounds,

(int[])bf\_crypt\_ciphertext.clone());

rs.append("$2");

if (minor >= 'a')

rs.append(minor);

rs.append("$");

if (rounds < 10)

rs.append("0");

if (rounds > 30) {

throw new IllegalArgumentException(

"rounds exceeds maximum (30)");

}

rs.append(Integer.toString(rounds));

rs.append("$");

rs.append(encode\_base64(saltb, saltb.length));

rs.append(encode\_base64(hashed,

bf\_crypt\_ciphertext.length \* 4 - 1));

return rs.toString();

}

/\*\*

\* Generate a salt for use with the BCrypt.hashpw() method

\* @param log\_rounds the log2 of the number of rounds of

\* hashing to apply - the work factor therefore increases as

\* 2\*\*log\_rounds.

\* @param random an instance of SecureRandom to use

\* @return an encoded salt value

\*/

public static String gensalt(int log\_rounds, SecureRandom random) {

StringBuffer rs = new StringBuffer();

byte rnd[] = new byte[BCRYPT\_SALT\_LEN];

random.nextBytes(rnd);

rs.append("$2a$");

if (log\_rounds < 10)

rs.append("0");

if (log\_rounds > 30) {

throw new IllegalArgumentException(

"log\_rounds exceeds maximum (30)");

}

rs.append(Integer.toString(log\_rounds));

rs.append("$");

rs.append(encode\_base64(rnd, rnd.length));

return rs.toString();

}

/\*\*

\* Generate a salt for use with the BCrypt.hashpw() method

\* @param log\_rounds the log2 of the number of rounds of

\* hashing to apply - the work factor therefore increases as

\* 2\*\*log\_rounds.

\* @return an encoded salt value

\*/

public static String gensalt(int log\_rounds) {

return gensalt(log\_rounds, new SecureRandom());

}

/\*\*

\* Generate a salt for use with the BCrypt.hashpw() method,

\* selecting a reasonable default for the number of hashing

\* rounds to apply

\* @return an encoded salt value

\*/

public static String gensalt() {

return gensalt(GENSALT\_DEFAULT\_LOG2\_ROUNDS);

}

/\*\*

\* Check that a plaintext password matches a previously hashed

\* one

\* @param plaintext the plaintext password to verify

\* @param hashed the previously-hashed password

\* @return true if the passwords match, false otherwise

\*/

public static boolean checkpw(String plaintext, String hashed) {

byte hashed\_bytes[];

byte try\_bytes[];

try {

String try\_pw = hashpw(plaintext, hashed);

hashed\_bytes = hashed.getBytes("UTF-8");

try\_bytes = try\_pw.getBytes("UTF-8");

} catch (UnsupportedEncodingException uee) {

return false;

}

if (hashed\_bytes.length != try\_bytes.length)

return false;

byte ret = 0;

for (int i = 0; i < try\_bytes.length; i++)

ret |= hashed\_bytes[i] ^ try\_bytes[i];

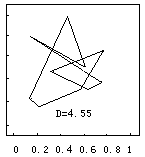
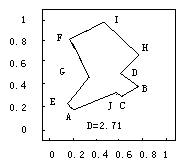
return ret == 0;

}

}

**图和表的格式参考**

图例编号为：章数.图序号



（a）始值加噪声时 （b）用离散Hopfield网络

产生的访问路径 求解得的访问路径

图1.2 10城市TSP计算机模拟结果**表示例**

表1.1 主要技术经济指标表

表编号为：

章数.表序号

不要让表头、说明与表分成2页

| 序号 | 名 称 | | 单 位 | 数 量 | 备 注 |
| --- | --- | --- | --- | --- | --- |
| 1 | 设计吞吐量 | | 万吨/年 | 25.5 | 粮食12万吨/年，杂货13.5万吨/年 |
| 2 | 泊位数 | | 个 | 2 | —— |
| 3 | 码头总长度 | | m | 120 | —— |
| 4 | 陆域纵深 | | m | 307 | —— |
| 5 | 陆域面积 | | 万m2 | 5.73 | —— |
| 6 | 水域  面积 | 码头前停泊水域 | 万m2 | 0.58 | 船舶停靠及装卸作业水域 |
| 回旋水域 | 万m2 | 1.63 | —— |

（资料来源：《中国船舶》，2002年第6期，第20页）

表1.2 各组分lgBi值

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | T=1500K | | T=2000K | |
| 组分 | lgBi | 组分 | lgBi |
| 1 | O2+ | 5.26 | HO2 | 6.43 |
| 2 | HO2 | 5.26 | O2+ | 6.42 |
| 3 | H2O+ | 4.76 | H2O+ | 6.18 |

注：“+”表示重要成分，“\*”表示冗余成分。（资料来源： 许成章：《船舶先进技术》，船舶工业出版社，2002.）

表1.3 压降损失计算结果  **Pa**

|  |
| --- |
| 换热器 热边压降损失 冷边压降损失 |
| 初级 2974.37 2931.52  次级 2924.65 3798.76 |

（资料来源：http://www.moe.edu.cn/edoas/website18/info8378.htm;）

此处插入了分页符，保证每章都另起一页