**西安电子科技大学研究生学位论文**

**撰写要求 (2015年修订版)**

**作者姓名 张 三**

**指导教师姓名、职称 李 四 选择**

**申请学位类别 选择一项**

**西安电子科技大学研究生学位论文**

**撰写要求 (2015年修订版)**

**作者姓名：**张 三

**一级学科：**选择学科

**二级学科（研究方向）：**选择学科

**学位类别：**选择学位

**指导教师姓名、职称：**李 四 职称

**学　　院：** 选择学院

**提交日期：**选取日期

**西安电子科技大学**

**选择类型论文**

**学　号** 1101110071

**密　级　 选择密级**

**学校代码 10701**

**分类号** TN82

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of selecting one

in selecting one

By

Zhang San

Supervisor: Li Si Title: Professor

February 2015

**Thesis/Dissertation Guide for Postgraduates**

**of XIDIAN UNIVERSITY**

**西安电子科技大学**

**学位论文独创性（或创新性）声明**

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同事对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文若有不实之处，本人承担一切法律责任。

本人签名： 日 期：

**西安电子科技大学**

**关于论文使用授权的说明**

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权属于西安电子科技大学。学校有权保留送交论文的复印件，允许查阅、借阅论文；学校可以公布论文的全部或部分内容，允许采用影印、缩印或其它复制手段保存论文。同时本人保证，结合学位论文研究成果完成的论文、发明专利等成果，署名单位为西安电子科技大学。

保密的学位论文在 年解密后适用本授权书。

本人签名： 导师签名：

日 期： 日 期：

摘要

摘要是学位论文的内容不加注释和评论的简短陈述，简明扼要陈述学位论文的研究目的、内容、方法、成果和结论，重点突出学位论文的创造性成果和观点。摘要包括中文摘要和英文摘要，硕士学位论文中文摘要字数一般为1000字左右，博士学位论文中文摘要字数一般为1500字左右。英文摘要内容与中文摘要内容保持一致，翻译力求简明精准。摘要的正文下方需注明论文的关键词，关键词一般为3～8个，关键词和关键词之间用逗号并空一格。

中文摘要格式要求为：宋体小四、两端对齐、首行缩进2字符，行距为固定值20磅，段落间距为段前0磅，段后0磅。

英文摘要格式要求为：Times New Roman、小四、两端对齐、首行不缩进，行距为固定值20磅，段落间距为段前0磅，段后0磅，段与段之间空一行。

**关 键 词**：XXX， XXX， XXX， XXX， XXX

ABSTRACT

The Abstract is a brief description of a thesis or dissertation without notes or comments. It represents concisely the research purpose, content, method, result and conclusion of the thesis or dissertation with emphasis on its innovative findings and perspectives. The Abstract Part consists of both the Chinese abstract and the English abstract. The Chinese abstract should have the length of approximately 1000 Chinese characters for a master thesis and 1500 for a Ph.D. dissertation. The English abstract should be consistent with the Chinese one in content. The keywords of a thesis or dissertation should be listed below the main body of the abstract, separated by commas and a space. The number of the keywords is typically 3 to 5.

The format of the Chinese Abstract is what follows: Song Ti, Small 4, justified, 2 characters indented in the first line, line spacing at a fixed value of 20 pounds, and paragraph spacing section at 0 pound.

The format of the English Abstract is what follows: Times New Roman, Small 4, justified, not indented in the first line, line spacing at a fixed value of 20 pounds, and paragraph spacing section at 0 pound with a blank line between paragraphs.

**Keywords**: XXX, XXX, XXX, XXX, XXX

插图索引

图序号 插图示例 X

表格索引

表格序号 表格示例 X

符号对照表

符号 符号名称

XXX XXX

XXX XXX

XXX XXX

缩略语对照表

缩略语 英文全称 中文对照

XXX XXX XXX

XXX XXX XXX

XXX XXX XXX

目录

[摘要 I](#_Toc413704332)

[ABSTRACT III](#_Toc413704333)

[插图索引 V](#_Toc413704334)

[表格索引 VII](#_Toc413704335)

[符号对照表 IX](#_Toc413704336)

[缩略语对照表 XI](#_Toc413704337)

[第一章 研究生学位论文撰写的总体要求 1](#_Toc413704338)

[第二章 研究生学位论文撰写的内容要求 3](#_Toc413704339)

[2.1 封面 3](#_Toc413704340)

[2.2 题名页 3](#_Toc413704341)

[2.3 声明 5](#_Toc413704342)

[2.4 摘要 5](#_Toc413704343)

[2.5 插图索引 5](#_Toc413704344)

[2.6 表格索引 6](#_Toc413704345)

[2.7 符号对照表 6](#_Toc413704346)

[2.8 缩略语对照表 6](#_Toc413704347)

[2.9 目录 6](#_Toc413704348)

[2.10 正文 6](#_Toc413704349)

[2.10.1 绪论 7](#_Toc413704350)

[2.10.2 各章节 7](#_Toc413704351)

[2.10.3 结论 8](#_Toc413704352)

[2.11 参考文献 8](#_Toc413704353)

[2.12 致谢 9](#_Toc413704354)

[2.13 作者简介 9](#_Toc413704355)

[2.14 其他 10](#_Toc413704356)

[第三章 研究生学位论文的编辑、打印、装订要求 11](#_Toc413704357)

[3.1 学位论文封面的编辑和打印要求 11](#_Toc413704358)

[3.2 学位论文的版面设置要求 11](#_Toc413704359)

[3.3 学位论文的打印、装订要求 11](#_Toc413704360)

[3.4 其他说明 11](#_Toc413704361)

[第四章 图、表、公式示例 13](#_Toc413704362)

[参考文献 15](#_Toc413704363)

[致谢 19](#_Toc413704364)

[作者简介 21](#_Toc413704365)

# 绪论

随着互联网和计算机的快速发展，零售行业信息化程度都迅速提高，企业对客户信息资源的管理与维护需求也随之骤增并且对于会员客户的合理化营销也对企业的发展产生了重要的影响。由于企业对于客户信息管理的规模总量巨大，在未来的企业信息化的发展过程中，务必会出现规模越来越大的营销需求，业务逻辑也会越来越复杂，所以我们需要建立完善的自动化营销体系，完成数据收集、客户营销的一站式服务。

企业营销系统可以帮助企业建立自有的数据中心、汇集全渠道的客户数据。通过多种渠道将客户数据汇集到营销系统，解决信息孤岛的问题。并且利用数据管理审查数据完整度，便于营销人员分析，通过策划营销活动，逐步完善补足数据。最为重要的是，将用户数据进行筛选后，利用微信、邮件、短信等渠道进行自动的个性化营销，实现多渠道联动，多条件触发，减少人为操作，大幅度提高工作效率。

## 论文的背景及意义

在当今世界，营销成为了一个企业推广自己所必不可少的方式。数字营销是指通过数字渠道（例如搜索引擎，网站，电子邮件和微信短信息）投放的广告。数字营销就是使用这些渠道，推广公司的商品，服务和品牌的方法。同时，如今消费者严重依赖数字营销手段来了解产品。例如，《Think with Google》的营销报告发现，49％的消费者是在搜索引擎上开始查询的，而33％的消费者是通过电子邮件的推广了解产品，而26％的用户是在移动应用程序中推送来了解商品的。

想要实现现代数字营销就意味着营销人员必须在其品牌上加入的庞大的营销管理系统，。为了发挥数字营销的真正潜力，营销系统的开发必须深入研究当今广阔而复杂的跨渠道世界，以发现通过互动营销产生的影响。互动营销就是收集相关的客户数据并且联系其与潜在客户之间的关系的一种营销手段，通过在数字环境中吸引客户，可以建立品牌知名度，成为行业领导羊，并且让客户在有意向购买该产品时第一个就会想到这个品牌。

对于数字营销的未来，我们未来可以看到随着可供消费者使用的可穿戴设备的数量持续增加，电子邮件营销将变得更加个性化，企业的营销也更加的精准。

## 国内外现状分析

### 国外研究现状

企业营销管理系统是企业信息管理系统的重要组成部分，因为零售行业的主要活动就是产品营销。西方发达国家的信息化建设起步比较， 企业信息系统建设相对比较完善，作为特殊行业的零售营销管理系统的建设在功能上和技术上都比较先进,，许多零售行业营销管理系统有效的结合CRM、 SCM，很大程度上提高了企业的综合竞争力[1]。除此之外，市场上也有许多知名的产品营销管理软件生产商，例如美国的Orade公司、德国的Sap公司。

另外，由于新冠疫情肆虐，导致人们更多的网上购物。因此越来越多的中小企业选择设计并使用企业营销管理系统，通过更大资金投入并且不断开拓创新相关的技术，如多线程高并发，微服务架构等来完善企业营销管理系统。使得企业不仅能够将潜在用户的信息保留在自己的数据库中，并且可以通过像诸如大数据等手段分析出相应的报告对他们进行精准营销。并且经济增长缓慢，企业不能再单纯依靠规模经济的优势来站稳脚跟，继续投入大量的资金来招募大量的员工和有才华的人才来刺激企业的成长不再有效。然而，通过大量的高新技术来对企业进行数字化转型带来的提升却非常显著，这样就会在无形之中大幅度提升自己的利润。

### 国内研究现状

相比之下，国内的企业信息化建设起步比较晚，零售行业营销管理系统相对比较落后。随着对企业信息化重要性的认识加深，目前，国内许多零售行业大型公司已经建设完成或者正在筹建营销管理系统。已经有许多非常成功的例子，但是也出现一些失败的现象，因为一些企业只是盲目跟风，没有考虑本企业的实际需要，导致建设的营销管理系统不符合本企业的需要，根本无法为企业带来利润。除大中型企业外，大部分小型企业的营销管理系统还是采用传统的工作模式，人工记录、分析、统计数据，这种数据处理方式,不仅效率低下，而且一旦出现错误，很难检查出来，错误数据日积月累，很难保证统计数据的正确性[3]。纸质保存的数据很难进行查阅，也无法进行快速比较，而且容易损坏，不易备份数据。企业仍然采用传统营销管理模式的原因有企业负责人没有意识到产品营销管理信息化建设的重要性；短时间内无法体现信息化建设的价值；企业信息化建设需要投入大量的人力物力力求通过企业信息化建设来促进企业的快速发展,实现利益最大化

同时，中小企业在我国公司占比之大，并且在我国经济发展中占据有重要地位，并且横跨多个领域，分布在中国社会生活的方方面面，因此每年都有非常多的新型小公司诞生，也有非常多的公司因为数字化程度不够或者其他原因导致竞争力不足，而在市场上销声匿迹。就这样，中小企业之间的竞争一定是往着越来越激烈的趋势变化的，只有将数字化科技应用于企业之中，不仅能提高企业的效率，提升企业的利润，更加重要的是，还能够提升人民的生活水平，各个企业通过激烈的数字化竞争，可以产生出更加成熟的软件产品，并且这个产品通过多次迭代和演变，可以更加迎合用户的口味。因此，我们的企业营销系统必定要采用当今最流行的分布式架构来设计，同时必须具备可以生产上线的能力，如果企业需要，就可以部署运行。

并且，一个良好的企业营销系统必须可以适应一个企业多变的营销策略，这样可以带来非常多的好处。它可以用来击败竞争对手并同时提高销售额和利润。我们都知道，如此多的企业在相同产品和服务上竞争。如何在其他方面脱颖而出将在很大程度上取决于营销策略的有效性，精心计划的数字营销策略将帮助品牌在与竞品相比之中彰显优势，真正建立品牌，并最终成为市场中的“大”品牌。但是，如今由于中国大多数公司的数字化水平较低，还不能应用上很多国外成熟的营销技术，并且提升数字化水平的手段有限，需要花费大量的时间。

## 论文工作内容

本文主要研究企业营销系统的设计和开发过程，包括当前数字营销的必要性以及企业营销系统在数字营销中所扮演的重要角色，国内外对企业营销系统应用的现状，实现一个与时俱进的企业营销系统的主要技术介绍（设计常用的多线程高并发技术，大数据搜索技术等），系统的各部分模块（例如营销模版模块，电子邮件、短信发送模块以及客户信息聚合模块等），系统各模块的业务需求分析以及模型，实现系统开发过程中用到的一些算法，设计框架中的源代码知识可以更好的结合业务逻辑来提高相应的响应效率等等。

首先调研了国内外企业营销系统的现状，并且据此分析国内的营销系统的需求，整理出相应的需求模型进行分析。对相应的需求模型设计相应的实现方案，例如微服务，前后端分离，分布式锁，高并发，算法等等，然后翻阅国外的技术网站查看相应的技术实现从而能够更好的将技术应用到项目中去。最后，按照需求开发相应的模块，并且使用单元测试技术和WebDriver等自动化测试技术对最后的项目进行测试。同时，本项目就是为了上线而打造的，所以相应的微服务部署都是经过周密考虑，在配置文件中也区分开发环境和生产环境，方便直接部署到容器Docker中去。在论文末尾会有详细的配置列表。

## 论文组织结构

论文一共有七个章节，各个章节的主要内容：

第一章：绪论。包括论文选题、对国内外现状相应系统和数字营销进行深度分析、论文工作内容和结构组织。

第二章:主要技术栈介绍。主要对本项目实现中运用的技术进行介绍，例如多线程，CAS, ReentrantLock,Vue.js, Spin Locks,前后端分离，鉴权，LomBok, Spring Cloud, Elasticsearch, Redis, RabbitMQ, Nginx, Docker等等技术进行介绍，并且通过对源码的分析来介绍为什么这些技术适合企业营销管理系统

第三章:主要数据结构与算法介绍。由于这个项目将来可能直接上线，所以代码质量需要尽可能优化，降低时间复杂度，本章详细介绍项目中会用到的用来提升效率的几种数据结构与算法，如ConcurrentHashMap, BinarySearchTree, DFS BFS剪枝等。

第三章：对企业营销管理系统的需求分析。通过实地在企业实习，以及对使用用户的走访调查总结出相应的需求，并通过用例图，时序图，类图等的方式将他们展示出来，方面后期开发能够对需求精准把控。

第四章: 由于现在企业的开发都是敏捷开发，本论文为了高度模拟企业开发流程，故引入了敏捷开发使用jenkins等敏捷开发组件实现CI\DI持续集成，方便部署上线，本章将对持续集成进行深入分析

第五章：对企业营销管理系统的实现，通过层层剖析前端MVVM的工作原理以及后端RestFUL api的相应作用以及负载均衡等功能，同时结合数据库中存储的数据库表之间字段以及elasticsearch中相应的搜索字段解析来达到对项目的深度剖析

第六章：对企业营销管理系统的测试，本项目开发中采用Junit和Swagger Mock来进行单元测试，上线后采用Webdriver作为测试工具，方便jenkins自动集成后自动测试,本章节主要对测试工具进行分析，以及分析为什么企业中广泛采取这种测试方式。

第七章：结束语。概括本论文的工作方向，以及相应的改进地方。

# 相关技术介绍

主要技术栈介绍。主要对本项目实现中运用的技术进行介绍，例如多线程，CAS, ReentrantLock,Vue.js, Spin Locks,前后端分离，鉴权，LomBok, Spring Cloud, Elasticsearch, Redis, RabbitMQ, Nginx, Docker等等技术进行介绍，并且通过对源码的分析来介绍为什么这些技术适合企业营销管理系统

## Spring框架

Spring框架是Java平台的应用程序框架，并且具有inversion of control的作用，用于管理应用中常用的Bean。该框架的核心功能可以被任何Java应用程序使用，但是在Java EE（企业版）平台之上有一些用于构建Web应用程序的扩展。尽管该框架没有要求任何特定的编程模型，但它已作为Java EJB企业模型的补充在Java社区中流行。Spring框架是开源的。

要灵活运用spring的ioc就得先彻底了解它底层的工作原理，这样将会对后来的调试和运用带来很大的便利，下面将对spring的源码进行分析以方便，想要彻底了解spring框架就得先从循环依赖注入开始分析：

1.以单例的实例化来介绍spring的IOC

以某个自己需要初始化的singleton为例，通过层层嵌套运行到doCreateBean()方法中，

boolean earlySingletonExposure = mbd.isSingleton() && this.allowCircularReferences && this.isSingletonCurrentlyInCreation(beanName);

会在判断这行代码是否为true以判断是否开启spring的循环依赖，默认是开启的，可以通过修改源代码或者Spring的BeanFactory提供的API来关闭循环依赖，判断是开启循环依赖后，然后运行以下方法

this.addSingletonFactory(beanName, () -> {

return this.getEarlyBeanReference(beanName, mbd, bean);

});

这里使用了工厂模式

就可以将此还没有完全创建好的Bean对象加入到第三级缓存中(三级缓存可以查看下面的专门说明),这个第三级缓存会在有某bean需要循环注入时，从factory中得到此earlySingleton加入BeanFactory的earlySingletonObjects中方便后面再继续使用循环依赖，一旦循环调用结束，这个earlySingletonObject通过registerSingleton就进入到singleObjects(一个专门存储已经走完所用BeanDefinition生命周期的Bean的Map集合，一级缓存)，总流程概览：

IndexService webService互为聚合关系，且都通过@Autowired自动注入对方：

getBean("indexService")=>this.populateBean中发现webService在singletonObjects的Map集合中没有且isSingletonCurrentlyInCreation(beanName)方法为false，那么就会返回null，实例化webService的BeanWrapper(BeanWrapper中封装实例对象已经属性描述器PropertyDescriptor)=>又走到this.populateBean()=>此时虽然singletonObjects中不存在这个BeanObject,但是isSingletonCurrentlyInCreation为true,走三级缓存拿对象，objectFactory中取earlySingleton后，在factory中删除相应factory，并且在earlySingletonObjects(第二级缓存)中加入此singletonObject方便后面还要对此对象进行循环依赖注入=>webService注入到IndexService,indexService完成剩下的生命周期,这里的三级缓存以及工厂模式用来提升效率的设计模式值得我们学习，后面本项目在开发时将充分利用SpringBoot的这一个思想，完整依赖注入代码分析如下

1.入口函数SpringApplication:

通过public static静态方法进入

这里面包含两个过程

1.instantialize SpringApplication对象：

public static ConfigurableApplicationContext run(Class<?>[] primarySources, String[] args) {

return (new SpringApplication(primarySources)).run(args);

}

2.进入run函数(无关代码已经省略)：

……….  
this.prepareContext(context, environment, listeners, applicationArguments, printedBanner);  
this.refreshContext(context);  
……….  
  
listeners.started(context);  
this.callRunners(context, applicationArguments);

3. refreshContext: 这个方法用来实例化Bean

private void refreshContext(ConfigurableApplicationContext context) {  
 this.refresh(context)；   
  
}

4.进入ApplicationContext的refresh进行实例化：

public void refresh() throws BeansException, IllegalStateException {  
 synchronized(this.startupShutdownMonitor) {  
 this.prepareRefresh();  
 ConfigurableListableBeanFactory beanFactory = this.obtainFreshBeanFactory();  
 this.prepareBeanFactory(beanFactory);  
  
 try {  
 this.postProcessBeanFactory(beanFactory);  
 this.invokeBeanFactoryPostProcessors(beanFactory);  
 this.registerBeanPostProcessors(beanFactory);  
 this.initMessageSource();  
 this.initApplicationEventMulticaster();  
 this.onRefresh();  
 this.registerListeners();  
 this.finishBeanFactoryInitialization(beanFactory);  
 this.finishRefresh();  
 } catch (BeansException var9) {  
 if (this.logger.isWarnEnabled()) {  
 this.logger.warn("Exception encountered during context initialization - cancelling refresh attempt: " + var9);  
 }  
  
 this.destroyBeans();  
 this.cancelRefresh(var9);  
 throw var9;  
 } finally {  
 this.resetCommonCaches();  
 }  
  
 }  
}

其中的finishBeanFactoryInitialization和finishRefresh用来初始化对象

6.BeanFactory的preInstantiateSingletons()方法解析：

这里的多层循环目的是为了让所有的单例bean和某些工厂bean都得到实例化,并且在最后可以方便对最后的初始化后的bean进行处理

public void preInstantiateSingletons() throws BeansException {  
 if (this.logger.isDebugEnabled()) {  
 this.logger.debug("Pre-instantiating singletons in " + this);  
 }  
  
 List<String> beanNames = new ArrayList(this.beanDefinitionNames);  
 Iterator var2 = beanNames.iterator();  
  
 while(true) {  
 String beanName;  
 Object bean;  
 do {  
 while(true) {  
 RootBeanDefinition bd;  
 do {  
 do {  
 do {  
 if (!var2.hasNext()) {  
 var2 = beanNames.iterator();  
  
 while(var2.hasNext()) {  
 beanName = (String)var2.next();  
 Object singletonInstance = this.getSingleton(beanName);  
 if (singletonInstance instanceof SmartInitializingSingleton) {  
 SmartInitializingSingleton smartSingleton = (SmartInitializingSingleton)singletonInstance;  
 if (System.getSecurityManager() != null) {  
 AccessController.doPrivileged(() -> {  
 smartSingleton.afterSingletonsInstantiated();  
 return null;  
 }, this.getAccessControlContext());  
 } else {  
 smartSingleton.afterSingletonsInstantiated();  
 }  
 }  
 }  
  
 return;  
 }  
  
 beanName = (String)var2.next();  
 bd = this.getMergedLocalBeanDefinition(beanName);  
 } while(bd.isAbstract()); //单例对象实例化不会经过这行  
 } while(!bd.isSingleton()); //单例对象实例化不会经过这行  
 } while(bd.isLazyInit());  
  
 if (this.isFactoryBean(beanName)) {  
 bean = this.getBean("&" + beanName);  
 break;  
 }  
  
 this.getBean(beanName) ; //这行代码进入bean的生命周期的下一个阶段,从以上的while循环可以知道,bean scope是Prototype和FactoryBean的不会经过这行,因为factory会break,而prototype会继续循环,进不来  
 }  
 } while(!(bean instanceof FactoryBean)); //单例显然不是FactoryBean回头继续初始化

//如果是工厂bean，那么会推断此工厂是否是eagerinit,获得工厂bean,在下面的代码推断是否需要isEagerInit  
 FactoryBean<?> factory = (FactoryBean)bean;  
 boolean isEagerInit;  
 if (System.getSecurityManager() != null && factory instanceof SmartFactoryBean) {  
 SmartFactoryBean var10000 = (SmartFactoryBean)factory;  
 ((SmartFactoryBean)factory).getClass();  
 isEagerInit = (Boolean)AccessController.doPrivileged(var10000::isEagerInit, this.getAccessControlContext());  
 } else {  
 isEagerInit = factory instanceof SmartFactoryBean && ((SmartFactoryBean)factory).isEagerInit();  
 }  
  
 if (isEagerInit) {  
 this.getBean(beanName);  
 }  
 }  
}

7.进入this.getBean通过层层空壳函数，会进入如下代码：

protected <T> T doGetBean(String name, @Nullable Class<T> requiredType, @Nullable Object[] args, boolean typeCheckOnly) throws BeansException {  
 String beanName = this.transformedBeanName(name); //对名字进行校验，验证bean的名字是否非法  
 Object sharedInstance = this.getSingleton(beanName); //从Spring的三级缓存中去取Singleton对象  
 Object bean;  
 if (sharedInstance != null && args == null) {  
 if (this.logger.isDebugEnabled()) {

if (this.isSingletonCurrentlyInCreation(beanName)) {  
 this.logger.debug("Returning eagerly cached instance of singleton bean '" + beanName + "' that is not fully initialized yet - a consequence of a circular reference");

//因为这个对象依赖了一个正在初始化的对象，所以返回一个eagerly cached instance of singleton.  
 } else {  
 this.logger.debug("Returning cached instance of singleton bean '" + beanName + "'");//得到一个已经初始化成功的对象  
 }  
 }  
 //不是工厂bean的话直接返回bean  
 bean = this.getObjectForBeanInstance(sharedInstance, name, beanName, (RootBeanDefinition)null);  
 } else {  
 ……….  
  
  
 try {

//对@DependsOn注解所标注的bean进行提前注册  
 RootBeanDefinition mbd = this.getMergedLocalBeanDefinition(beanName);  
 this.checkMergedBeanDefinition(mbd, beanName, args);  
 String[] dependsOn = mbd.getDependsOn();  
 String[] var11;  
 if (dependsOn != null) {  
 var11 = dependsOn;  
 int var12 = dependsOn.length;  
  
 for(int var13 = 0; var13 < var12; ++var13) {  
 String dep = var11[var13];

//判断是否循环depends-on,spring不支持循环depends-on @DependsOn  
 if (this.isDependent(beanName, dep)) {  
 throw new BeanCreationException(mbd.getResourceDescription(), beanName, "Circular depends-on relationship between '" + beanName + "' and '" + dep + "'");  
 }  
  
 this.registerDependentBean(dep, beanName);  
  
 try {  
 this.getBean(dep); //对依赖bean提前进行注册  
 } catch (NoSuchBeanDefinitionException var24) {  
 throw new BeanCreationException(mbd.getResourceDescription(), beanName, "'" + beanName + "' depends on missing bean '" + dep + "'", var24);  
 }  
 }  
 }  
  
 if (mbd.isSingleton()) {  
 sharedInstance = this.getSingleton(beanName, () -> {  
 try {  
 return this.createBean(beanName, mbd, args);  
 } catch (BeansException var5) {  
 this.destroySingleton(beanName);  
 throw var5;  
 }  
 });

//在对其dependson的bean初始化完成后，初始化此Bean  
 bean = this.getObjectForBeanInstance(sharedInstance, name, beanName, mbd);  
   
………..

8.getSingleton中的具体代码：

public Object getSingleton(String beanName, ObjectFactory<?> singletonFactory) {  
 Assert.notNull(beanName, "Bean name must not be null");  
 synchronized(this.singletonObjects) {  
 Object singletonObject = this.singletonObjects.get(beanName);  
 if (singletonObject == null) {  
 if (this.singletonsCurrentlyInDestruction) {  
 throw new BeanCreationNotAllowedException(beanName, "Singleton bean creation not allowed while singletons of this factory are in destruction (Do not request a bean from a BeanFactory in a destroy method implementation!)");  
 }  
  
 if (this.logger.isDebugEnabled()) {  
 this.logger.debug("Creating shared instance of singleton bean '" + beanName + "'");  
 }  
  
 this.beforeSingletonCreation(beanName); //这一步标记了beanCurrentlyInCreation,使得一个未被初始化完成的Bean也可以被其他bean引用  
 boolean newSingleton = false;  
 boolean recordSuppressedExceptions = this.suppressedExceptions == null;  
 if (recordSuppressedExceptions) {  
 this.suppressedExceptions = new LinkedHashSet();  
 }  
  
 try {  
 singletonObject = singletonFactory.getObject();  
 newSingleton = true;  
 } catch (IllegalStateException var16) {  
 singletonObject = this.singletonObjects.get(beanName);  
 if (singletonObject == null) {  
 throw var16;  
 }  
 } catch (BeanCreationException var17) {  
 BeanCreationException ex = var17;  
 if (recordSuppressedExceptions) {  
 Iterator var8 = this.suppressedExceptions.iterator();  
  
 while(var8.hasNext()) {  
 Exception suppressedException = (Exception)var8.next();  
 ex.addRelatedCause(suppressedException);  
 }  
 }  
  
 throw ex;  
 } finally {  
 if (recordSuppressedExceptions) {  
 this.suppressedExceptions = null;  
 }  
  
 this.afterSingletonCreation(beanName);  
 }  
  
 if (newSingleton) {  
 this.addSingleton(beanName, singletonObject);  
 }  
 }  
  
 return singletonObject;  
 }  
}

8.Spring中的三级缓存详解

Spring工厂实现循环依赖的三级缓存：

1.singletonObjects(Spring的一级缓存)

2.ObjectFactory

3.earlySingletonObjects

protected Object getSingleton(String beanName, boolean allowEarlyReference) {

Object singletonObject = this.singletonObjects.get(beanName); // private final Map<String, Object> singletonObjects = new ConcurrentHashMap(256）；用来存储所有已经初始化(属性已经成功注入)

// 的对象

if (singletonObject == null && this.isSingletonCurrentlyInCreation(beanName)) { //发现没有初始化好且此对象正在初始化中

synchronized(this.singletonObjects) {

singletonObject = this.earlySingletonObjects.get(beanName);

if (singletonObject == null && allowEarlyReference) {

ObjectFactory<?> singletonFactory = (ObjectFactory)this.singletonFactories.get(beanName);

if (singletonFactory != null) {

singletonObject = singletonFactory.getObject();

this.earlySingletonObjects.put(beanName, singletonObject);

this.singletonFactories.remove(beanName);

}

}

}

}

return singletonObject;

}

private final Map<String, Object> singletonObjects = new ConcurrentHashMap(256）；

9.doCreateBean:

protected Object doCreateBean(String beanName, RootBeanDefinition mbd, @Nullable Object[] args) throws BeanCreationException {

BeanWrapper instanceWrapper = null;

if (mbd.isSingleton()) {

instanceWrapper = (BeanWrapper)this.factoryBeanInstanceCache.remove(beanName);

}

if (instanceWrapper == null) {

instanceWrapper = this.createBeanInstance(beanName, mbd, args);

}

Object bean = instanceWrapper.getWrappedInstance(); //这个就是被new出的一个对象bean

Class<?> beanType = instanceWrapper.getWrappedClass();

if (beanType != NullBean.class) {

mbd.resolvedTargetType = beanType;

}

synchronized(mbd.postProcessingLock) {

if (!mbd.postProcessed) {

try {

this.applyMergedBeanDefinitionPostProcessors(mbd, beanType, beanName);

} catch (Throwable var17) {

throw new BeanCreationException(mbd.getResourceDescription(), beanName, "Post-processing of merged bean definition failed", var17);

}

mbd.postProcessed = true;

}

}

boolean earlySingletonExposure = mbd.isSingleton() && this.allowCircularReferences && this.isSingletonCurrentlyInCreation(beanName); //允许循环依赖

if (earlySingletonExposure) {

if (this.logger.isTraceEnabled()) {

this.logger.trace("Eagerly caching bean '" + beanName + "' to allow for resolving potential circular references"); //缓存bean来方便后面潜在的循环依赖

}

this.addSingletonFactory(beanName, () -> {

return this.getEarlyBeanReference(beanName, mbd, bean); //earlySingletonExposure和允许循环依赖的意思差不多，这时候此对象还没有被初始化，将一个

});

}

Object exposedObject = bean;

try {

this.populateBean(beanName, mbd, instanceWrapper);

exposedObject = this.initializeBean(beanName, exposedObject, mbd);

} catch (Throwable var18) {

if (var18 instanceof BeanCreationException && beanName.equals(((BeanCreationException)var18).getBeanName())) {

throw (BeanCreationException)var18;

}

throw new BeanCreationException(mbd.getResourceDescription(), beanName, "Initialization of bean failed", var18);

}

if (earlySingletonExposure) {

Object earlySingletonReference = this.getSingleton(beanName, false);

if (earlySingletonReference != null) {

if (exposedObject == bean) {

exposedObject = earlySingletonReference;

} else if (!this.allowRawInjectionDespiteWrapping && this.hasDependentBean(beanName)) {

String[] dependentBeans = this.getDependentBeans(beanName);

Set<String> actualDependentBeans = new LinkedHashSet(dependentBeans.length);

String[] var12 = dependentBeans;

int var13 = dependentBeans.length;

for(int var14 = 0; var14 < var13; ++var14) {

String dependentBean = var12[var14];

if (!this.removeSingletonIfCreatedForTypeCheckOnly(dependentBean)) {

actualDependentBeans.add(dependentBean);

}

}

if (!actualDependentBeans.isEmpty()) {

throw new BeanCurrentlyInCreationException(beanName, "Bean with name '" + beanName + "' has been injected into other beans [" + StringUtils.collectionToCommaDelimitedString(actualDependentBeans) + "] in its raw version as part of a circular reference, but has eventually been wrapped. This means that said other beans do not use the final version of the bean. This is often the result of over-eager type matching - consider using 'getBeanNamesOfType' with the 'allowEagerInit' flag turned off, for example.");

}

}

}

}

try {

this.registerDisposableBeanIfNecessary(beanName, bean, mbd);

return exposedObject;

} catch (BeanDefinitionValidationException var16) {

throw new BeanCreationException(mbd.getResourceDescription(), beanName, "Invalid destruction signature", var16);

}

}

10.楼上的addSingletonFactory的工厂类用来获得这个正在create的bean：

源码：

public interface ObjectFactory<T> {

T getObject() throws BeansException;

}

this.addSingletonFactory(beanName, () -> {

return this.getEarlyBeanReference(beanName, mbd, bean);

}); //判断是否要做aop

protected Object getEarlyBeanReference(String beanName, RootBeanDefinition mbd, Object bean) {

Object exposedObject = bean;

if (!mbd.isSynthetic() && this.hasInstantiationAwareBeanPostProcessors()) {

Iterator var5 = this.getBeanPostProcessors().iterator();

while(var5.hasNext()) {

BeanPostProcessor bp = (BeanPostProcessor)var5.next();

if (bp instanceof SmartInstantiationAwareBeanPostProcessor) {

SmartInstantiationAwareBeanPostProcessor ibp = (SmartInstantiationAwareBeanPostProcessor)bp;

exposedObject = ibp.getEarlyBeanReference(exposedObject, beanName);

}

}

}

return exposedObject;

}

protected void addSingletonFactory(String beanName, ObjectFactory<?> singletonFactory) {

Assert.notNull(singletonFactory, "Singleton factory must not be null");

synchronized(this.singletonObjects) {

if (!this.singletonObjects.containsKey(beanName)) {

this.singletonFactories.put(beanName, singletonFactory);

this.earlySingletonObjects.remove(beanName);

this.registeredSingletons.add(beanName);

}

}

}

11.

Spring三级缓存：

private final Map<String, Object> singletonObjects = new ConcurrentHashMap(256); //1.一级缓存 已经初始化后的单例对象缓存 单例池 spring容器

private final Map<String, ObjectFactory<?>> singletonFactories = new HashMap(16); //2.三级缓存 单例对象工厂，一旦返回成功一次早暴露的对象

private final Map<String, Object> earlySingletonObjects = new HashMap(16); //3. 二级缓存 存在三级缓存是为了防止重复创建(因为多个实例可能依赖同个对象，你不能创建多个，直接存入三级缓存，以后每次从三级缓存中拿就行)，因为singletonFactories返回时会经过一系列步骤，这样也会节省时间

private final Set<String> registeredSingletons = new LinkedHashSet(256); //已经注册的Bean的BeanNames,不一定要走完生命周期

private final Set<String> singletonsCurrentlyInCreation = Collections.newSetFromMap(new ConcurrentHashMap(16)); //正在创建中的BeanNames

protected void beforeSingletonCreation(String beanName) {

if (!this.inCreationCheckExclusions.contains(beanName) && !this.singletonsCurrentlyInCreation.add(beanName)) {

throw new BeanCurrentlyInCreationException(beanName);

}

} //通过这个方法标识上某个Bean正在创建过程中

12.

注册Singleton到singletonObjects已注册bean中

public void registerSingleton(String beanName, Object singletonObject) throws IllegalStateException {

Assert.notNull(beanName, "Bean name must not be null");

Assert.notNull(singletonObject, "Singleton object must not be null");

synchronized(this.singletonObjects) {

Object oldObject = this.singletonObjects.get(beanName);

if (oldObject != null) {

throw new IllegalStateException("Could not register object [" + singletonObject + "] under bean name '" + beanName + "': there is already object [" + oldObject + "] bound");

} else {

this.addSingleton(beanName, singletonObject);

}

}

}

protected void addSingleton(String beanName, Object singletonObject) {

synchronized(this.singletonObjects) {

this.singletonObjects.put(beanName, singletonObject);

this.singletonFactories.remove(beanName);

this.earlySingletonObjects.remove(beanName);

this.registeredSingletons.add(beanName);

SpringBoot是由Pivotal团队2013年进行开发设计的、并且在2014年4月发布第一个版本的全新开源的轻量级框架。结合Spring4.0，其拥有Spring框架所拥有的所有优良性质，最关键的是它可以进一步简化在之前Spring开发中臃肿的配置文件而造成程序员本应该专注于业务而因为基础框架的搭建而浪费了大量的时间。而SpringBoot这个脚手架通过大量的简化配置来使得Spring应用的搭建和开发全部过程变得如此轻松与容易，程序员可以更加专注于业务逻辑的设计而非基础框架。同时，SpringBoot使得程序员只需要继承一个SpringBoot parent工程就可以集成大量的框架并且避免依赖包的版本冲突，以及解决了引用的不稳定性等问题。

Spring Boot所具备的特征有：

（1）可以创建独立的Spring应用程序，并且基于其Maven或Gradle插件，可以创建可执行的JARs和WARs；

（2）内嵌Tomcat或Jetty等Servlet容器；

（3）提供自动配置的“starter”项目对象模型（POMS）以简化Maven配置；

（4）尽可能自动配置Spring容器；

（5）提供准备好的特性，如指标、健康检查和外部化配置；

（6）绝对没有代码生成，不需要XML配置。 [1] [2]

SpringBoot框架中还有两个非常重要的策略：开箱即用和约定优于配置。开箱即用，

Outofbox，是指在开发过程中，通过在MAVEN项目的pom文件中添加相关依赖包，然后使用对应注解来代替繁琐的XML配置文件以管理对象的生命周期。这个特点使得开发人员摆脱了复杂的配置工作以及依赖的管理工作，更加专注于业务逻辑。约定优于配置，Convention over configuration，是一种由SpringBoot本身来配置目标结构，由开发者在结构中添加信息的软件设计范式。这一特点虽降低了部分灵活性，增加了BUG定位的复杂性，但减少了开发人员需要做出决定的数量，同时减少了大量的XML配置，并且可以将代码编译、测试和打包等工作自动化。

SpringBoot应用系统开发模板的基本架构设计从前端到后台进行说明：前端常使用模板引擎，主要有FreeMarker和Thymeleaf，它们都是用Java语言编写的，渲染模板并输出相应文本，使得界面的设计与应用的逻辑分离，同时前端开发还会使用到Bootstrap、AngularJS、JQuery等；在浏览器的数据传输格式上采用Json，非xml，同时提供RESTfulAPI；SpringMVC框架用于数据到达服务器后处理请求

虽然SpringBoot的使用非常简单，但是我们在开始项目前需要对其源代码有更加详细的了解，这样才能在应用中更加灵活自如。而SpringBoot中最让人困惑的可能就是自动配置了。我们从SpringBoot的入口类开始一步一步详细解释为什么SpringBoot可以实现自动配置。

@SpringBootApplication

public class SpringApplicationExample{

public static void main(String[] args) {

SpringApplication.run(SpringApplicationExample.class);

}

}

通过Debug可以知道SpringBoot项目的自动配置入口文件是@EnableAutoConfiguration中的AutoConfigurationImportSelector.class

@Target({ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

@Documented

@Inherited

@AutoConfigurationPackage

@Import({AutoConfigurationImportSelector.class})

public @interface EnableAutoConfiguration {

String ENABLED\_OVERRIDE\_PROPERTY = "spring.boot.enableautoconfiguration";

Class<?>[] exclude() default {};

String[] excludeName() default {};

}

继续分析AutoConfigurationImportSelector源码中selectImports方法可以得知：

public String[] selectImports(AnnotationMetadata annotationMetadata) {

if (!this.isEnabled(annotationMetadata)) {

return NO\_IMPORTS;

} else {

try {

AutoConfigurationMetadata autoConfigurationMetadata = AutoConfigurationMetadataLoader.loadMetadata(this.beanClassLoader);

AnnotationAttributes attributes = this.getAttributes(annotationMetadata);

List<String> configurations = this.getCandidateConfigurations(annotationMetadata, attributes);

configurations = this.removeDuplicates(configurations);

configurations = this.sort(configurations, autoConfigurationMetadata);

Set<String> exclusions = this.getExclusions(annotationMetadata, attributes);

this.checkExcludedClasses(configurations, exclusions);

configurations.removeAll(exclusions);

configurations = this.filter(configurations, autoConfigurationMetadata);

this.fireAutoConfigurationImportEvents(configurations, exclusions);

return (String[])configurations.toArray(new String[configurations.size()]);

} catch (IOException var6) {

throw new IllegalStateException(var6);

}

}

}

1.this.isEnabled(annotationMetaData)就是检查是否在yaml中配置了自动配置为false，如果用户自行配置了false，那么springboot将不会进行自动配置，则用户需要自行手动配置所有的插件

2.其中的静态方法AutoConfigurationMetadataLoader.loadMetadata会加载META-INF/spring-autoconfigure-metadata.properties的文件，而这个文件对后面SpringBoot过滤某些不需要的bean的配置有重要作用

截取里面一部分的内容进行解释：

org.springframework.boot.autoconfigure.mongo.MongoAutoConfiguration.ConditionalOnClass=com.mongodb.MongoClient

当springboot加载MongoAutoConfiguration自动配置类时会先自行检查是否已经存在MongoClient，如果没有存在，则自动配置就不会配置当前类，自动过滤掉，关于SpringBoot是如何实现这一个特性的我们在下文进行分析

3.接下来就是对自动配置类的过滤和筛选了，我们先对实例annotationMetaData进行仔细剖析：

public class StandardAnnotationMetadata extends StandardClassMetadata implements AnnotationMetadata {

private final Annotation[] annotations;

private final boolean nestedAnnotationsAsMap;

public StandardAnnotationMetadata(Class<?> introspectedClass) {

this(introspectedClass, false);

}

在SpringBoot中这里的运行实例是StandardAnonotationMetaData类型，这个类型存储了我们入口类的字节码文件(即上面的introspectedClass)，在这里存储的就是SpringApplicationExample.class, 方便在进行自动配置时读取入口类的注解上的配置属性，可以告诉SpringBoot我们哪些Bean需要手动配置，哪些需要自动配置，让我们的开发更加得心应手

4.然后SpringBoot就会获取@SpringBootApplication复合注解中的@EnableAutonConfiguration的所有参数信息装入AnnotationAttributes实例中去，通过debug可以看到这里面有两个属性exclude和excludeName也证实了它的作用

5.然后SpringBoot通过getCandidateConfigurations从路径META-INF/spring.factories文件中获得等待配置的候选自动配置类

protected List<String> getCandidateConfigurations(AnnotationMetadata metadata, AnnotationAttributes attributes) {

List<String> configurations = SpringFactoriesLoader.loadFactoryNames(this.getSpringFactoriesLoaderFactoryClass(), this.getBeanClassLoader());

Assert.notEmpty(configurations, "No auto configuration classes found in META-INF/spring.factories. If you are using a custom packaging, make sure that file is correct.");

return configurations;

}

在这一步中的removeDuplicates(configurations)值得注意

configurations = this.removeDuplicates(configurations);

这里实现配置的去重是通过将一个list实例传入linkedHashSet中构造一个linkedhashset，而hashset只存储不同的string值，最后再构造一个arraylist实例就成功实现去重了

protected final <T> List<T> removeDuplicates(List<T> list) {

return new ArrayList(new LinkedHashSet(list));

}

接下来通过这两行代码

Set<String> exclusions = this.getExclusions(annotationMetadata, attributes);

this.checkExcludedClasses(configurations, exclusions);

configurations.removeAll(exclusions);

Springboot将会从候选bean类中去除用户在@EnableAutoConfiguration中指定的取消自动配置的bean

用于SpringBoot自动配置，很多class的配置需要依赖另外class的存在，而filter这个方法的最主要的作用就是进一步筛选没有前置class的那些class从自动配置class的List中删除掉

private List<String> filter(List<String> configurations, AutoConfigurationMetadata autoConfigurationMetadata) {

long startTime = System.nanoTime();

String[] candidates = (String[])configurations.toArray(new String[configurations.size()]);

boolean[] skip = new boolean[candidates.length];

boolean skipped = false;

Iterator var8 = this.getAutoConfigurationImportFilters().iterator();

while(var8.hasNext()) {

AutoConfigurationImportFilter filter = (AutoConfigurationImportFilter)var8.next();

this.invokeAwareMethods(filter);

boolean[] match = filter.match(candidates, autoConfigurationMetadata);

for(int i = 0; i < match.length; ++i) {

if (!match[i]) {

skip[i] = true;

skipped = true;

}

}

}

if (!skipped) {

return configurations;

} else {

List<String> result = new ArrayList(candidates.length);

int numberFiltered;

for(numberFiltered = 0; numberFiltered < candidates.length; ++numberFiltered) {

if (!skip[numberFiltered]) {

result.add(candidates[numberFiltered]);

}

}

if (logger.isTraceEnabled()) {

numberFiltered = configurations.size() - result.size();

logger.trace("Filtered " + numberFiltered + " auto configuration class in " + TimeUnit.NANOSECONDS.toMillis(System.nanoTime() - startTime) + " ms");

}

return new ArrayList(result);

}

}

其中过滤class中最关键的方法就filter.match(candidates,autoConfigurationMetaData);

public boolean[] match(String[] autoConfigurationClasses, AutoConfigurationMetadata autoConfigurationMetadata) {

ConditionEvaluationReport report = this.getConditionEvaluationReport();

ConditionOutcome[] outcomes = this.getOutcomes(autoConfigurationClasses, autoConfigurationMetadata);

boolean[] match = new boolean[outcomes.length];

for(int i = 0; i < outcomes.length; ++i) {

match[i] = outcomes[i] == null || outcomes[i].isMatch();

if (!match[i] && outcomes[i] != null) {

this.logOutcome(autoConfigurationClasses[i], outcomes[i]);

if (report != null) {

report.recordConditionEvaluation(autoConfigurationClasses[i], this, outcomes[i]);

}

}

}

return match;

}

## RabbitMQ

RabbitMQ是一个开源的消息中间件，实现了AMQP规范，即Advanced Message Queuing Protocol,高级消息队列协议，RabbitMQ消息中间件是由erlang语言编写的，而其他语言都有办法通过RabbitMQ提供的接口来进行消息队列的操作，其中具有三个非常重要的角色，producer生产者，consumer消费者以及queue队列。

**Producer**

表示消息的生产者，用于创建消息后发送RabbitMQ的代理服务器上，消息包括有效载荷和标签

**Consumer**

表示消息的接收者，例如客户端

**Queue**

相当于信箱用于接收生产者的消息存储在消息队列中等待消费者的异步消费

在RabbitMQ中发送方和接收方发送和接受消息的模式主要这几种：

Fanout: 广播，将消息交给所有绑定到交换机的队列

Direct：定向，把消息交给符合指定routing key 的队列

Topic：通配符，把消息交给符合routing pattern（路由模式） 的队列

在我们的企业市场营销管理系统中，各个部门的数据有时需要和我们的系统进行同步但又不能影响工作速度，所以可以设置消息队列来进行数据同步，以及电子邮件模块和短信模块的发送信息可以通过消息队列与营销活动模块进行互动

# 企业市场营销系统的需求分析

本章节的内容主要是关于本系统的需求分析，根据实地公司实习，并且询问相应需求方的需求后，抽象出需求模型，并进行详细分析后，可以确定本系统包含以下几个功能模块：零售行业营销系统的功能模块、产品管理功能模块、营销活动管理功能模块、统计分析功能模块、交易管理功能模块、用户管理功能模块

## 系统总体需求分析

在敏捷开发中，软件开发的第一个阶段就是收集和分析需求，在思考什么样的需求需要被采纳时，开发者和需求方需要达成一致，这样最后的产品才能满足大家的需求。所以在软件开发中，需求分析需要格外的仔细。营销管理系统的需求分析给开发者提供了一个最终产品的初步印象，这样，在开发的时候开发者可以设定各模块的优先级，并且利用自己的专业知识来事先思考这些模块是怎么实现的，提前准备，比起后期开发偏离方向再回头修改不仅省时省力，更让开发者和需求方节省了很大的成本。而且没有清晰的需求分析，一旦出现错误的决定有时也会导致严重的后果发生。

营销管理系统对一个公司的重要性可想而知，一个好的营销管理系统不仅能适应多变的公司营销管理策略还能为提升公司的产品地位打下坚实的基础。并且在我们的营销管理系统中，由于公司的业务庞大，经常需要存储数量巨大的用户，并且为了能够精准营销，需要这些用户的信息量丰富以便让公司拥有足够的信息去精细推销自己的产品。并且由于使用人员一般缺乏计算机的专业知识，我们需要对界面进行一定的优化，如果出错尽量不要直接反馈给用户，同时使用人员涉及的部门非常广，因此我们也需要对系统进行更加详细的设计以便根据不同部门用户的特点可以做到都可以很方便运用。同时，也要注意到现在市面上的一些营销管理系统的问题，例如界面简陋，功能单一，存储数据量少，搜索速度慢，无法为公司员工推送精准的用户信息等。我们需要注意这些问题，并且能尽量把问题解决，解决好以便更好的促进公司的利益提升。

同时由于现在公司都会在不断扩大自己的员工数量和工作地点，架构如果能采用微服务分布式并且拥有集群负载均衡的特点，可以为公司日后壮大提升规模后，如果需要增加用户数量减少很大的成本，公司只要简单的部署相应的容器就可以对业务进行扩展。

由于现在数字化营销需要结合个性化的推销手段给用户耳目一新的产品广告视觉，这样能让用户有更强的购买愿望。所以市场营销管理系统还必须要具有个性化邮件和短信的功能，同时需要提供相应的模板方便公司的需求使用用户能更方便的选择自己相应的产品广告模板，精准营销。

市场营销系统的邮件和短信发送功能还需要具有定时功能，不能在用户上班工作时间对用户进行打扰，这时候相应的部门和市场营销部门需要共享用户数据库，用户在相应部门注册与登记信息的时候必定会记录下自己方便的时间，然后相应的市场营销部门的数据库也要进行同步更新，这样市场营销部门可以在用户有空的时候对其发送推广消息，以避免打搅用户休息。同时公司的推广模板不能够一成不变，为了迎合营销需求，公司员工将可以自行创建模板，模板可以是markdown格式创建，也可以是html的方式，这样随着时代的发展，广告模板也能够与时俱进。公司的人数将会随着业务的发展越来越多，如果没有对联系客户进行精细的管理和分组也会造成管理困难的问题。

因此，根据上述问题的分析，可以将企业市场营销管理系统抽象为以下几个主要模块，方便进行进一步设计与开发：

系统需要拥有界面美观的UI，功能需要容易使用不能过于抽象，尽可能结合微服务构架开发前后端分离的系统，满足各个部门员工快速上手的需求，减少培训的消耗。

系统需要有简洁大方的主页可以显示统计信息，因为企业市场营销系统涉及的内容非常广泛，所以主页需要显示各个模块的统计信息，例如当日订单金额，登记在案用户人数，用户分群数量，网站客户流量，市场营销转化利益计算，模板数量，当日新增用户人数，当日推广人数等。用最与时俱进的技术做到更加简洁美观直观的显示信息，做到让公司用户登录系统就懂得从哪里下手。

企业市场营销管理系统也需要拥有鉴权功能，这个鉴权功能需要足够安全，结合到时下最新的加密技术并且能够动态显示菜单，对后台api的权限进行过滤，避免权限不足人员的误操作。

企业市场营销管理系统需要具有消息队列框架的搭建，由于各个部门之间的数据库都不同步，为了能够实现同步该系统必须具有消息队列方便其他部门在新增用户记录的时候也能对市场营销管理系统的数据做到同步更新。

企业市场营销管理系统需要有短信收发和电子邮件收发的基本功能，并且默认就具有相应的模板，方便不需要个性化的需求用户直接使用，而且这个电子邮件的收发也需要依赖到消息队列的使用，并且需要调用诸如阿里云和华为云的短信服务，然后通过消息队列进行异步收发。

企业市场营销系统需要对繁杂的联系人信息进行分类，因为各个部门的人员所面向的用户都各不相同，所以为了能够给各个部门更加精准的推送他们所需要的用户，系统必须具有对搜索结果进行分类，精细过滤的功能，这个过滤功能不能过于简单，必须要对不同的用户的各个条件字段进行相应的聚合，且搜索速度一定要越快越好，在企业中节省的每分每秒都意味着巨大的利益提升。

综上所述，企业市场营销管理系统所面向的用户有系统管理人员，各部门的数据录入操作人员，各部门的营销人员。系统管理人员可以具有管理系统的权限，各部门的数据录入人员可以具有联系人管理模块，联系人分群管理模块，部门信息修改模块的访问权限，各部门的营销人员具有电子邮件模块，短信模块，联系人管理模块，营销业绩概览和分析的访问权限。

## 系统交互分析

### 系统外部关系

从上面的分析我们可以知道，企业市场营销管理系统是支撑一个企业利益获取的一个十分重要的系统。一个大型企业内部必然具有多个部门，每个部门下面都具有自己数量庞大的用户群体，所以这些部门想要对自己的用户进行精准营销，必然要用到企业市场营销管理系统。同时，企业市场营销管理系统需要公司的用到公司的账号管理系统的数据，以方便根据不同员工的身份信息显示与其权限相匹配的界面。从分析可以看出，企业市场营销管理系统与企业中各部门的子系统之间的关系一定是紧密相连，密不可分的，系统外部结构如图所示

各部门的用户管理系统在有新的用户详细信息录入后需要通过消息队列实现各自用户管理系统的信息和企业市场营销管理系统的数据库信息达到同步的目的，同时各自部门的营销人员可能在企业市场营销管理系统中对客户的信息进行修改以方便营销所以企业市场营销管理系统也要利用到消息队列保持数据的一致性。各个部门的订单系统需要将他们成交的订单信息内容发送到消息队列，让企业市场营销管理系统的统计分析功能可以获取到营销的成果数据，并且各个部门的部门管理系统也要通过企业营销管理系统获取到本部门的营销统计结果数据，再者公司的账号管理系统需要向企业营销管理系统提供相应的账号安全信息方便企业市场营销管理系统根据不同的用户实现鉴权操作。

### 系统内部关系

客户管理将收集全渠道的客户信息，满足客户信息的单个创建、批量创建、编辑、删除、批量删除的基础操作，其次在客户分组中可以对不同的用户属性字段进行条件筛选，完成客户数据的过滤为精准化营销提供保证。而电子邮件模块，短信模块需要依赖客户管理的信息进行发送，同时营销活动管理模块需要依赖电子邮件模块和短信模块进行相应的营销活动。并且交易模块还需要依赖订单信息模块，最后统计分析模块也要依赖订单信息模块、交易模块、电子邮件、短信模块和客户管理模块。

客户管理可以对从其他部门的客户管理系统导入的客户数据信息进行查找分析，并且根据不同的字段可以进行精细化查找，查找后的聚合结果还可以被企业市场营销管理系统的统计分析模块利用并显示在主页上，客户管理可以从csv等文件中批量扫描信息，并将信息录入到数据库中去，而这个过程也可以通过消息队列同步到相应部门的子系统数据库中去，形成数据统一，其他部门子系统的客户端的订单信息也会通过消息队列传送到企业营销管理系统中，而系统从消息队列通过异步读取这些订单信息并把相应的统计信息显示在主页的统计分析中。营销活动模块的运行也会调用到电子邮件模块和短信模块，并且每一次调用都会引起统计分析模块的状态变化，而电子邮件模块和短信模块也会通过消息队列调用到华为云或者阿里云的短信或电子邮件发送接口，实现营销信息的异步发送。

## 用户管理的需求分析

（1）账号管理：可以设置不同的子账号配置不同的权限进行子账号的管理。

（2）权限管理：提供子账号不同的权限设置。对于权限划分为操作权限与数据权限两种。

a．操作权限：包括对于平台各个模块的功能的增删改查操作进行细分。

b．数据权限：对于客户信息、客户记录、素材、营销任务可以设置可见的数据范围。比如本人的数据或者全组织的数据。

## 统计分析业务的需求分析

（1）数据接入：在数据中心按进行数据接入，系统提供了多样化的数据接入方式，目前支持服务端API接口这种接入方式来获取源数据。(服务端API接口：采用REST原则设计，使用秘钥授权和身份认证。请求和返回均采用规范的JSON格式，且编码为UTF8。)中查看对应的数据错误详情。

（2）营销任务概览：统计时间段内的所有旅程的任务概览。

a．在数据来源内选择筛选时间段 b．在数据来源内选择指定营销任务

（3）发送报告：统计时间段内某渠道（短信、邮件和微信）的详细数据报告。

a．在数据来源内选择筛选时间段 b．在数据来源内选择指定营销任务 渠道报告：列举这段时间或者某营销任务内的不同渠道的发送素材总数和送达总数，点击“任务明细”可查看任务报告。 任务报告：当前仅可查看单次和重复营销任务的数据报告。也可以查看当前渠道内所有素材的发送报告。

（4）数据导出：可导出邮件、短信相关的数据等所有属于统计报告

用户分析：增长趋势、渠道偏好、消费偏好等；

订单分析：渠道影响、热销商品、客单价等；

渠道分析：发送趋势、转化漏斗、活跃用户等；

## 营销活动业务的需求分析

短信、邮件、等全渠道用户生命周期规划建议（新用户培育、复购用户再营销、沉睡用户激活等），营销日程上设定的生命周期活动，对增量用户可不断重复营销任务。合理搭配最优沟通渠道，促进营销转化。

单次营销：针对既定得目标受众进行多渠道得推送，可以设置条件分支、随机分支等控件。

周期营销：周期营销相当于单次任务得周期性运行，重复周期可按周、天、月。

触发营销：触发营销是由客户事件得触发为开始条件，并且由客户触发事件实时驱动自动流程。实时任务开始后满足触发条件的客户会实时进入任务，未触发的客户不会进入实时任务。

a．触发事件如点击网页按钮等。

b．客户的数据库字段发生属性变更事件等。

营销场景：条件控件

可以增加判断条件，字段属性、互动行为（邮件互动、短信互动、网页互动）等判断条件，

题名页

题名页包括中文题名页和英文题名页，主要由学校代码、分类号、学号、密级、论文题目、作者姓名、一级学科、二级学科（博士学位论文、学术型和同等学力硕士学位论文）、领域（专业学位硕士学位论文）、学位类别、指导教师姓名、职称（博士学位论文、学术型和同等学力硕士学位论文）、学校、企业导师姓名、职称（专业学位硕士学位论文）、提交日期等部分组成。没有企业导师的专业学位类别请将“企业导师姓名及职称”栏目删除。

（1）学校代码：指本单位编号，我校代码是10701。

（2）分类号：指在《中国图书资料分类法》中的分类号（填写前四位即可）。

（3）学号：按照入学时研究生院编制的统一编号填写。

（4）密级：密级由导师确定，分为公开和秘密两种。

学校代码和分类号位于确定位置的文本框中，文本框格式为水平位置：相对于右侧页边距绝对位置0.2厘米；垂直位置：相对于下侧页边距绝对位置0.3厘米；文字环绕方式为浮于文字上方；文本框大小：高度为绝对值1.1厘米，宽度为绝对值4.5厘米。学号和密级位于确定位置的文本框中，文本框格式为水平位置：相对于右侧页边距绝对位置10.9厘米；垂直位置：相对于下侧页边距绝对位置0.3厘米；文字环绕方式为浮于文字上方；文本框大小：高度为绝对值1.1厘米，宽度为绝对值4.5厘米。中文题名页中的学校代码、分类号、学号和密级的字体为宋体，字号为五号加粗，行距为多倍行距1.2，段落间距为段前0磅，段后0磅；

学位论文题目位于确定位置的文本框中，文本框格式为水平位置：相对于右侧页边距绝对位置0厘米；垂直位置：相对于下侧页边距绝对位置11厘米；文字环绕方式为浮于文字上方；文本框大小：高度为绝对值3.2厘米，宽度为绝对值15.5厘米。字体为宋体，字号为二号加粗，行距为固定值30磅，段落间距为段前0磅，段后0磅；

作者姓名、指导教师姓名职称、一级学科、二级学科、领域、学位类别、提交日期位于确定位置的文本框中，文本框格式为水平位置：相对于右侧页边距绝对位置4.5厘米；垂直位置：相对于下侧页边距绝对位置16厘米；文字环绕方式为浮于文字上方；文本框大小：高度为绝对值8.6厘米，宽度为绝对值10.5厘米。标题和具体内容的字体为宋体，标题字号为四号加粗，具体内容的字号为四号不加粗，行距为固定值32磅，段落间距为段前0磅，段后0磅。

英文题名页中的学科填写一级学科（专业学位填写类别），学位论文题目字体为Times New Roman，字号二号加粗，行距为固定值30磅，段落间距为段前0磅，段后0磅，其他内容的字体为Times New Roman，字号三号，行距为固定值30磅，段落间距为段前0磅，段后0磅。学位论文题目位于确定位置的文本框中，文本框格式为水平位置：相对于右侧页边距绝对位置0厘米；垂直位置：相对于下侧页边距绝对位置0厘米；文字环绕方式为浮于文字上方；文本框大小：高度为绝对值3.5厘米，宽度为绝对值15.5厘米。学科信息文本框格式为水平位置：相对于右侧页边距绝对位置0厘米；垂直位置：相对于下侧页边距绝对位置6厘米；文字环绕方式为浮于文字上方；文本框大小：高度为绝对值5.5厘米，宽度为绝对值15.5厘米。作者信息文本框格式为水平位置：相对于右侧页边距绝对位置0厘米；垂直位置：相对于下侧页边距绝对位置18.7厘米；文字环绕方式为浮于文字上方；文本框大小：高度为绝对值4.5厘米，宽度为绝对值15.5厘米。

## 声明

声明是对学位论文创新性和使用授权的声明和说明，论文提交图书馆和存档时作者本人和指导教师必须签名确认。

声明部分标题字体为宋体，字号为四号加粗，居中排列，行距为固定值20磅，段落间距为段前0磅，段后0磅；正文字体为宋体，字号为小四号，行距为固定值20磅，段落间距为段前0磅，段后0磅；标题与正文之间空一行，签名行与正文之间空一行，日期行与签名行之间空一行。

## 摘要

摘要是学位论文的内容不加注释和评论的简短陈述，简明扼要陈述学位论文的研究目的、内容、方法、成果和结论，重点突出学位论文的创造性成果和观点。摘要包括中文摘要和英文摘要，硕士学位论文中文摘要字数一般为1000字左右，博士学位论文中文摘要字数一般为1500字左右。英文摘要内容与中文摘要内容保持一致，翻译力求简明精准。摘要的正文下方需注明论文的关键词，关键词一般为3～8个，关键词和关键词之间用逗号并空一格。

中文摘要标题字体为黑体，字号为三号，居中排列，行距为固定值20磅，段落间距为段前24磅，段后18磅；正文字体为宋体，字号为小四号，行距为固定值20磅，段落间距为段前0磅，段后0磅；关键词和正文之间空一行，关键词字体为宋体，字号为小四号，标题加粗。英文摘要标题字体为Times New Roman，字号为三号，居中排列，行距为固定值20磅，段落间距为段前24磅，段后18磅；正文的每一段落首行不空格，段落与段落之间空一行；正文字体为Times New Roman，字号为小四号，行距为固定值20磅，段落间距为段前0磅，段后0磅；关键词字体为Times New Roman，字号为小四号，标题加粗。

## 插图索引

学位论文中插图的目录索引。插图索引标题字体为黑体，字号为三号，居中排列，行距为固定值20磅，段落间距为段前24磅，段后18磅；正文内容字体为宋体，字号为小四号，行距为固定值20磅，段落间距为段前0磅，段后0磅。

## 表格索引

学位论文中表格的目录索引。表格索引标题字体为黑体，字号为三号，居中排列，行距为固定值20磅，段落间距为段前24磅，段后18磅；正文内容字体为宋体，字号为小四号，行距为固定值20磅，段落间距为段前0磅，段后0磅。

## 符号对照表

学位论文中符号代表的意义及单位（或量纲）的说明。符号对照表标题字体为黑体，字号为三号，居中排列，行距为固定值20磅，段落间距为段前24磅，段后18磅；正文内容字体为宋体，字号为小四号，行距为固定值20磅。

## 缩略语对照表

学位论文中缩略语代表意义的说明。缩略语按照英文单词首字母顺序排列，对照表标题字体为黑体，字号为三号，居中排列，行距为固定值20磅，段落间距为段前24磅，段后18磅；正文内容中文字体为宋体，字号为小四号，英文字体为Times New Roman，字号为小四号，行距为固定值20磅。

## 目录

目录是学位论文的提纲，是论文各组成部分的小标题，应分别依次列出并注明页码。各级标题分别以第一章、1.1、1.1.1等数字依次标出，目录中最多列出三级标题，正文中如果确需四级标题，用（1）、（2）形式标出。学位论文的前置部分（摘要、插图索引、表格索引、符号对照表、缩略语对照表）和学位论文的主体部分（正文、参考文献、致谢、作者简介）都要在目录中列出。

目录标题字体为黑体，字号为三号，居中排列，行距为固定值20磅，段落间距为段前24磅，段后18磅；目录内容中一级标题字体为黑体，字号为小四号，其余标题字体为宋体，字号为小四号。

## 正文

正文是学位论文的主体和核心部分。正文的一级标题居中排列，字体为黑体，字号为三号，行距为固定值20磅，段落间距为段前24磅，段后18磅；二级标题不缩进，字体为宋体加粗，字号为小三号，行距为固定值20磅，段落间距为段前18磅，段后12磅；三级标题缩进2字符，字体为宋体，字号为四号加粗，行距为固定值20磅，段落间距为段前12磅，段后6磅；正文内容字体为宋体，字号为小四号，行距为固定值20磅，段落间距为段前0磅，段后0磅。正文一般包括以下几个方面：

### 绪论

绪论是学位论文主体部分的开端，切忌与摘要雷同或成为摘要的注解。绪论除了要说明论文的研究目的、研究方法和研究结果外，还应评述与论文研究内容相关的国内外研究现状和相关领域中已有的研究成果；其次还要介绍本项研究工作的前提和任务、理论依据、实验基础、涉及范围、预期结果以及该论文在已有基础上所要解决的问题。

### 各章节

各章节一般由标题、文字叙述、图、表、公式等构成，章节内容总体要求立论正确，逻辑清晰，数据可靠，层次分明，文字通畅，编排规范。论文中若有与指导教师或他人共同研究的成果，必须明确标注；如果引用他人的结论，必须明确注明出处，并与参考文献保持一致。

（1）图：包括曲线图、示意图、流程图、框图等。图序号一律用阿拉伯数字分章依序编码，如：图1.3、图2.11。每一个图应有简短确切的图名，连同图序号置于图的正下方。图名称、图中的内容字号为五号，中文字体为宋体，英文字体为Times New Roman，行距一般为单倍行距。图中坐标上标注的符号和缩略词必须与正文保持一致。引用图应在图题右上角标出文献来源；曲线图的纵横坐标必须标注“量、标准规定符号、单位”，这三者只有在不必要标明（如无量纲等）的情况下方可省略。

（2）表：包括分类项目和数据，一般要求分类项目由左至右横排，数据从上到下竖列。分类项目横排中必须标明符号或单位，竖列的数据栏中不要出现“同上”、“同左”等词语，一律要填写具体的数字或文字。表序号一律用阿拉伯数字分章依序编码，如：表2.5、表10.3。每一个表格应有简短确切的题名，连同表序号置于表的正上方。表名称、表中的内容居中排列，字号为五号，中文字体为宋体，英文字体为Times New Roman，行距一般与正文保持一致。表格线统一用单线条，磅值为0.5磅。表格与正文之间一般应空一行。

（3）公式：正文中的公式、算式、方程式等必须编排序号，序号一律用阿拉伯数字分章依序编码，如：(3-32)、 (6-21)。对于较长的公式，另起行居中横排，只可在符号处（如：+、-、\*、/、< >等）转行。公式序号标注于该式所在行（当有续行时，应标注于最后一行）的最右边。连续性的公式在“=”处排列整齐。大于999的整数或多于三位的小数，一律用半个阿拉伯数字符的小间隔分开；小于1的数应将0置于小数点之前。公式的行距一般为单倍行距。

（4）计量单位：学位论文中出现的计量单位一律采用国务院1984年2月27日发布的《中华人民共和国法定计量单位》标准。

### 结论

结论是学位论文最终和总体的结论，不是正文中各段的小结的简单重复，应准确、精炼、完整，其中要着重阐述作者研究的创造性成果以及在本研究领域中的重大意义，还可提出有待进一步研究和探讨的问题。

## 参考文献

参考文献是文中引用的有具体文字来源的文献集合，博士学位论文参考文献一般不少于80篇，其中近5年的参考文献不少于20篇，硕士学位论文参考文献一般不少于30篇，其中近5年的参考文献不少于5篇。参考文献标题字体为黑体，字号为三号，居中排列，段落间距为段前24磅，段后18磅；参考文献若是中文文献，字体为宋体，字号为五号，若是英文文献，字体为Times New Roman，字号为五号。学位论文的撰写要本着严谨求实的科学态度，凡有引用他人成果之处，引用处右上角用方括号标注阿拉伯数字编排的序号（必须与参考文献一致），同时所有引用的文献必须用全称，不能缩写，并按论文中所引用的顺序列于文末。引用文献的作者不超过3位时全部列出，超过时列前3位，后加“等”字或“et al.”。参考文献的著录要符合《文后参考文献著录规则》（GB/T7714-2005）要求：

（1）期刊(报纸)参考文献：[序号] 主要责任者. 文献名称[文献类别代码]. 期刊(报纸)名, 年份, 卷(期): 引文页码.

（2）专著参考文献：[序号] 主要责任者. 专著名称[文献类别代码]. 其他责任者. 出版地: 出版单位, 出版年份.

（3）专利参考文献：[序号] 主要责任者. 专利名称: 国别, 专利号[文献类别代码]. 出版日期.

（4）技术标准参考文献：[序号] 起草责任者. 标准代号-标准顺序号-发布年. 标准名称[文献类别代码]. 出版地: 出版单位,出版年份.

（5）电子参考文献：[序号] 主要责任者. 题名[文献类别代码]. 获取和访问路径. [引用日期].

（6）会议论文集参考文献：[序号] 编者. 论文集名. (供选择项：会议名, 会址, 开会年)出版地: 出版者, 出版年份.

（7）学位论文参考文献：[序号] 主要责任者. 文献题名[文献类别代码]. 保存地: 保存单位, 年份.

（8）国际、国家标准参考文献：[序号] 标准代号. 标准名称[文献类别代码]. 出版地: 出版者, 出版年.

（9）报告类参考文献：[序号] 主要责任者. 文献题名[文献类别代码]. 报告地: 报告会主办单位, 年份.

参考文献著录中的文献类别代码：

（1）普通图书：M

（2）会议录：C

（3）汇编：G

（4）报纸：N

（5）期刊：J

（6）学位论文：D

（7）报告：R

（8）标准：S

（9）专利：P

（10）数据库：DB

（11）计算机程序：CP

（12）电子公告：EB

载体类型：

网络：OL

磁带：MT

磁盘：MK

光盘：CD

## 致谢

作者对完成论文提供帮助和支持的组织和个人表示感谢的文字记载。致谢标题字体为黑体，字号为三号，居中排列，行距为固定值20磅，段落间距为段前24磅，段后18磅；正文内容字体为宋体，字号为小四号，行距为固定值20磅，段落间距为段前0磅，段后0磅。

## 作者简介

对作者的简要介绍，主要包括个人基本情况、教育背景、攻读博士/硕士学位期间的研究成果等三个部分内容。攻读博士/硕士学位期间的研究成果是指本人攻读博士/硕士学位期间发表（或录用）的学术论文，申请（授权）专利、参与的科研项目及科研获奖等情况，分别按时间顺序列出。其中，发表论文、申请（授权）专利、科研获奖只列出作者排名前3名的，参与的科研项目按重要程度最多列出5项。作者简介标题字体为黑体，字号为三号，居中排列，行距为固定值20磅，段落间距为段前24磅，段后18磅。作者简介的正文内容严格按照本模板中的范例书写。

## 其他

学位论文中如果需要注释，可作为脚注在页下分别著录，切忌在文中注释；如果有附录部分，可编写在正文之后，与正文连续编页码，每一附录均另页起，附录依次用大写英文字母A、B、C……编序号，如：附录A、附录B等。

# 研究生学位论文的编辑、打印、装订要求

## 学位论文封面的编辑和打印要求

学位论文的封面由研究生院按国家规定统一制定印刷，封面内容必须打印，不得手写。

## 学位论文的版面设置要求

（1）行间距：固定值20磅（题名页除外）。

（2）字符间距：标准。

（3）页眉设置：单面页码页眉标题为章节题目，每一章节的起始页必须在单面页码，双面页码页眉标题统一为“西安电子科技大学博/硕士学位论文”，页眉标题居中排列，字体为宋体，字号为五号。页眉文字下添加双横线，双横线宽度为0.5磅，距正文距离为：上下各1磅，左右各4磅。

（4）页码设置：学位论文的前置部分和主体部分分开设置页码，前置部分的页码用罗马数字标识，字体为Times New Roman，字号为小五号；主体部分的页码用阿拉伯数字标识，字体为宋体，字号为小五号。页码统一居于页面底端中部，不加任何修饰。

（5）页面设置：为了便于装订，要求每页纸的四周留有足够的空白边缘，其中页边距为上3厘米、下2厘米；内侧2.5厘米、外侧2.5厘米；装订线为0.5厘米；页眉2厘米，页脚1.75厘米。

## 学位论文的打印、装订要求

（1）打印：学位论文必须用A4纸页面排版，双面打印；

（2）装订：依次按照中文题名页、英文题名页、声明、摘要、插图索引、表格索引、符号对照表、缩略语对照表、目录、正文、附录（可选）、参考文献、致谢、作者简介的顺序，用学校统一印制的学位论文封面装订成册。盲审论文必须删除致谢部分的文字内容（致谢标题须保留）以及封面和研究成果中的作者和指导教师姓名，研究成果列表中应体现作者的排序，如第一作者、第一发明人等。

## 其他说明

本规定由研究生院负责解释，从申请2015年9月毕业和授位的研究生开始执行，其它有关规定同时废止。研究生毕业论文撰写要求参照学位论文撰写要求执行。

# 图、表、公式示例

图：包括曲线图、示意图、流程图、框图等。图序号一律用阿拉伯数字分章依序编码，如：图1.3、图2.11。

每一个图应有简短确切的图名，连同图序号置于图的正下方。图名称、图中的内容字号为五号，中文字体为宋体，英文字体为Times New Roman，行距一般为单倍行距。图中坐标上标注的符号和缩略词必须与正文保持一致。引用图应在图题右上角标出文献来源；曲线图的纵横坐标必须标注“量、标准规定符号、单位”，这三者只有在不必要标明（如无量纲等）的情况下方可省略。

图与正文之间一般应空一行。



插图示例

公式：正文中的公式、算式、方程式等必须编排序号，序号一律用阿拉伯数字分章依序编码，如：(3-32)、 (6-21)。

对于较长的公式，另起行居中横排，只可在符号处（如：+、-、\*、/、< >等）转行。公式序号标注于该式所在行（当有续行时，应标注于最后一行）的最右边。连续性的公式在“=”处排列整齐。大于999的整数或多于三位的小数，一律用半个阿拉伯数字符的小间隔分开；小于1的数应将0置于小数点之前。公式的行距一般为单倍行距。

公式与正文之间一般应空一行。



表：包括分类项目和数据，一般要求分类项目由左至右横排，数据从上到下竖列。

分类项目横排中必须标明符号或单位，竖列的数据栏中不要出现“同上”、“同左”等词语，一律要填写具体的数字或文字。表序号一律用阿拉伯数字分章依序编码，如：表2.5、表10.3。

每一个表格应有简短确切的题名，连同表序号置于表的正上方。表名称、表中的内容居中排列，字号为五号，中文字体为宋体，英文字体为Times New Roman，行距一般与正文保持一致。表格线统一用单线条，磅值为0.5磅。

表格与正文之间一般应空一行。

表格示例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 馈电方式  电性能参数 | 探针 | 环形缝隙 | 探针和缝隙 | | 缝隙和CPW | |
| 探针 | 缝隙 | 缝隙 | CPW |
| 谐振频率 | 9.5 GHz | 8.8 GHz | 9.4 GHz | 9.8 GHz | 9.2 GHz | 9.3 GHz |
| 带宽  （|S11|<-10 dB） | 7.3% | 4.5% | 6.9% | 6.8% | 4.9% | 5.3% |
| 隔离度  （带内最差） | -16.5 dB | -17 dB | -31 dB | | -22 dB | |
| 方向图 | 不对称 | 对称 | 不对称 | 对称 | 对称 | 对称 |
| 交叉极化电平 | 高 | 低 | 高 | 低 | 低 | 低 |

计量单位：学位论文中出现的计量单位一律采用国务院1984年2月27日发布的《中华人民共和国法定计量单位》标准。

参考文献

普通图书示例：

1. 广西壮族自治区林业厅. 广西自然保护区[M]. 北京: 中国林业出版社, 1993.
2. 蒋有绪, 郭泉水, 马娟等. 中国森林群落分类及其群落学特征[M]. 北京: 科学出版社, 1998.
3. 唐绪军. 报业经济与报业经营[M] .北京: 新华出版社, 1999: 117-121.
4. 赵凯华, 罗蔚茵. 新概念物理教程: 力学[M]. 北京: 高等教育出版社, 1995.
5. 汪昂. (增补)本草备要[M]. 石印本. 上海: 同文书局, 1912 .
6. CRAWFPRD W, GORMAN M. Future libraries: dreams, madness, & reality[M]. Chicago: American Library Association, 1995.
7. International Federation of Library Association and Institutions. Names of persons: national usages for entry in catalogues[M]. 3rd ed. London: IFLA International Office for UBC, 1977.
8. O’BRIEN J A. Introduction to information systems[M]. 7th ed. Burr Ridge, III.: Irwin, 1994.
9. ROOD H J. Logic and structured design for computer programmers[M]. 3rd ed. Watertown: Brooks/Cole Thomson Learning, 2001.

论文集、会议录示例：

1. 中国力学学会. 第3届全国实验流体力学学术会议论文集[C]. 天津: 南开大学出版社,1990.
2. ROSENTHALL E M. Proceedings of the Fifth Canadian Mathematical Congress, University of Montreal, 1961[C]. Toronto: University of Toronto Press, 1963.
3. GANZHA V G, MAYR E W, VOROZHTSOV E V. Computer algebra in scientific computing: CASC 2000: proceedings of the Third Workshop on Computer Algebra in Scientific Computing, Samarkand, October 5-9, 2000[C]. Berlin: Springer, c2000.

科技报告示例：

1. U. S. Department of Transportation Federal Highway Administration. Guidelines for bandling excavated acid-producing materials, PB 91-194001[R]. Springfield:U. S.Department of Commerce National Information Service, 1990.
2. World Health Organization. Factors regulating the immune response: report of WHO Scientific Group[R]. Geneva: WHO, 1970.

学位论文示例：

1. 张志祥. 间断动力系统的随机扰动及其在守恒律方程中的应用[D]. 北京: 北京大学数学学院, 1998.
2. CALMS R B. Infrared spectroscopic studies on solid oxygen[D]. Berkeley: Univ. of California. 1965.

专利文献示例：

1. 刘加林. 多功能一次性压舌板: 中国, 92214985. 2[P]. 1993-04-14.
2. 河北绿洲生态环境科技有限公司.一种荒漠化地区生态植被综合培育种植方法: 中国, 01129210.5[P/OL]. 2001-10-24[2002-05-28]. http://211.152.9.47/sipoasp/zlijs/hyjs-yx- new.asp? recid=01129210.5&leixin.
3. KOSEKI A, MOMOSE H, KAWAHITO M, et al. Compiler: US, 828402[P/OL]. 2002-05-25 [2002-05-28].http://FF&p=1&u=netahtml/PTO/search-bool.html&r=5&f=G&1=50&co1=AND&d=PGOl&sl=IBM. AS. &OS=AN/IBM&RS=AN/IBM.

专著中析出的文献示例：

1. 国家标准局信息分类编码研究所. GB/T 2659-1986世界各国和地区名称代码[S]//全国文献工作标准化技术委员会. 文献工作国家标准汇编: 3.北京: 中国标准出版社, 1988:59-92.
2. 韩吉人. 论职工教育的特点[G]//中国职工教育研究会. 职工教育研究论文集. 北京: 人民教育出版社, 1985:90-99.
3. BUSECK P R, NORD G L, Jr. , VEBLEN D R. Subsolidus phenomena in pyroxenes[M]// PREWITT C T. Pyroxense. Washington, D. C. : Mineralogical Society of America, c1980: 117-211.
4. FOURNEY M E. Advances in holographic photoelasticity [C]// American Society of Mechanical Engineers. Applied Mechanics Division. Symposium on Applications of Holography in Mechanics, August 23-25, 1971, University of Southern California, Los Angeles, California. New York: ASME, c1971 : 17-38.
5. MARTIN G.. Control of electronic resources inAustralia[M]//PATTLE L W, COX BJ. Electronic resources: selection and bibliographic control. New York: The Haworth Press, 1996: 85-96.

期刊中析出的文献示例：

1. 李炳穆. 理想的图书馆员和信息专家的素质与形象[J]. 图书情报工作, 2000(2):5-8.
2. 陶仁骥. 密码学与数学[J]. 自然杂志, 1984,7(7):527.
3. 习亚洲地质图编目组. 亚洲地层与地质历史概述[J]. 地质学报, 1978, 3: 194-208.
4. DES MARAIS D J, STRAUSS H, SUMMONS R E, et al. Carbon isotope evidence for the stepwise oxidation of the Proterozoic environment [J]. Nature, 1992, 359: 605-609.
5. HEWITT J A. Technical services in 1983[J]. Library Resource Services, 1984, 28(3): 205-218.

报纸中析出的文献示例：

1. 丁文祥. 数字革命与竞争国际化[N]. 中国青年报, 2000-11-20(15).
2. 张田勤. 罪犯DNA库与生命伦理学计划[N]. 大众科技报, 2000-11-12(7).

电子文献(包括专著或连续出版物中析出的电子文献)示例：

1. 江向东. 互联网环境下的信息处理与图书管理系统解决方案[J/OL]. 情报学报, 1999, 18(2): 4[2000-01-18].
2. 萧钮. 出版业信息化迈人快车道[EB/OL]. (2001-12-19) [2002-04-15]
3. CHRISTINE M. Plant physiology: plant biology in the Genome Era[J/OL]. Science, 1998, 281:331-332[1998-09-23].
4. METCALF S W. The Tort Hall air emission study[C/OL]//The International Congress on Hazardous Waste, Atlanta Marriott Marquis Hotel, Atlanta,Georgia, June 5-8, 1995: impact on human and ecological health[1998-09-22].
5. TURCOTTE D L. Fractals and chaos in geology and geophysics[M/OL]. New York: Cambridge University Press, 1992[1998-09-231.
6. Scitor Corporation. Project scheduler[CP/DK]. Sunnyvale, Calif. : Scitor Corporation, c1983.

致谢

本论文是在导师的悉心指导下完成的，从论文的选题到论文的撰写，无不渗透着导师的心血，……值此论文完稿之际，谨对导师的辛勤培育以及谆谆教诲表示最衷心的感谢!

作者简介

##### 基本情况

张三，男，陕西西安人，1982年8月出生，西安电子科技大学XX学院XX专业2008级硕士研究生。

##### 教育背景

2001.08～2005.07西安电子科技大学，本科，专业：电子信息工程

2008.08～ 西安电子科技大学，硕士研究生，专业：电磁场与微波技术

##### 攻读硕士学位期间的研究成果

###### 发表学术论文

1. XXX, XXX, XXX. Rapid development technique for drip irrigation emitters[J]. RP Journal,UK.,2003,9(2): 104-110.(SCI: 672CZ, EI: 03187452127)
2. XXX, XXX, XXX. 基于快速成型制造的滴管快速制造技术研究[J]. 西安交通大学学报, 2001, 15(9): 935-939. (EI: 02226959521)

###### 申请（授权）专利

1. XXX, XXX, XXX等. 专利名称: 国别,专利号[P]. 出版日期.

###### 参与科研项目及获奖

1. XXX项目, 项目名称, 起止时间, 完成情况, 作者贡献。
2. XXX, XXX, XXX等. 科研项目名称. 陕西省科技进步三等奖, 获奖日期.