北京大学

硕士研究生学位论文

题目：基于 HTML 渲染引擎的电子合同托管平台的设计与实现

姓　　名：	李戬
学　　号：	1201210649
院　　系：	软件与微电子学院
专　　业：	软件工程
研究方向：	电子商务与物流
导师姓名：	李杰 教授

二〇一五 年 七 月

摘要

近几年来，随着互联网金融的飞速发展，网贷平台呈现爆发式增长趋势。绝大多数网贷平台在撮合出借人和借款人达成交易之时，需要借款人、出借人、网贷平台甚至担保方共同订立一份电子合同。该合同明确借款人、借款金额、还款方式等相关信息，规定了合约各方的权利及义务等。因网贷行业的特殊性，借款的还款周期一般比较长，合同的内容也不尽相同，“电子合同保存”（亦称“证据保全”）对于网贷平台和用户就显得非常重要。但目前网贷平台在野蛮式增长过程中存在一些不规范之处，例如电子合同的订立流程设计不健全，加之网贷平台本身也存在跑路风险，在线电子合同托管的需求便应运而生。

本项目旨在分析、设计和开发一个遵循《合同法》、《电子签名法》与《电子合同在线流程规范》的电子合同托管平台。主要过程为网贷平台通过后台实时 post 请求，向托管平台发送电子合同签名请求，附加合同文本数据（以 HTML 文本形式发送）以及签名信息。合同托管平台接收、处理合同文本数据，验证签名请求后，返回请求结果。托管平台解析结果之后，如果成功，继续通过浏览器前台跳转方式请求托管系统显示合同全文，并请求用户授权签名。用户输入签名密码授权后，利用数字签章技术对合同文本进行签名，托管平台使用 HTML 渲染引擎技术创建加密的 PDF 电子文档，并存储在安全的本地服务器上，发送给网贷平台的用户签名结果。最后网贷平台根据返回的签名结果做相应处理操作。用户和网贷平台可登录托管平台，查询、浏览和下载所属的电子合同文件。

本论文主要贡献包括：

（1）针对电子合同托管流程，改进了一款轻量、高效、支持更多并发的 HTML 渲染引擎。根据业务实际需求，提高渲染速度，支持不严谨的 HTML 代码和常用的 CSS 样式；

（2）分析了电子合同托管业务需求，提出、设计和优化了一套合同托管业务流程，具有比较好的应用价值。

关键字：合同托管，Spring 框架，CSS，HTML 渲染引擎

Design and Implementation of E-contract Depository Platform Based on the HTML Renderer

Li Jian (Software Engineering)
Directed by Li Jie

ABSTRACT

In recent years, with the rapid development of Internet Finance, the peer-to-peer lending platform (P2P platform) comes to an explosive growth. When the P2P platform makes a deal, the lender, the borrower, the P2P platform, sometimes the guarantor together, should set up an e-contract. The e-contract is to confirm the information that who the borrower is, how much to loan, how to repay and the rights and obligations each party takes. Because of the peculiarities of the P2P industry, the repayment cycles are usually quite long and the contents of e-contract are always different. So the preservation of e-contract is important to the P2P platform and the users. However, there still have been many irregular operations when signing e-contract. And there is a risk that the P2P platform may “run away”. Online e-contract depositing demand therefore emerges.

This project aims to research and develop an e-contract depository system, making sure it’s law compliant. The main processes include the following steps:

P2P platforms send request of signing to the depository system, together with e-contract text data and signature information. E-contract depository platform receives and processes the data to verify the signature and return the result. After the merchant system successfully analyses the result information, the user’s browser jumps to e-contract depository platform and the platform indicate the contract text and then ask the user to authorize to sign by inputting the signature password. Then the users sign the e-contract using the digital signature technology. After that the depository platform will generate an encrypted PDF file with HTML Renderer and store in a safe local server. If finished, the depository platform will return the successful result. Finally, P2P platforms response to the result again and do some process. All users and P2P platforms can query, browse and download their online e-contract by visiting the depository platform.

The major contributions of this dissertation include:

(1) Improving a lightweight and efficient HTML Renderer engine supporting more

concurrent and to meet the actual needs of the business. It is able to parse sloppy HTML code and CSS style.

(2) Analyzing and designing a set of e-contract depository process, which has wide application.

KEY WORDS: E-contract Depository, CSS, Spring Framework, HTML Renderer

目录

图表目录

第一章 绪论

1.1　课题背景与意义

近几年来，利用互联网先进思维和技术改造传统金融行业的呼声日益强烈，随着互联网金融行业的兴起和飞速发展，P2P 网贷平台数量爆发式增长，对传统银行业存贷业务冲击日渐显现。

无论传统金融行业还是互联网金融行业其核心都是信用和安全。目前，绝大部分网贷平台在撮合借款交易达成之后，需要借款人、出借人、网贷平台甚至担保方共同订立一份电子合同。该电子合同明确了借款人、借款金额、还款方式等相关信息，规定了交易各方享有的权利和必须履行的义务等。但是网贷行业具有其特殊性，一笔借款的还款周期一般都会比较长，因此需要长时间保存电子合同文件。总之，电子合同的保存和管理对于网贷用户和借贷平台非常重要。

目前，绝大多数网贷平台订立电子合同的过程并不规范，第三方提供在线电子合同托管的服务需求便应运而生。市场对于第三方合同托管的需求非常迫切，特别是在证据保全、互联网金融、电子商务、档案管理等相关行业。但是当前，国内实际成熟的应用案例比较鲜见。

因此有必要深入研究一下合同托管这一领域，分析其业务需求，突破技术难题，做好功能模块的设计与开发，搭建一个第三方电子合同托管平台，助推互联网金融及相关行业更好、更快、更安全的发展。

1.1.1　互联网金融的发展

互联网金融主要是指以互联网技术和移动通信技术为技术基础，利用现代互联网资源实现融资、支付以及信用中介，这是一种全新的金融模式，是传统金融业与互联网精神结合新兴的领域。现阶段主要有六种商业模式：第一种是第三方支付模式，以支付宝、汇付天下为代表；第二种是依托大数据技术进行征信的模式，以蚂蚁金服、鹏元征信为代表；第三种是渠道业务模式，以京东金融、91 金融超市为代表；第四种是虚拟货币模式，以比特币（Bitcoin）、莱特币（Litecoin）为代表；第五种是 P2P 网贷模式，以人人贷、宜信为代表；第六种是众筹模式，以众筹网、点名时间为代表。

自从余额宝诞生之后，互联网金融行业步入快速发展的轨道，互联网金融产品日益获得越来越多互联网用户，特别是年轻用户的青睐，人们慢慢开始学习使用互联网金融产品进行理财投资。当前各式各样的互联网金融平台及产品纷纷涌现。按照互联网金融平台（产品）目标主要分为：（1）为用户提供借贷理财渠道的金融中介产品；

（2）利用大数据技术收集用户个人学历、社交、经济等信息并提供个人征信服务的产品；（3）满足用户金融信息、金融产品搜索的金融信息中介平台。互联网金融创新不仅仅是金融模式的创新，较传统金融行业，其获得竞争优势的一个重要方面是良好的用户体验[1]。总之，在移动支付、社交网络、搜索引擎和云计算等现代信息科技推动下，个体直接金融交易这一人类最早金融模式会突破传统的安全边界和商业可行边界，焕发出新的活力[2]。

近年来，随着互联网技术的发展，小额信贷与网络技术结合形成的 P2P 网贷平台（下文统称“网贷平台”）纷纷涌现，P2P 网络借贷指个体与个体之间通过互联网直接实现的借贷[3]。自 2005 年以来，P2P 借贷服务行业已经在世界各地蓬勃发展。据极客网统计，截至 2014 年 12 月，我国网贷平台数近 3000 家，仅 2014 年就增加近千家。2014 年市场总交易额预计超过 1000 亿元。2015 年，网贷平台数量和交易总量还将继续增加，但随着监管收紧、竞争加剧、行业微利的快速到来，网贷平台的运营成本大大增加，P2P 网贷平台风险频现[4]，截至 2015 年 2 月行业累计问题平台达到 494 家，2015 年将有大批存在风险和违约的平台被淘态。

1.1.2 在线电子合同托管需求

在电子商务时代，用户在电商平台选购商品，然后在线付款或者线下到付，所选商品通过快递物流最终配送到用户手中。这一过程中，需要用户和电商平台订立了在线买卖合同，为了追求用户购物体验，订单和合同在实际操作过程中和技术层面并没有明显区分，用户对合同存在的感知比较弱。用户未收到商品或者收到的商品有质量问题时，可以直接向该电子商务平台投诉。

电子商务蓬勃发展的时期，对于合同托管的迫切需求尚不明显，因为电子购物的周期一般比较短，商品客单价值相对比较小，合同信息一般比较简单、固定和明确，电子商务平台为了维护自身的信用，窜改合同信息成本高昂。

随着互联网金融行业的爆炸式发展，越来越多的互联网金融产品和服务在被用户使用时，平台必须与用户（借款人、投资人）订立具有法律效力，能够维护平台和用户自身利益的合同。合同是交易的桥梁，是商务活动的核心[5]。订立合同既能保护借款用户和投资用户的利益，也能维护网络服务平台的信誉和安全。在互联网金融时代，网贷平台发生借款还款的周期一般比较长，短则数月，多则数年，一般交易发生金额比较大，借款表现出高风险高收益的特点，借款用户和出借用户不能仅仅依靠网贷平台保存合同，必须亲自持有合同，以备合同违约后进行司法诉讼。电子合同、交易记录、资金来往凭证等都可以作为司法诉讼的凭证。电子合同最好由用户持有和保存，交易记录一般都是由交易平台掌握发布，资金来往凭证则需要第三方支付平台或者传统银行的金融机构出具。

在网贷平台发展初期，平台借款业务量较少，业务开展主要集中在线下人员能够覆盖的地区，网贷平台主要目标是发展和挖掘借款客户和出借客户为主。具体借款业务中，网贷企业的信审人员依据收集获得的借款用户资料，审核借款人的借款用户的信用信息，审核通过放款。放款前，网贷平台线下的业务工作人员会与借款用户通过面签形式签订纸质版本的借款合同协议，完成借款交易。借款客户按照合同约定，通过线上银行转账或者线下还款的形式按时还款。当业务量少时，面签成本尚可接受，但是随着网贷平台向全国范围迅速扩张,传统的线下面签的难度加大,成本迅速上升，线上远程签订合同日渐成为网贷平台下一步发展的必然趋势。

当今，网贷平台群雄逐鹿，各个网贷平台的业务流程存在众多不成熟的问题和不规范的地方。绝大部分互联网金融平台都将研发资源集中在网贷产品的设计、研发，提升用户体验、优化网贷流程和加强市场运营上，对于金融产品业务和营销规范上注意力不够，很多平台与用户拟定的合同存在霸王条款，绝大部分平台与用户订立合同的过程过于简单形式，没有完全遵守中华人民共和国商务部 2013 年颁布的《电子合同在线流程规范》（下文简称《合同规范》）中针对电子合同订立和存储的有关规定，存在侵害用户的权益的风险。未来平台在运营中，一旦发生部分大规模借款客户停止还款，信用违约，网贷平台将承受巨大信用压力，面临轻则产品下线，重则公司倒闭的风险。如果发生网贷公司索要比当初约定的更高额的利息的情况，而且用户未能备份最初在线确认的合同，那么用户将很难通过司法途径维护自身合法权益。这些情况对于网贷用户极为不利。部分网贷平台有雄厚的资金、或者政府、金融部门背景做背书，可以依靠自身信用承诺保护用户权益，但仅仅依靠平台自身信用背书，并不能够保证用户自身的利益就绝对不受到侵害。网贷平台作为一个信息中介平台，是连接投资人和借贷人之间的桥梁，一旦平台跑路，投资人就可能血本无归。最近一系列投资人维权案例中，关于平台跑路导致投资人无法追回借款的案例屡见不鲜。无论网贷平台出现何种问题，都不应该影响借贷双方的借贷关系。除了通过债务流转的方式解决债务问题，电子合同保全的技术手段也是必不可少的。

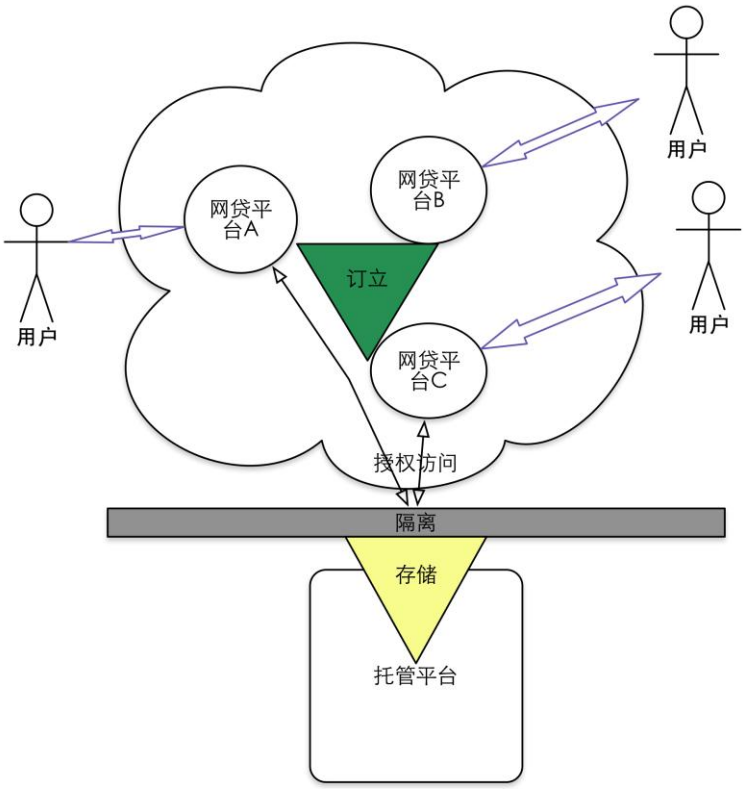


图 1.1　合同托管平台作用图

因此，需要借助第三方合同托管平台维护用户的合法利益。如图 1.1 所示，托管平台核心部分和互联网中的各个网贷平台是隔离的，只有托管平台用户授权才能访问托管平台系统相应的电子合同信息，以保证用户合同的安全，同时遵循《合同规范》核心要求，合同的订立和存储过程分开进行，由不同主体承担，网贷平台负责订立过程，托管平台负责存储过程。即便网贷平台跑路，合同仍然存在，借贷双方仍能够按照合同约定行使自己的合法权益。

1.1.3　现状分析

当前国际学术研究方面，主要关注于电子合同谈判和签署系统的建模上[6]、电子合同的可信第三方存储（TTP、Trusted Third Party）以及从博弈论角度探讨合同订立是否安全。2005 年以后国外研究转向电子合同 Web 服务角度。2007 年，T. Kwok 等人设计了一个基于 Web 和电子邮件驱动的电子合同管理系统，并应用于 IBM 几个试点项目，该系统支持企业内部的工作流程[7]。2012 年，J. Yang 等人解释了当前中国商务谈判的现状，分析了应用电子签名在商业活动推广的困难，提出了先锋网络平台对电子合同的制作及其应用[8]。2013 年，基于 fair non-repudiation 协议的自动处理机制，

K. Chatterjee 等人展示了通过指定参与者的目标，清楚地解释了作为“三人博弈”中的安全均衡策略，最终得到结论是系统安全可以被有效的保证[9]。国内研究主要关注企业的合同管理系统的设计研发、合同订立中的法律问题研究方面，2009 年杨晨阳基于 EXT 和 Ajax 技术开发一个公司内部使用合同管理系统[10]，2014 年段苏设计和实现了一个海尔公司内部使用的合同管理系统，作为构建智能企业合同管理的基础[11]。

目前业界实践中，将书面形式的文件与存储在电脑中的电子文件有机结合起来尚存在一些问题需要解决，用户体验欠佳，所以纸质合同仍然占据合同订立保管市场的主导地位。数字签章技术的使用正在改变这种现状，使用电子签章以后的文件将被锁定，除电子密钥持有者本人以外都无法篡改里面的任何内容，只有密钥持有人才能进行签章操作，因此具有抗抵赖性。

互联网上已陆续出现一些以律师事务所为背景的商业合同托管平台。目前这些商业合同托管服务网站提供的服务主要包括合同线上承揽业务，线下提供托管，司法咨询、诉讼等相关服务。从当前国内最新的应用情况来看，第三方在线合同托管是一个比较新兴的行业，成熟的应用案例比较少见，行业整体还处于探索阶段。据 2014 年 12 月 12 日《中国日报网》报道，钱盒子（深圳一家行业领先的网贷平台），与华商律师事务所举行了“第三方合同托管”签约仪式，双方将合作推出法律咨询、合同审查、证据托管、法律保全四项措施，以此保障用户的资金及交易安全。电子合同不应该存放在政府机构，政府不应为商业行为（借贷）背书，在目前提供服务的商业公司很少的情况下，在线合同托管业务平台与律师事务所合作是一个不错的选择，但是与律所合作也存在一些问题：一方面，律所目前缺乏合同托管行业实操经验，也没有形成可以借鉴参考的业务流程规范，服务意识欠缺；另一方面，律所自身 IT 队伍和技术力量的建设相对比较落后，无法提供优质的合同托管服务。因此，建设真正意义上遵守合同托管业务流程规范化的，依靠信息化支撑的，面向互联网服务的专业合同托管商业平台势在必行。

1.2　课题来源、研究目标、研究内容

1.2.1　课题来源

本课题来源于某企业内部的一个实际项目，在产品设计与研发实践过程中，团队研究讨论过部分网贷平台的业务流程及产品设计问题，发现就合同保存这一领域来说，已有的平台设计往往存在明显的安全缺陷，不符合《合同规范》要求。与之前研究者的侧重有所不同，本文侧重于从与互联网实际应用角度，梳理分析和设计在线电子合同托管的一套业务流程和业务系统。

一般的网贷平台在合同确认过程中，只要用户确认合同，系统就会通过自身建立

的资金池账户或资金托管平台（例如：汇付天下、易宝支付和汇付宝等）将钱转入借款用户的账户。合同本身的作用被弱化了，其保障用户权益，特别是出借用户的作用没有体现出来，埋下了信用安全隐患。因此在产品研发设计过程中，产品团队和技术团队提出建设一个独立的在线合同托管系统，该系统运行在独立的网络和服务器上，专门存储用户的合同。该系统应具有独立的用户管理系统，与公司自身网贷平台系统完全隔离，按照《合同规范》要求进行设计和研发，以保护用户的合法权益。未来待系统成熟之后，可以将在线合同托管系统完全独立出来，成立专门公司运作该平台，并为其它网贷平台提供合同托管服务。

1.2.2 研究目标和内容

本课题主要针对电子合同托管业务特点与需求进行研究。研究发现，系统需要解决的核心问题是：

1) 如何将各个网贷平台提供的各种不同合同内容的，以 HTML 形式描述的文本高效地转换成 PDF 文件；
2) 合同文件订立和存储中存在的各种安全问题。

本人在整个项目中主要承担了系统的业务需求设计、系统的整体架构的创新性搭建以及构建 HTML 的渲染引擎等任务。研究和开发过程中，主要改进了一款适应业务需求的 HTML 渲染引擎，解决 CSS 样式渲染速度问题、复杂对象的排版问题、HTML 不严谨代码检查校验问题和并发问题等。在此技术上，研发以用户为中心的，具有良好用户体验的，遵守《合同规范》要求的电子合同托管平台。该平台设计的要求是实现信息传输的保密性、数据交换的完整性、发送信息的不可抵赖性、合同订立者身份的真实性，业务运行的可靠性。

本论文主要内容是通过对在线合同托管需求分析，设计和实现一个适用于企业合同托管需求的业务平台。该平台可提供已订立的在线合同的保管和查询服务。该系统能够生成 PDF 格式的合同文档，向外开放接口，提供合同模板设置、安全设置等功能。本论文的重点在于将合同托管业务特点和业务流程、合同存储与合同查询下载有机结合，创建一种更为有效的订立和存储合同的方式，方便企业和用户。本论文主要介绍了基于 HTML 转 PDF 引擎的电子合同托管平台的设计与实现，应用于某互联网金融平台内部测试运行，未来目标是对外开发服务接口。

1.3 论文的组织结构

本文总共分为 6 章：

第 1 章：主要介绍了合同托管系统研发的背景及意义，分析了该领域的现状，概

述了主要研究内容，以及论文篇章结构的组织安排。

第 2 章：主要介绍了电子合同的概念和特征，交易合同和服务合同的区别，明确了网贷合同的范畴，详解了电子合同的规范要求，电子合同订立过程中存在的问题。紧接着介绍了电子合同托管系统开发中应用到的相关技术，包括框架技术，安全技术和关键技术。

第 3 章：主要从电子合同托管业务形成入手，分析了电子合同托管系统的业务总体业务需求，具体功能模块的功能性需求、安全性需求以及系统的非功能性需求。

第 4 章：主要说明了系统设计遵循的原则，进行了系统的总体设计，包括系统的总体框架，系统的部署设计和实现，介绍了系统的 E-R 模型设计，接着详细剖析了 HTML 渲染引擎的设计与实现，接口设计以及关键业务的设计与实现，包括托管业务与消息业务。

第 5 章：通过具体功能测试、整体系统测试来分析系统地正确性和稳定性，介绍了测试工具和测试结果等。

第 6 章：对全文进行总结，对于系统在具体实施过程中的表现以及存在的问题进行总结，然后并对课题后续可以持续研究方向进行了展望。

第二章 托管业务和应用技术

2.1　电子合同托管

电子合同托管顾名思义就是将电子合同委托给可信的第三方组织或机构（TTP）保存管理。电子合同管理与电子合同托管概念之间有微小的差别，前者强调由合同拥有者管理合同，后者强调寻求第三方进行协助管理，第三方是业务参与者。电子合同托管也可以称之为一种电子证据托管，通俗上来讲就是把借贷双方的债权文件交由第三方具有证据托管资质的平台来保管，而这个第三方也可以协助投资人通过法律的程序追回欠款。概括说第三方托管机构提供的服务有法律咨询、合同审查、证据托管、法律保全四项措施。从保护投资者的角度来看，是投资者权益预先保护机制的建立；从司法角度来看，是证据保全的前置；从金融监管的角度来看，是处置机制的建立。

2.1.1　电子合同基本概念和特征

电子合同，指在网络条件下，合同当事人之间为了实现一定的目的，通过电子邮件或电子数据交换，明确相互权利义务关系的协议或者契约[12]。合同当事人是平等主体的自然人、法人或其它组织。数据交换以数据电文作为载体。合同订立借助电子通信手段进行。合同订立方式一般采取要约和承诺方式。与传统合同相比，由于电子合同缔约方式和存储方式的特殊性，使电子合同出现许多独特的特征：

1) 电子合同当事人身份的不确定性。传统合同通常是在当事人双方均在场的情况下，通过双方当事人面对面的商谈和讨价还价，缔结合同，合同当事人的身份非常容易识别。而电子合同的当事人主要大多通过互联网平台进行要约和承诺，就合同条款达成一致，在绝大多数情况下是未见面就订立合同，其对合同签字的完成或对合同身份的确认需要借助第三方组织或机构帮助建立起来的电子签名、电子认证和客户信息系统的来完成。
2) 电子合同表现特性的无形性。传统合同的形式主要由两大类，即书面形式和口头形式[13]。书面形式经由各方当事人亲自签字或盖章，体现为一种有形物，可以由当事人各方单独保存。而电子合同体现为数字交换报文，具有无形性，电子合同数据存储在介质中，无法直接识别，极易复制。
3) 电子签名的可替代性。在传统合同中，当权利义务确定后由当事人亲自在合同上签名盖章，以此来确认当事人对合同内容的认可，使合同产生法律效力。在电子合同中，当事人是无法通过这种方式来完成该程序的，取而代之的是当事人的电子签名，即由计算机生成数据，通过对该数据的识别来确认合同

4) 当事人的身份。在2005年4月1日颁布实施的《中华人民共和国电子签名法》，其第十四条明确指出“可靠的电子签名与手写签名或者盖章具有同等的法律效力”[14]。

5) 电子合同内容的不稳定性。当事人意思一致后，订立书面合同，然后各持一份合同的原件，原件内容一旦被修改，两份合同文本将有明显的区别。而电子合同采用电子数据交换的方式和手段订立，合同内容以电子形式储存介质中，电子数据在网络上进行传输时，极易被截取，攻击出错或者篡改、从而使得合同的内容发生变化。总之，电子合同的内容比传统合同更加容易被篡改而不留痕迹，不容易被发现[15]。

2.1.2 在线服务合同与在线交易合同区别

电子合同按照用途性质可分为在线交易合同和在线服务合同。

在线服务合同是指网络运营商通过网络为用户提供有偿或无偿服务的合同。在线交易合同是指通过网络转移财产权利的电子合同，可以分为有形商品买卖合同与无形信息产品买卖合同。在线服务合同与在线交易合同的关系，可以分两种情况[16]：一种是信息产品交易和网络服务合同；另一种是网站作为交易中介的服务合同。这两种合同属于不同的法律关系范畴。

一般互联网平台均具有信息服务功能，其提供的服务各式各样，有免费服务也有收费服务，部分服务是为了吸引用户通过其网站交易或购买产品，部分服务是为了收取佣金撮合交易双方。当一个网站提供内容服务，比如付费浏览、在线视听、有偿软件下载(许可或买卖)之时，在线信息产品似乎构成网站服务合同的内容。实际上是，网站向用户提供包括信息产品交易在内的综合性服务。如果用户从该网站购买货物、信息服务，那么网络服务合同也是买卖合同或信息服务合同的组成部分，二者不能截然分开。但一般这两者毕竟是两个法律关系，一些网站意愿将两者混同，利用本身的一些服务条款规避其在买卖或信息许可中的责任。国内网站将二者捆绑在一起已经成为业界的通行做法。最重要的是，如果某人从网站下载一个软件(即是一种购买行为)，那么他可能受买卖合同、网站服务合同、软件的许可协议同时约束，此时发生纠纷诉讼到法院，如果这些条款均适用，那么消费者就可能处于非常不利的地位。

在互联网多数应用场景中，网络平台主要作为在线信息产品或货物交易的中介。此时在线交易双方尽管与网站可能均有服务关系，但是，在线交易合同关系的当事人绝不是网络平台，网络平台只是交易当事人之外的第三人。此时网络平台的服务条款对交易双方不一定有强约束力。理论上而言，网络服务合同只界定每个交易当事人与网站之间的权利和义务关系，而不界定交易双方当事人，此时，网络服务合同对交易双方当事人没有约束力。因此用户在使用网贷平台提供的借贷服务时要格外注意合同

条款内容是否有针对所有用户都应当遵守的规则。如果存在，网站上服务条款对双方就有了约束力。

电子商务交易过程中订立的合同一般属于在线交易合同范畴。有形商品的在线销售可以在网上完成合同的缔结、货款的支付和相关信息的传递，但仍需利用传统物流配送渠道(如 EMS、顺丰快递等)将商品送至买方手中。无形信息产品（软件、多媒体作品、数字化文字作品等数字产品），均可通过网络传输、下载实现买卖合同标的物的交付。在网贷平台交易过程中，则涉及资金财产权的转移，但网贷平台主要提供有偿中介服务，借贷双方应当格外注意在网贷平台订立的合同条款内容。

2.1.3 在线电子合同规范

电子合同托管系统研发必须严格遵守所在地国家的相关法律法规。

新《合同法》已经颁布实行，明确电子合同与传统合同具有同等的法律效力。商务部《合同规范》明确定义了电子合同的订立系统，电子合同第三方存储服务商、电子签名与认证服务提供商等合同在线订立过程中参与的系统角色，如图 2.1 所示。订立系统负责订立合同，然后交由存储服务商管理，订立系统查询需要通过存储服务商提供的接口，两者之间的通信，需要电子签名与认证提供商提供认证服务，保障安全。规范规定：

1） 订立系统和第三方存储系统必须严格保守合同内容信息，不得查看、披露或公开电子合同的内容；未经缔约人准许，不得公开和向第三人披露其身份信息，法律法规另有规定的除外；

2） 电子合同第三方存储服务商不得同时提供电子订约系统服务。电子合同订立系统设立人和电子合同第三方存储服务商应当严格执行国家计算机网络系统安全规定，保证系统的安全运行。

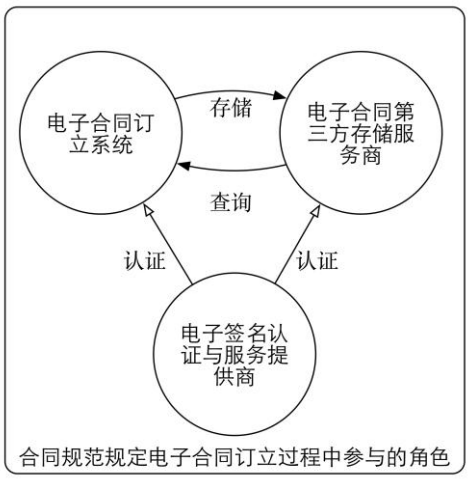


图 2.1　电子合同订立过程参与角色关系图

《合同规范》规定电子合同订立系统应当：允许使用人引入合同范本，能够按照使用人要求的格式生成符合法律要求的合同文本;允许进行在线谈判和合同文本修改;实现合同信息和谈判信息的实时传输、备份。信息数据可追溯、可复制；合同文本应采用电子签名或其它方式以查验合同内容是否完整及改动等。签名和认证服务提供商电子应当遵守《电子签名法》和国务院信息产业主管部门的相关规定。同时规定第三方存储系统：

1) 合同存储服务商应具有固定的经营场所和满足电子合同存储要求的物理环境，具有符合国家有关安全标准的技术、设备和管理制度，配备合格的管理人员；
2) 电子签名与认证服务提供商、合同存储服务商和其它辅助服务商均应在电子缔约系统首页公示其身份信息和业务规则，告知使用人操作方法和相关风险。

2.1.4 订立存储中可能存在的问题及解决之道

在互联网金融借款撮合行为一般视为借款双方在线发生的金钱交易行为。在线交易的合同主体应当遵循三个基本原则：第一，民事主体真实原则；第二，民事主体资格法定原则；第三，民事主体公示原则[17]。

网贷平台在订立存储合同过程中可能存在的问题：

1) 网贷平台主体真实性和合规问题，某些 P2P 网贷平台存在假借保护借款人信息名义，虚构或者伪造借款主体信息，例如伪造营业执照、伪造身份证明、或者虚构借款人家庭信息等来骗取出借人合法资金的行为。某些上线运营的 P2P 网贷平台，甚至根本没有法人资格，不符合商法规定。因此可由托管平台审核网贷主体的真实性；
2) 信息公示问题，在线交易中，一般会将当事人信息模糊化。网贷平台作为中介，加剧借贷双方信息的不对称。信息公示就是将面纱揭开，展示参与各方真实身份。因此可以由托管平台提供当事人信息查询功能，对当事人的身份及相关资料予以公示，查询必须符合资格并获取相应权限；
3) 平台跑路问题，网贷行业尚处三无状态，缺乏监管，个别平台被爆“跑路”或出现提现困难给整个行业的发展蒙上了一层阴影，引发了一系列信任危机。有些平台采用资金池模式，导致在平台发生违约风险时，网贷平台率先卷款跑路。因此可由托管平台协助借贷双方继续履行合同；
4) 合同稳定性的问题，关于电子合同的有效性已经被法律认可，但是电子合同所依赖的电子数据具有不稳定性。现阶段的电子合同，无论是引入签名或认证技术，都很难无条件成为认定全部借款事实的直接且充分证据。因此可由托管平台作为证人；

5) 业务流程问题，现有的合同订立存储流程极力模仿书面形式，利用 Javascript 技术实现动态签名，图层叠加，更多地是给以用户心理安慰，如图 2.2（合同内容进行了高斯处理），并不能保证合同托管绝对安全。用户手写签名的过程实际上增加了用户的操作负担，牺牲了用户体验，托管平台提供的签名密码验证简化流程，保证安全；
6) 技术问题，网贷平台信用审核技术不过关，平台建设技术不成熟，可能引起平台发生系统性风险。网贷平台需要自身加强技术建设，将 PDF 文件生成和数字签名交由托管平台完成，减轻网贷平台的技术压力。

图 2.2　某合同订立系统合同实例示意图

2.2　应用的框架技术

系统采用 Java 编程语言开发，采用了 JEE 框架技术。系统使用业界通用的、成熟、可靠的 Spring 框架快速迭代开发。数据库服务使用开源 MySQL 项目。Web 框架使用 SpringMVC，遵循 MVC 框架模式。系统使用 Mybatis 作为 ORM 中间件，管理与数据库的连接和操作。身份认证层使用 Shiro 这个开源权限验证框架搭建系统权限认证模块，显示层使用 JSP（Java Server Pages）技术和 Tiles3 布局框架。下面主要介绍一下：JEE 架构、Spring 框架，Mybatis 中间件，Tiles 布局框架和 Shiro 权限框架。

2.2.1　JEE 架构

JEE，以前叫 J2EE，即 Java2 平台企业版（Java2Platform Enterprise Edition）。它是一种能简化企业解决方案的开发、部署和管理相关的复杂问题的体系结构[18]。JEE 非常适合于搭建互联网金融系统和相关安全系统，具有安全、灵活、高效和易于维护的特性。JEE 拥有了一套通用的标准和规范，由众多组件构成，具有很高的可移植性、

可靠性和可复用性，这些标准和规范应用于 JEE 架构下的各个组件，服务和层次中，为 Web 开发者提供了“砖头”，依靠这些标准和规范，JEE 架构得以存在于不同的平台之间，实现系统之间，组件之间良好的兼容性。本系统在开发中使用了 JDK 提供的 JMX、DOM 等标准技术。

2.2.2 Spring 框架

Struts2、Hibernate、Spring 组合的 SSH 开发框架是一个流行的 JEE 应用开发框架，Struts2 提供 MVC 框架，Hibernate 提供 ORM 中间件，Spring 管理前两者。这一框架较为复杂，降低了开发效率，所以本项目开发中，使用 SpringMVC 替代 Struts2，Mybatis 替代 Hibernate。

Spring 框架是 Rod Johnson 等开发的，用于支持 JavaBean 构件运行的容器[19]。Spring 当前最新版本为 4.1。Spring 是一个开源轻量级应用框架，主要目标为减轻企业应用开发的复杂性[20]。Spring 框架的核心特征是控制反转（IoC）和面向切面（AOP）。它可以解决在 JEE 应用开发过程中出现的诸多难题，同时也提高了软件开发的效率和质量。在 Spring 出现之前，企业级开发一般都采用 EJB 框架进行开发的，EJB 是一个侵入式的框架，编程中用 EJB 提供的功能，就必须在代码中体现出来，比如继承一个接口，声明一个成员变量等。Spring 提供了事务管理、声明式事务、持久化和分布计算等等支持，大大“简化”了企业级应用的开发。该框架的最大优点是分层化、模块化。Spring 的用途除了开发服务器端，也可以做任何 Java 应用开发的框架，具有简单、可测试和松耦合等优点，如今的 Spring 框架体系已经异常庞大，由 Boot、Batch、Mobile、Social、Web Flow 等众多项目组成，选择版本，添加依赖过程需要谨慎小心，各个 Spring 子项目间容易产生冲突。

Spring 可以通过 XML 配置文件简单的配置，或者通过注解的形式直接配置，例如业务类使用@Service 注解，控制器类使用@Controller 注解等，然后构建出复杂的应用。Spring 也提供了很多基础功能（事务管理、持久化框架集成、多线程框架集成等）开发者可以专注于应用逻辑的开发。Spring 设计的核心是 org.springframework.beans 包，这个包通常不是由用户直接使用，而是由服务器将其用作其他多数功能的底层中介。Spring 中最核心的接口是 BeanFactory 接口，它是工厂设计模式的典型实现，允许通过 bean 的名称创建和检索 bean 对象及其信息。BeanFactory 也可以管理对象之间的关系。Spring 的 IOC 容器就是围绕 BeanFactory 进行扩展的。IOC 让程序员专注于实际需要做的事情，而不是纠缠在依赖关系和生命周期的管理上。

2.2.3 MyBatis 中间件

MySQL 是世界范围内最流行的开源关系型数据库管理系统，开发者为瑞典

MySQLAB 公司，目前属于 Oracle 公司所有。支持标准的 SQL 查询。当前发布版本为 5.7，其具有高效、可靠、易用等特点，可以开源免费使用。它已成为互联网公司首选的数据库产品，例如 Google、Yahoo、YouTube 等[21]。本系统使用 MYSQL 作为数据库使用，采用 Mybatis 中间件管理数据库连接和操作。

Mybatis 是一个基于 Java 的支持普通 SQL 查询，存储过程和高级映射的持久层框架。原本是 Apache 的一个开源项目 iBatis，2010 年这个项目由 Apache 软件基金会迁移到了 google code，并且改名为 Mybatis。2013 年 11 月再次迁移到 Github 平台上。当前最新版本是 3.2.8。本系统使用 Mybatis 作为持久层框架。

Mybatis 消除了几乎所有的 JDBC 代码和参数的手工设置以及结果集的检索。Mybatis 使用简单的 XML 或注解用于配置和原始映射，将接口和 Java 的 POJOs（Plain Old Java Objects，普通的 Java 对象）映射成数据库中的记录。Mybatis 最强大的特性之一就是它的动态语句功能。Mybatis 从功能看有三层，包括 API 层、数据处理层和基础支撑层。API 层向外部提供 API，数据处理层负责具体的 SQL 操作，包括查找、解析、执行和结果映射处理等，基础支撑层负责最基础的功能，包括事务管理、连接管理、配置管理和缓存配置使用等。开发者通过这些 API 使用数据库。API 层每收到一次调用 request，然后继续调用数据处理层进行具体的数据库操作，数据操作依赖底层的基础支撑层提供的组件。

Mybatis 配置文件可以自动生成。目前，可以使用 Mybatis Generator 插件自动生成部分代码。自动生成代码需要借助 generatorConfig.xml 配置文件，Eclipse 和 IDEA 都能很好的集成该插件，该插件生成的代码都有特定的注释信息，所以在重复生成时都会检查原文件中的代码信息，不会修改用户自己编写的代码。自动生成代码之前，需要手动配置好数据库的连接信息。自动生成的代码包括：

1）client 包，定义和存放 mapper 接口文件；

2）model 包，定义和存放实体 bean 文件；

3）mapper 包，定义和存放 mapper 的 xml 文件。

2.2.4　Apache Shiro 权限框架

Apache Shiro 是功能强大并且容易集成的开源权限框架[22]，它能够完成认证、授权、加密、会话管理等功能。其功能有：

1) 认证指用户身份识别，常被称为用户“登录”；

2) 授权指访问控制；

3) 密码加密指保护或隐藏数据防止被偷窥；

4) 会话管理指每用户相关的时间敏感的状态。

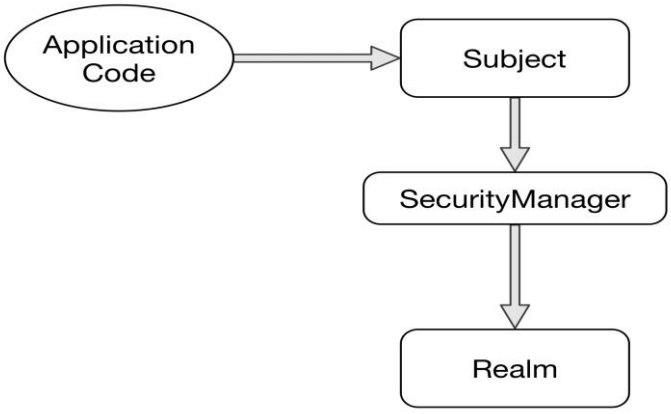


图 2.3　Shiro 框架高层概念图

如图 2.3，Shiro 框架有三个核心组件：Subject， SecurityManager 和 Realm。在 Shiro 中，Subject 指当前用户的安全操作，这一概念并不仅仅指人，也可以是第三方进程、后台帐户或其他类似事物。SecurityManager 则管理所有用户的安全操作，是 Shiro 框架的核心，采用典型的 Façade 设计模式，Shiro 通过 SecurityManager 来管理内部组件实例，并通过它来提供安全管理的各种服务。Realm 则充当了 Shiro 与应用安全数据间的“桥梁”或者“连接器”。也就是说，当对用户执行认证（登录）和授权（访问控制）验证时，Shiro 会从应用配置的 Realm 中查找用户及其权限信息。Realm 实质上是一个安全相关的 DAO，它封装了数据源的连接细节，并在需要时将相关数据提供给 Shiro。当配置 Shiro 时，你必须至少指定一个 Realm，用于认证和（或）授权。配置多个 Realm 是可以的，但是至少需要一个。 Shiro 已内置了可以连接大量安全数据源（又名目录）的 Realm，如 LDAP、关系数据库（JDBC）、类似 INI 的文本配置资源以及属性文件等。如果缺省的 Realm 不能满足需求，你还可以插入代表自定义数据源的自己的 Realm 实现。

本项目中因为涉及到面向内部的管理端和面向服务端服务端的开发，为进行用户权限的统一管理和授权，非常适合应用这一权限管理的技术方案。

2.2.5　Tiles 布局框架

Tiles 是一种 JSP 布局框架，主要目的是将重复的 jsp 格式的页面作为一个的页面的部分机能，组合成一个最终表示页面，便于对页面的各个机能的变更及维护，更容易实现代码的重用。当前最新版本是 v3。

Spring 框架很容易与 Tiles 完美集成，配置比较简单，在 Spring 配置文件添加

TilesConfigurar 类配置器和 TilesView 视图解析器。添加 Tile 模板定义 XML 文件。

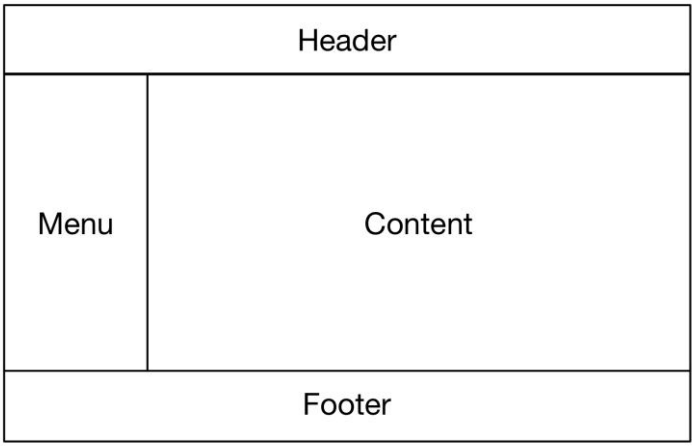


图 2.4　Tiles 块概念图

Tiles 引入了 Layout 的概念，把一个页面划分为几块。通常的来说一个页面大概可以划分为如下几块，如图 2-4 所示：

1) header 页面头部，存放一个运用的公共信息：logo 等，如果是网站可能是最上面的一块；
2) menu 页面菜单，放置一个运用中需要使用的菜单，或者在每一个页面都使用的连接；
3) footer 页面尾部，如版权信息，友情链接等；
4) content 页面主题内容，每个页面相对独立的内容。

2.3　PKI 基础设施

安全技术的合理运用是系统能否确保安全可靠的关键。第三方合同托管系统对安全要求很高，所以本系统必须使用 PKI 基础设施和电子签章技术对信息进行签名加密。美国是世界上最早使用数字签名技术的国家。从技术角度讲，数字签名主要是利用一种复杂的算法技术为当事人创建一个特殊的电子密码，并确保这个密码能证明该人的身份，也要有保障可靠安全的一系列措施，确保发件人发出的资料内容不被篡改。确定发送方的身份准确和确保电子文件在传送中不被篡改是其主要作用。因为数字签名具有抗抵赖性，同时又能防止文件发出以后接收方或发送方否认，所以通过这一技术能够有效保障电子文件的完整和真实性。

PKI（Public Key Infrastructure）公钥基础设施，是一个用非对称密码算法原理和技术实现的，具有通用性的安全基础设施[23]。PKI 利用数字证书管理公钥，通过第三

方可信任的认证机构 CA（Certificate Authority），把用户的公钥和用户的其他标识信息绑定在一起，唯一标记密钥持有者的身份。对密钥进行规范化的管理，透明地为应用系统提供身份认证服务、数据保密服务，满足各种应用系统的安全需求。为组织或机构构建和维护一个可信的安全环境和各种必要的安全保障，其功能归纳起来就是为应用提供如下安全支持：

1）PKI 应实现 CA 以及证书库，CRL 等基本的证书管理功能；

2）密钥备份及恢复；

3）交叉认证；

4）加密密钥和签名密钥的分隔；

5）密钥历史的管理。

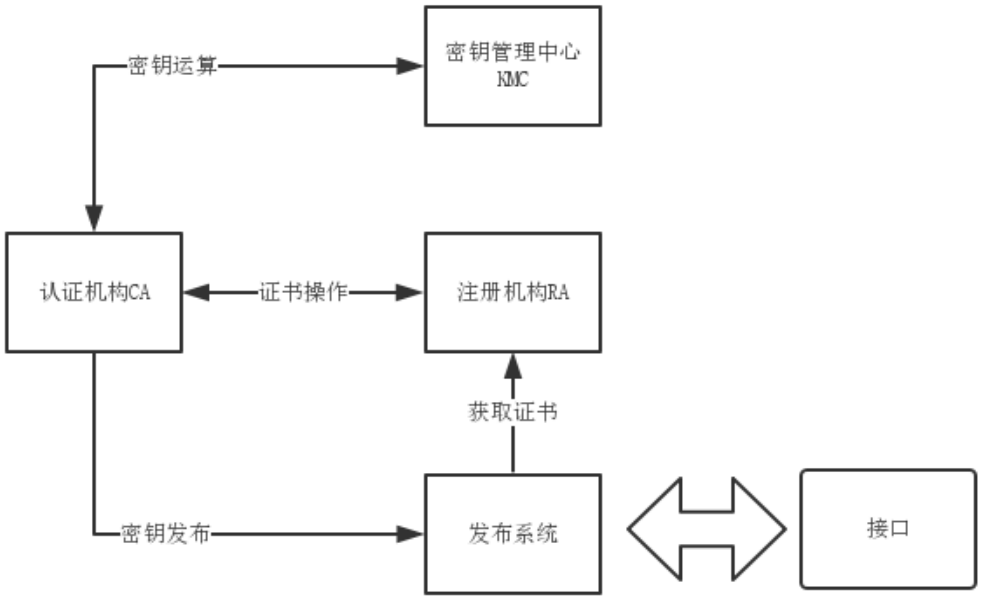


图 2.5　PKI 体系示意图

如图 2.5，PKI 公钥基础设施体系主要由密钥管理中心，CA 认证机构、RA 注册审核机构、证书/CRL 发布系统和应用接口系统五部分组成。其中，密钥管理中心(KMC)向 CA 服务提供相关密钥服务，如密钥生成、密钥存储、密钥备份、密钥恢复、密钥托管和密钥运算等。PKI 公钥基础设施的重心是 CA 认证机构，它主要负责生成和签发证书，生成、签署和发布证书撤销列表（CRL）到目录服务器，维护证书存储的数据库和日志审计等功能。RA 注册审核机构是 CA 认证机构的延伸，负责数字证书的申请、审核，注册和发证，同时对所发放证书进行相应的管理。发放的数字证书可以存放在多种介质中，例如硬盘、IC 卡或 USB Key 中。发布系统主要提供用户提供在线注册、证书和 CRL 的目录浏览服务和证书状态在线查询服务。接口系统一般采用

API，JavaBean，COM 等多种形式，主要向外界提供 PKI 安全服务使用的入口。

数字签名作为一种电子身份的认证的手段，被普遍用于网上银行，安全网络通信等领域。使用过程有两步：第一步是使用私有密钥进行加密（称为签名过程），第二步是接受方或验证方用公开密钥进行解密（称为验证过程）。因此，从某种意义上讲，使用电子文件和数字签章，甚至比使用经过签字盖章的书面文件安全得多[24]。数字签名有两种功效：一是能确定消息确实是由发送方签名并发出来的；二是数字签名能确定消息的完整性，因为数字签名的特点是它代表了文件的特征，文件如果发生改变，数字摘要的值也将发生变化，这一过程不可逆。不同的文件理论上不可能得到相同的数字摘要。一次数字签名涉及到一个哈希函数、发送者的公钥、发送者的私钥。

2.4　关键技术

电子合同在存储介质中保存的以 PDF 文件形式和非关系型数据库协同保存为佳，如图 2.6 所示，PDF 存储文件有利于传递交换，下载打印，方便阅读，方便对文件加密，服务器将其视为静态资源，缺点是依赖于操作系统文件系统，一般操作系统中的文件系统对于文件数量要求都有限制，以 linux 的 ext3 文件系统为例，每个目录最多能放置 32000 个文件。非关系型数据库存储合同文本二进制数据中有利于文件原始信息的提取、处理。适合大规模和高并发读写请求，效率更高，缺点是不方便用户使用，增加服务器处理压力。本系统的实现的核心功能是自动将 HTML 代码转变为 PDF 文件，要求无人处理，支持批量处理，个性化处理。

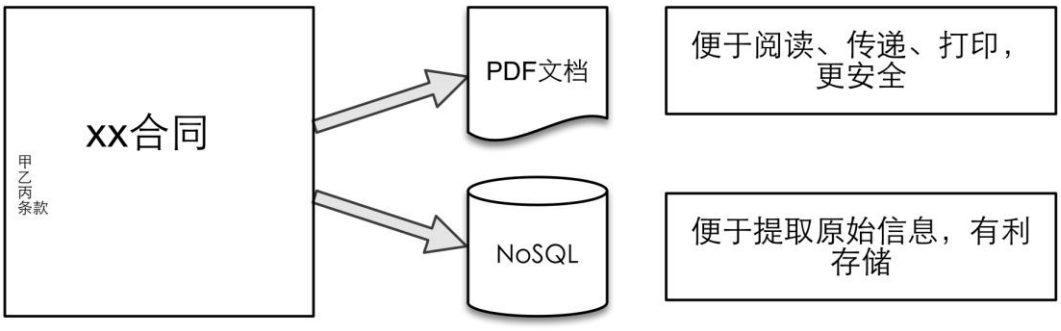


图 2.6　合同存储示意图

目前业界生成 PDF 文档，主要流行的 Java 语言的解决方案有两种。一种是 PDFBox，Apache 推出的一个开源工具库，它是一个用 java 语言实现的类库，采用对象的方式描述 PDF 文档。PDFBox 在其 org.pdfbox.cos 包中定义这些基本对象类型。在 org.pdfbox.pdfmodel 包种定义了一些熟悉的访问 PDF 文档对象的高层 API，它的基础

是 COS 模型。另一种重要的工具是 iText，其是开源项目，遵守 AGPL 协议，目前商业项目中使用该类库，需要购买商业许可。

将 HTML 代码转换为 PDF 主要有三种方案 iText、PD4ML、flying saucer 三种，其中 PD4L 速度快，纠错能力强，支持多种中文字体，但是是一个闭源商业软件。flying-saucer[25]是一个纯 Java 开发的库，遵守 LGPL 协议的开源项目。该项目是将 XML 或者将应用了 CSS2.1 样式脚本的 XHTML 文件渲染为 PDF 文件、图像或通过 Swing 展现在桌面上，其纠错能力差，支持多种中文字体（部分样式不能识别）。iText 速度快，纠错能力差，支持中文（要求 HTML 使用 Unicode 编码），但只支持一种中文字体。

鉴于三者应对业务实际效果都不是很理想，所以本项目在学习 flying saucer 基础上，改进开发了一款适合合同托管业务需求的 HTML 渲染引擎，能够处理常用 HTML 标签和 CSS 样式，对不严谨的 HTML 代码有很好地容错能力，引入缓存技术，提高了渲染效率。

2.4.1 PDF 技术

电子合同主要呈现在计算机上，为了满足电子合同的订立、格式、阅读、储存、传输、打印以及防伪等需求，电子合同必须采用一定的文件格式。PDF 文档能够满足这些要求。

PDF（全名 Portable Document Format），是一种可移植文档格式，能在不同语言的操作系统、不同的硬件平台上使用，迁移频率相对较小，是互联网上发行电子文档和传播数字化信息的理想文档格式，能够可降低管理程序和管理费用，减少文件长期保存中数据丢失的风险，方便共享文件资源。其内容也更容易从一种媒体向另一种媒体转换，或从一种数字平台向另一个数字平台转换，这无疑可以减低文件长期保存中因迁移而可能带来数据丢失的风险[26]。目前，PDF 格式文件事实上已成为数字化信息的一个行业标准，支持特长文件，集成度和安全可靠性都较高。

PDF 将文字、超文本链接、字型、格式、颜色以及独立于设备和分辨率的图形图像等封装在一个文件中。PDF 文件使用了工业标准的压缩算法，所以一般比较小巧，易于传输与储存。一个 PDF 文件从物理角度可分为文件头（Header），文件体(body)，交叉参考(cross-reference)，文件尾(trailer)，增量更新（incremental updates）等组成部分。一个 PDF 文件能够标注使用的格式版本和结构定位信息，包含一个或多个“页面（Page）”，PDF 处理引擎可以单独处理各页面，大大提高渲染处理效率，特别适合多线程系统。每个页面包含若干对象，这些对象可能是文本，图形，或图像，并且这些对象由内容流来描述，内容流包含了一系列的组件用来描述页面或者其它图实体（graphics entity）。PDF 文档包含一个图形状态的数据结构，它存储当前控制图形的参

数信息。在图形渲染中需要用到下列参数：

1) CTM 数组，指变换矩阵，它将位置从当前空间映射到设备空间；
2) 裁剪路径，指当前的裁剪路径，定义了输出设备的显示边界。其初始值是输出页面可见边界；
3) 色彩空间，指当前颜色值所在的空间；
4) 颜色，指当前颜色是在绘制操作中应用的；
5) 文本状态，指多个参数组成，用来绘制文本的；
6) 线宽，指轮廓线的宽度；
7) 线冠，指线条终点的形状；
8) 线角，指两条线结合处的形状；
9) milter limit，指线条转角处的最大长度；
10) dash pattern，指虚线条黑白线间的方式；
11) Flatness，指曲线渲染到输出设备的精度；
12) Rending intent ，指 CIE-Based 色彩空间向设备色彩空间转换时使用的。
13) 软遮罩，指不透明度；
14) alpha 常量，有 stroking 和 non-stroking 两种类型；
15) 光滑性，色彩在输出设备表示时，色彩斜率的精度；
16) 渲染模式，指色彩融合效果。

2.4.2 iText 组件

iText 是程序开发中用于生成 PDF 文档的一个 java 类库。使用纯 Java 语言编写，通过 JVM 与底层设备联系。第一个版本发布时间是 2000 年至 2008 年间，基于 MPL/LGPL 协议，允许商业软件通过类库引用（library link）方式使用 iText 类库而不需要开源商业软件的代码。iText 公司成立于 2008 年，为了筹措项目资金，支持项目的可持续发展，2009 年 iText 项目更改为遵守 AGPL 协议的开源项目，项目包名也由 com.lowagie 替换为 com.itextpdf。 Java 开发人员可以利用 iText 提供的 API 和文档说明创建 PDF 文档，非常适合动态文本，内容可被自由定制，能够实时批处理，无须人工参与，它有如下强大功能:

1) 支持 FDF 文档布局、设置页面大小、页边距；
2) 支持文档加密；
3) 支持创建表格，对列表排序或者打乱顺序；
4) 支持添加页眉、页脚和页号、支持多种字体和颜色；
5) 支持多种格式的图像、制造阴影和水纹等效果、生成文档模板[24]；
6) 支持中文，需要下载 iTextAsian.jar 包。其核心包都不支持中文问题。中文不

能正常显示的问题。

最后必须说明，商业项目使用 iText，必须向 iText 公司购买商业许可（Commercial License）。iText 本身提供了一个简单的 HTML 的解析器，它可以把 HTML 转化成需要的 PDF 的 document。但其无法识别很多 HTML 的标记（tag）和属性（attribute）。无法识别 CSS，本项目中 HTML 渲染引擎在绘制环节需要借助 iText 的功能，最终生成 PDF 文档。

2.4.3 HTML 语言

HTML 是指超文本标记语言（Hyper Text Markup Language），不仅包含文字，也可以添加图像、链接和音乐等。结构上由头（head）和主体（body）两部分组成。其中头部描述网页的相关信息，主体部分是网页的真正内容。HTML 标记符的作用是告诉浏览器按照怎样的格式显示文件内容。在渲染过程中，浏览器顺序遍历网页内容，然后依据标记符解析和显示其文本内容，一般来说，浏览器都支持对 HTML 代码容错，不会因为错误信息停止解释和执行过程，编码人员可以通过显示效果找到出错部分，分析出错原因，不同浏览器解释对同一标记解释可能不同。当前最新标准是 HTML5，HTML4.01 是最常见版本。

HTML 文档是由 HTML 元素定义的。HTML 元素指的是从开始标签到结束标签的全部代码，例如： this is a example 。大多数 HTML 元素是嵌套组合的。HTML 文本具有很好的扩展性，能够通过层叠样式表，来重新布局文本内容的显示。样式表分为外部样式表，内部样式表和内联样式，当浏览器读到样式表，它就会按照这个样式表来对文档进行格式化。

选择 HTML 作为合同输入文本，使用 HTML 渲染引擎进行渲染，主要因为一般网络平台呈现电子合同都是以文本形式呈现，HTML 可以通过样式表方便的控制文档布局。开发人员只要一次写入合同，可以处处使用，不用在在后台代码中使用 PDF 渲染组件提供的 API，笨拙的手动布局和调试，修改合同样式也非常方便。合同数据传送到托管平台，托管平台可以根据样式定义，还原合同内容和格式。

合同文本的常用的 HTML 标记有 div、h1 到 h5、br、img、table、tr、th、span 和 a 等。所有的 HTML 元素，都要么是 block(块元素)、要么是 inline(内联元素)，并且块元素和内联元素的基本差异是块元素一般都从新一行开始。

2.5 本章小结

本章主要介绍了合同托管的业务，从电子合同基本概念和特征出发，简述了电子交易合同和电子服务合同的区别，明确了电子合同托管业务中电子合同托管的基本范

畴，从电子合同订立规范出发，引出了当前电子合同订立和存储中存在的问题和解决之道。紧接着，详细介绍了系统开发过程中使用到的重要框架技术、安全技术和关键技术，框架技术从项目的技术选型 JEE 技术到 Spring 框架技术，在该框架技术基础上介绍了 Mybatis 中间件，Shiro 权限框架，Tiles 布局框架。安全技术主要强调了 PKI 基础设施，安全设施对于项目安全运营非常重要。关键技术介绍了 PDF 文件技术、iText 组件和 HTML 语言基础，阐明了为什么使用 HTML 渲染引擎生成电子合同。

第三章 在线合同托管需求分析

就目前实际应用案例来看，电子合同一般都是借款用户直接和网贷平台签订，用户下载合同文件并保存在存储介质中，平台通过数据库和文件系统保存历史合同数据。这一过程不符合《合同规范》要求。某些合同的订立过程较为复杂，需要用户和平台进行多次互动，现有的网贷平台难以满足需求。当前的一些托管平台在形式上追求与线下合同订立同样的效果，采用图片形式显示合同文件内容，需要用户手动书写信息和签名，体验较差。因此必须对合同托管平台的需求分析做全面考量，重新设计一个可靠的、体验良好的业务流程。

3.1　业务形成

合同托管的需求分析必须有理有据，通过分析合同托管业务规范，其信息交互一般都是通过 Web 页面展现，Web 页面则是通过 HTML 代码渲染生成，内容可以由文本组成，也可以由图片组成。为了实现展现、传递和存储的一致，方便动态数据和文本直接融合，降低托管平台和合同托管平台的开发工作量，需要寻求更为有效的解决方案。

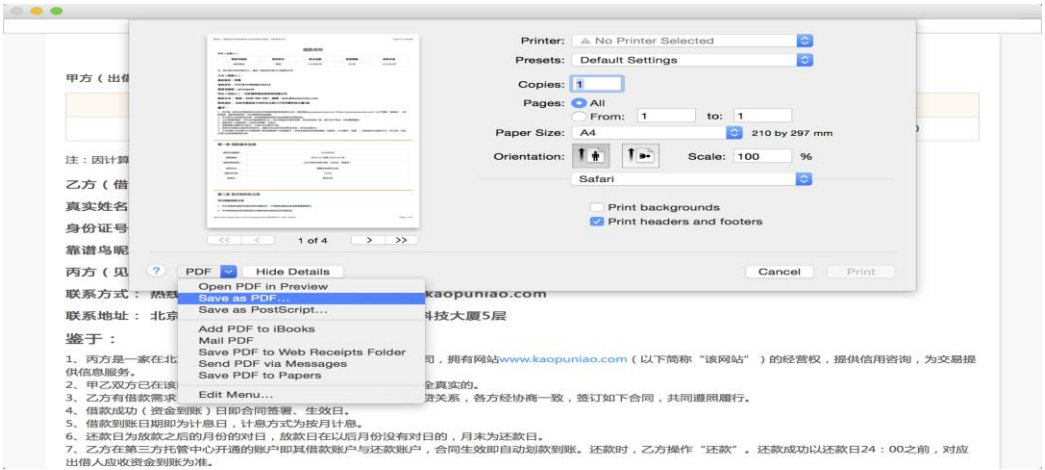


图 3.1　客户端渲染示例图

最直接有效的解决思路就是将 Web 页面的 HTML 代码直接渲染成 PDF 文档。现在有这样的业务场景，网贷平台拟定一份已经补全了借款信息的电子合同，以 HTML 代码经由浏览器渲染之后展现给用户，当用户阅读并点击“同意按钮”后，就正式和平台订立了电子合同。当前，HTML 代码格式转换为 PDF 文档格式的主流方案有两种：

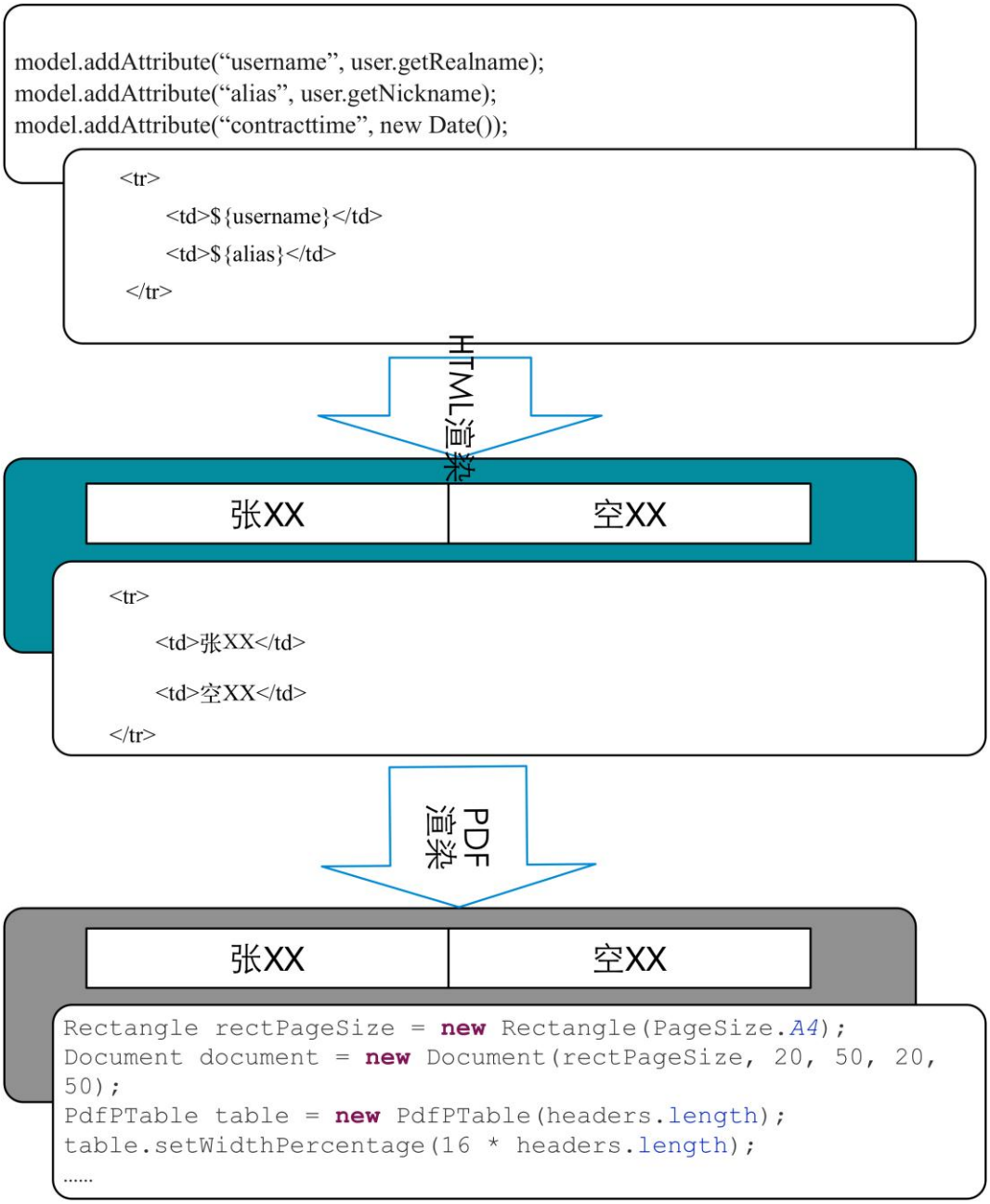


图 3.2　服务端渲染示例图

1) 客户端渲染，用户点击“合同下载”按钮，调用 Web 浏览器 PDF 插件提供的打印功能，用户选择打印到 PDF 文档，点击打印后，保存 PDF 文档到用户计算机上，用户可以将该文档保存在自己的计算机上或者备份到其它存储设备上；如图 3.1 所示；
2) 服务端渲染，开发人员将合同按照一定格式、图文内容要求硬编码为后台业务代码，用户点击“生成按钮”时，向网贷平台发送生成 PDF 文件请求，后台

验证请求合法后，控制器调用相应生成合同的业务代码，在服务器上渲染 PDF 文件，向客户端动态返回相应的 PDF 文档，现在浏览器一般都集成了 PDF 浏览插件，用户通过浏览器浏览或保存 PDF 文档，如图 3-2 所示。

这两种方案各有优势和局限，如表 3.1 所示：

1) 客户端渲染展现效果和打印效果能够保持一致（如果显示页面添加过多广告、图标、链接等元素，将影响输出效果），合同文本一般以 HTML（XML）语言编写，工作量小，在合同文本变更时，不需要重新编译源码，但用户和网贷平台生成合同的操作是分离的，网贷平台一般不存储合同的 PDF 文件，这意味着公司内部管理运营人员调阅合同文本数据比较麻烦，需要后台管理系统调用数据库中的合同数据，或者借助管理人员手动操作生成；
2) 服务端渲染合同文本是固化在后台程序源代码里面的，通过调用第三方 PDF 渲染组件提供的强大输出功能实现。管理端和客户端能够同时生成 PDF 文件形式的合同文本，方便管理，但代码变更时，开发人员不得不针对合同内容和格式手动编写或修改后台代码，然后重新编译部署，非常麻烦。

表 3.1　渲染方案比较表

方案	优点	缺点
客户端渲染	效果一致、工作量小、不需要编译源码	操作复杂，管理麻烦
服务端渲染	客户端和服务端同时生成，方便管理	工作量大，灵活性不足，每次更新版本需要编译源码

托管平台的存在，不仅能够改善网贷平台业务流程中存在的缺陷，也能克服以上两个方案的缺点，发挥他们的优点。

根据网贷业务需求，网贷主体角色划分为网贷平台客户（用户），网贷服务平台（商户），担保公司（担保方）和合同托管平台四类。其中网贷平台客户可再分为借款客户与出借客户。如下图 3.3 展示了一个具体的业务流程案例，为了简化和更明确的说明，主题角色中不涉及担保方：

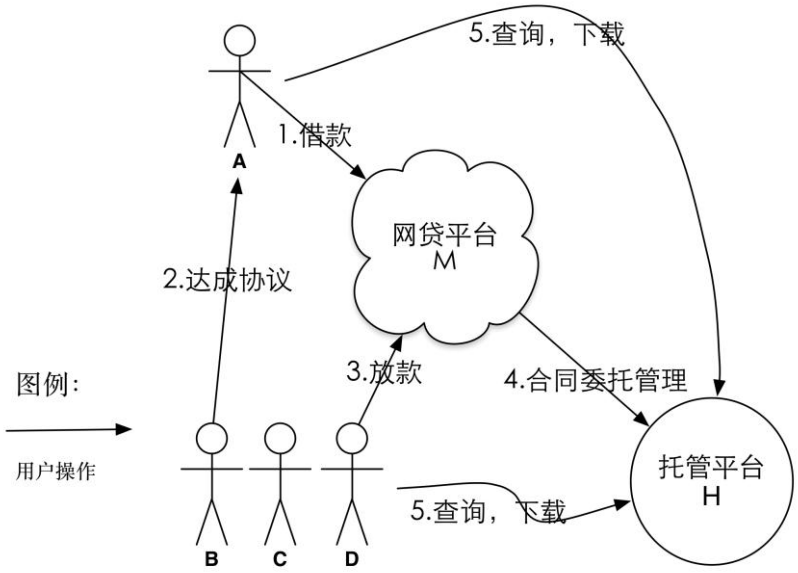


图 3.3　角色关系和业务流程图

借款用户 A 在网贷平台商户 M 筹措一定金额借款，B、C、D 三位用户愿意借钱给用户 A，他们达成借款交易（协议），授权商户 M 拟定与用户 A 的借款合同，并将资金交予商户 M 暂存或托管的资金划转到 A 的名下，商户 M 在满足规则条件后，用户 A 决定接受放款操作，这时为了保护各方利益需要拟定一份三方借款合同。商户 M 根据借款情况，拟定一份电子合同，邀请用户 A 登录商户平台确认合同。用户 A 确认合同后，商户 M 将借款转到用户的账户。这份合同则由用户 A、B、C、D、商户 M 分别持有。

一般情况下，用户在确认合同后，借款会按约定到账，用户 A、B、C、D 四人，为保护自身合法利益都应该登录商户平台，查看和下载这份电子合同的 PDF 文档或者将其保存 HTML 文件或图片文件到自己的计算机上，并妥善保存和备份。进一步引入合同托管平台 H，商户 M 首先申请注册使用合同托管平台 H 提供的合同托管服务。在审核通过之后，合同托管平台发给商户 M 唯一的商户 ID、接口 API 说明文档、SDK 包、插件 jar 包以及一对公私钥。商户 M 的研发人员按照文档说明开发相应的合同托管调用模块。用户 A、B、C、D 在使用商户 B 的服务时，必须在在托管平台 H 上开通托管账户，开通账户非常简单，不应增加用户痛苦度，已经开通过账户的用户不需要重复开通，直接跳转登录。用户 A 和商户 B 订立合同时，商户 B 将已经拟定的合同文本数据以 HTML（XML）形式发送到托管平台 H，托管平台 H 验证请求信息合法后，在其服务端渲染生成 PDF 文件的合同文档，并备份请求数据。该合同文档授权给用户 A、B、C、D 和商户 M 共同所有，用户登录后可以查看和下载具有访问权限的电子合同，所有合同都以 PDF 文件和二进制数据形式存储在 H 的服务器上。

在完成合同签订之后，托管平台会将签名信息附加时间信息生成二维码图片，附加在电子合同文本的末端，一同渲染成 PDF 文件。托管平台 H 在托管成功后将二维码图片返回给用户，用户可以将该二维码图片分享在其它社交网站上，以备作为“时间戳”认证服务之用（部分替代联合信任时间戳服务中心提供的“时间戳”服务），可以借助更多可信的第三方社交平台进行辅助证明。

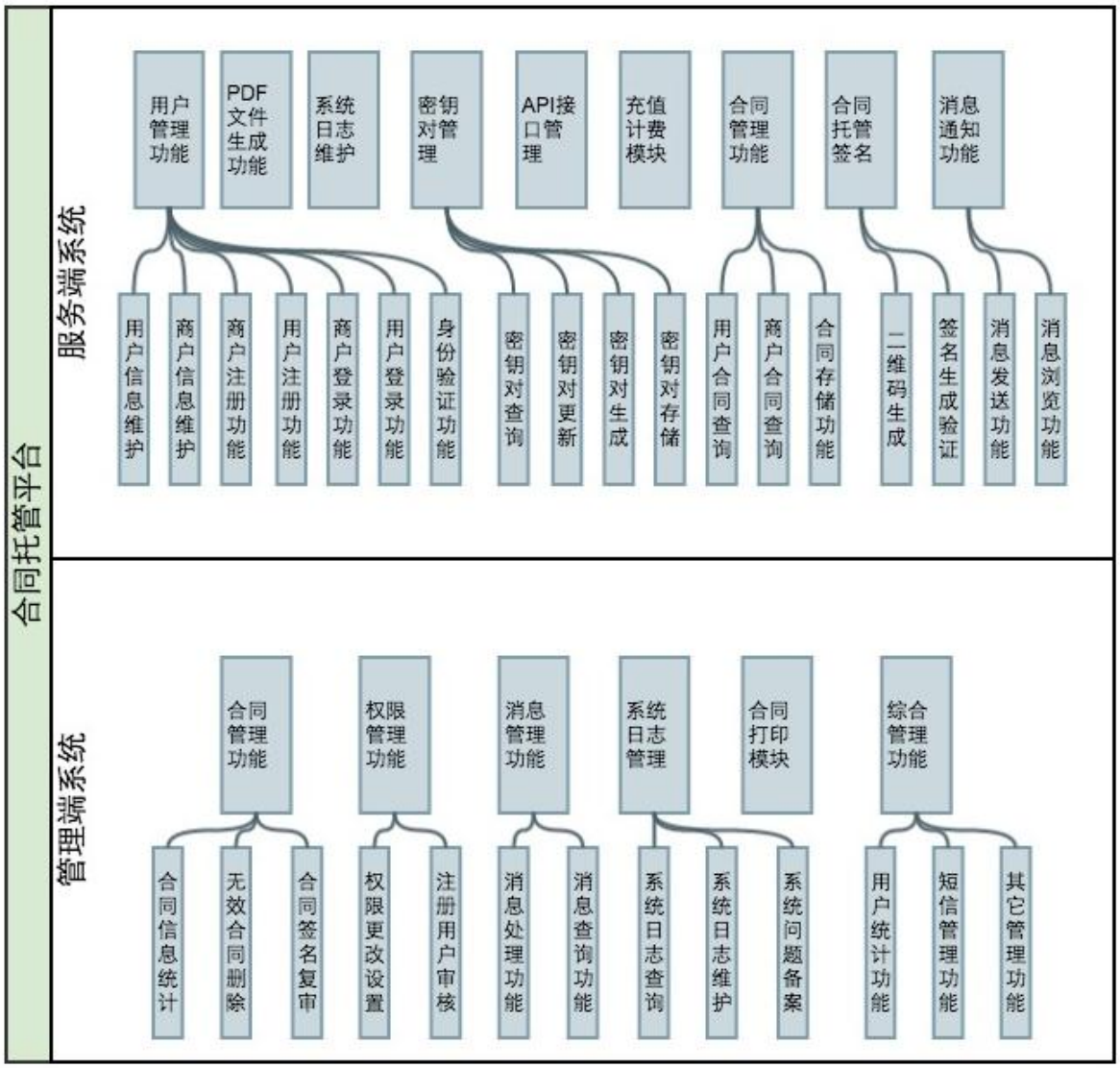


图 3.4　系统总体功能图

3.2 总体需求分析

本系统核心目标是提供一套用户体验良好、保证系统安全的电子合同订立存管解决方案。保证合同稳定性的关键是实现订立和存管的分离。本系统将会对接众多网贷平台，不同平台合同内容和形式也会千差万别，因此系统必须适应不同合同托管和渲染的请求。系统主要为商户提供在线合同生成、存管和查询功能，降低商户研发合同模块的难度，优化了商户合同订立的流程，为普通用户提供在线合同的存管、查询和下载功能，实现一个帐号管理所有在线合同的功能。

合同在订立过程中必须由用户亲自“确认”合同，不能代为执行，该过程应置于电子合同托管平台上进行。规范强调：经过谈判后形成的电子合同文本，应征求电子合同订立方的确认，确认后进入电子签名阶段。电子合同签名时，必须附加各方的电子签名信息，才有和纸质合同相同的法律效力。电子签名人应该妥善地保存和管理自己的电子签名数据。签名当事人在知道电子签名制作数据已失密或者可能失密的情况下，应当及时告知有关各方，并终止签名，更换其它未失密的电子签名制作数据。用户的电子签名应由托管平台代为保存，只要平台不发生泄露或遗失签名数据的情况，就能够保证遵守规范要求。此外，用户亲自保存电子签名，往往需要保存到 Flash 盘、USBKey 或硬盘上，不便于移动使用，体验相对不好。经过签名的电子合同文本应当保存在电子合同订约系统或第三方存储服务商服务器上，以便随时查询下载。电子合同的完整储存信息应当包含合同内容、签约时间、合同订立各方主体信息、电子签名信息等。

如图 3.4 系统总体功能图，展现了电子合同托管系统的总体全貌，合同托管平台总体分为服务端和管理端两部分。服务端主要面向用户和商户提供服务，向用户展现 Web 页面，为商户提供 Web 服务和接口服务。管理端为企业内部运营和管理人员提供业务受理和监控服务。服务端的主要功能有用户管理功能、合同管理功能、合同托管签名、PDF 文件生成功能、密钥对管理、API 接口管理、充值计费模块、消息通知功能和系统日志维护功能等。管理端主要有合同管理功能、权限管理功能、消息管理功能、综合管理功能、系统日志管理和合同打印模块等。

电子合同托管平台最基础的功能是用户管理功能，用户管理功能在系统服务端表现为普通用户和商户，因为普通用户和商户需要提供的注册和申请信息不同。用户管理功能主要包括用户注册、登录和信息维护。

用户密码通过 Hash 加 Salt 算法为用户登录密码和签名密码进行加密，盐值是随机生成的，与用户信息一起存储在数据库表中，极大增加黑客利用查表法、逆向查表法或者彩虹表等方法进行破解的难度，保证用户账户的安全性。用户注册完成之后，系统调用密钥对生成模块为注册用户分配一对公私钥，系统调用密钥对存储功能，通

过 DES 加密算法，将公私钥按照一定顺序组合后加密，安全存储在数据库中，用户可以输入密码，借助 DES 算法解密，查询或更新自己的密钥，从系统级别保证密钥对安全，通过对数据库的安全优化防止用户信息的泄露。

用户在与商户平台签订合同需要 PDF 文件生成模块、API 接口管理模块、合同托管签名等多个模块协调完成，其中用户的签名需要借助公私钥完成。

用户完成合同生成后，用户可以通过合同管理功能，管理自己已订立的合同，如果合同已经过期失效，用户可以将合同删除，如果合同订立有问题，用户可以举报申诉。用户可以随时查询和下载自己的合同。

商户注册完成后，商户平台需要将其系统对接到合同托管平台，所以需要合同托管平台提供 API 接口功能以及调用文档说明，方便商户调用，和快速部署系统。商户根据 API 接口提供的功能和用法构建自己平台的合同模块。通过 API 调用将商户系统和托管平台连接起来。商户登陆托管平台服务端系统之后，可以按照时间和合同订单号来查询发生在自己业务平台上的电子合同数据和统计信息。

合同托管平台必须对商户和用户进行身份认证，认证时触发身份认证功能。商户的调用信息必须进行身份验证，保证系统安全。合同托管平台的安全建立在 CA 安全认证体系之上。通过基于 CA 证书的安全体系机制保证托管平台的安全性。合同托管平台必须申请证书，商户可以申请证书，支持无证书商户的托管请求。

托管平台需要托管业务管理平台。支持后台审核商户用户，为商户生成密钥对的功能，分配权限的功能。管理端需要为自己的管理人员分配业务权限，也需要通过权限管理功能管理商户和普通用户的权限。统计平台用户量、保存合同数量等信息，批量检查合同签名信息，以及对问题合同进行删除的功能。

不同商户提供的电子合同在格式和内容上是不同的，合同可能有图片、表格等复杂元素，这要求将 HTML 代码渲染必须能够适应这些要求。因此需要开发满足业务需要的 HTML 代码托管引擎。该引擎必须保证托管合同的安全性，完整性，一致性和准确性。

在托管业务发生之后，需要将成功信息及时返回给用户，因此托管平台必须提供消息通知和管理功能。管理端也需要对已发出去的信息进行管理。

为了保证系统安全运行，同时保证在系统发生故障时能够及时定位问题原因，方便技术和管理人员,托管平台必须设计一个覆盖管理端和服务端的系统日志维护功能。对系统运行信息进行及时记录，方便查询和对问题备案处理。

托管系统必须保证电子合同的隐私性，系统必须确保合同内容不能被内部管理员访问到，因此管理端系统开发过程中不能提供访问合同内容的接口和功能。

整个系统运营过程中提供的价值是有成本的，因此在设计合理的盈利模式之后，需要考虑收益，所以需要引入充值计费模块，为付费用户提供增值服务。

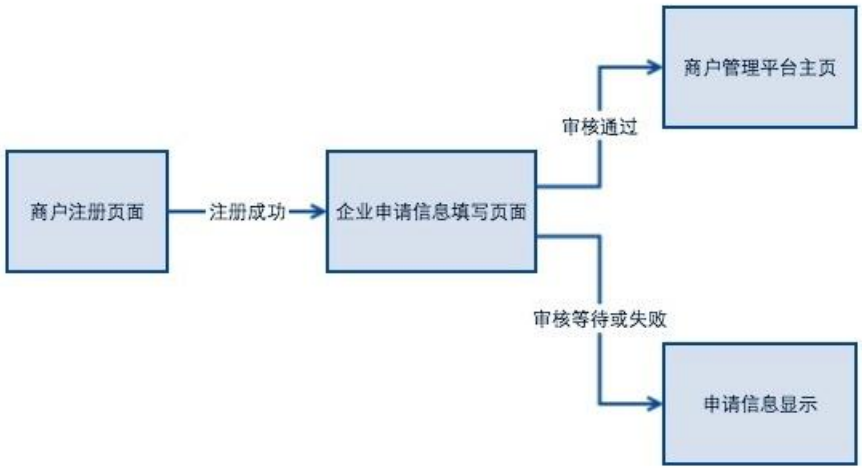


图 3.5　商户注册流程图

3.3　功能性需求分析

合同管理平台业务比较明确，但功能较为复杂，本节描述了主要功能性需求。合同托管平台服务的对象是普通互联网用户和寻求合同托管业务的商户。从基本的注册登录功能来看，因为不同群体其性质和需求不同，所以托管系统在设计时，必须将两个群体的差异体现出来。本系统设计时将普通用户注册登录和商户注册登录分别设计。

如图 3.5 所示，描述了商户的注册流程。商户使用合同托管系统首先需要开户注册，开户注册信息需要系统管理员审核通过，由商户负责人提供开户名称及手机号码（商户负责人的号码），输入手机验证码，设置登录密码完成开户注册。用户注册成功后跳转到企业申请信息填写页面。

商户在企业申请信息填写页面，按照要求填写企业申请信息，并阅读托管合同业务申请说明协议，同意协议条款后提交申请。这一页面的设计原则是意图减轻商户申请的痛苦度，只要求填写必要、便于审核的信息 。

企业申请信息包括：

1)　企业名称，企业在工商部门登记备案的名称；

2)　注册地址（国家，省（直辖市），市（区、县），街道），企业公司住所；

3)　企业法人，填写企业注册法定代表人姓名；

4)　经营范围，填写企业注册的经营范围；

5)　注册类型，公司性质；

6)　联系方式，方便与企业沟通。

用户填写完成信息并提交申请后，该笔申请单就被提交给管理端系统，管理端管

理员根据用户的申请信息，赋予商户使用权限，为商户生成一对唯一的商户数字签名证书（Digital Certificate）。在这个过程中如果尚在审核中或者审核未通过的商户登录系统看到的只有申请信息展示页面，并显示给用户商户的等待结果或者失败原因，方便商户重新申请。如果商户通过审核，那么商户登录后直接跳转到商户管理平台的主页。

相比商户，普通用户的注册要求简单和快速，只要填写邮箱、手机号（辅助认证）和登录密码，系统会向用户邮箱发送安全链接，用户点击链接轻松完成注册、也可以接入三方账户平台一键注册功能。用户注册后，使用注册邮箱和密码可登录用户管理系统主页。

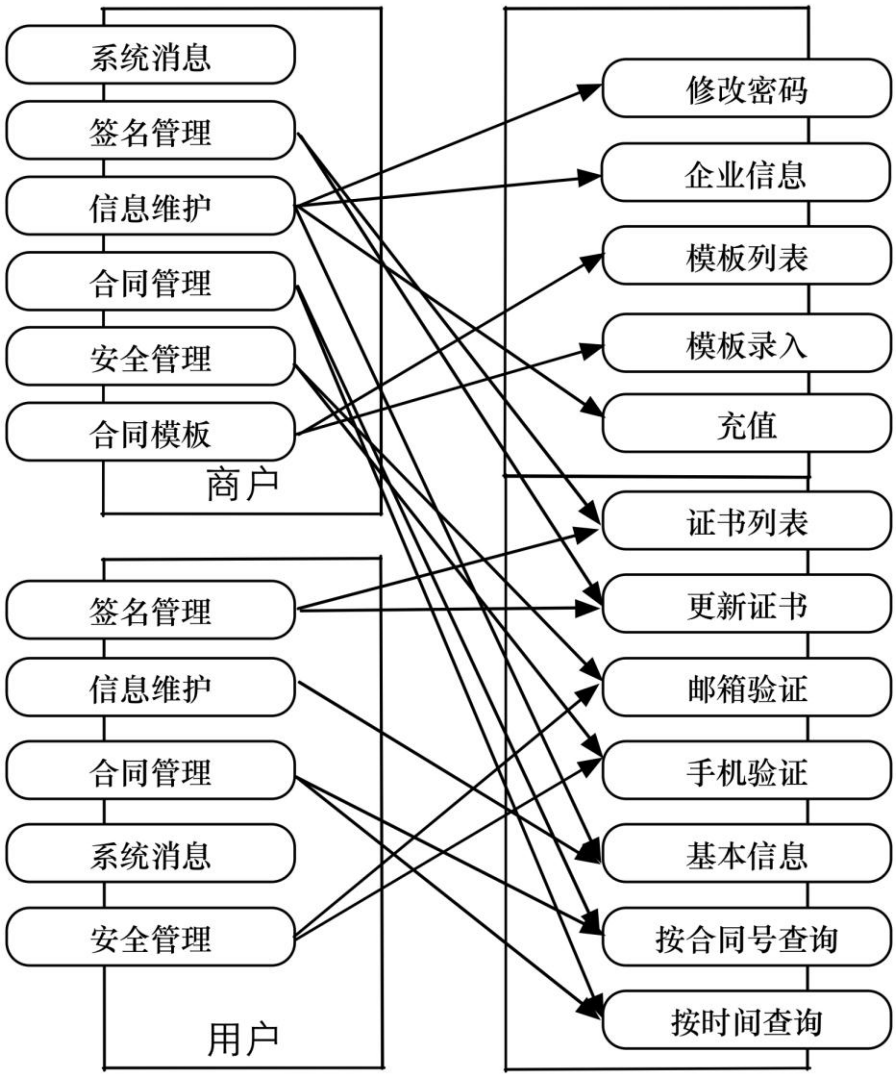


图 3.6　商户和用户界面菜单列表

用户管理系统主页和商户管理系统主页内容基本一致，都提供了安全信息查询维护、签名证书查询维护、合同查询、消息浏览等功能。用户和商户可以在安全管理页

面修改自己的登录密码、邮箱和手机号等信息，修改邮箱需要邮件验证和手机号需要6位校验码验证。如图3.6展示了商户和普通用户登录系统后分别看到的菜单列表，系统为商户提供统计信息功能，以利于商户做出商业决策，商户也可以通过充值选项购买增值服务。

系统的核心功能是合同托管功能。合同托管功能需要几个功能模块协同完成的。如图 3.7 展示了合同管理的用例图。合同请求托管用例包含合同存储用例和合同生成用例。合同存储用例有多种形式的扩展用例，本系统存在数据库存储和 PDF 文件存储两种形式的扩展，如果考虑非二进制形式，还包含纸质存储用例。尽管系统安全保存了需要的电子合同,但是纸质合同的保存是必要的,以防系统不可恢复性的文件丢失。因此管理端系统必须支持合同打印功能。管理端系统每日跑批，打印当日发生的电子合同，进行纸质备份。

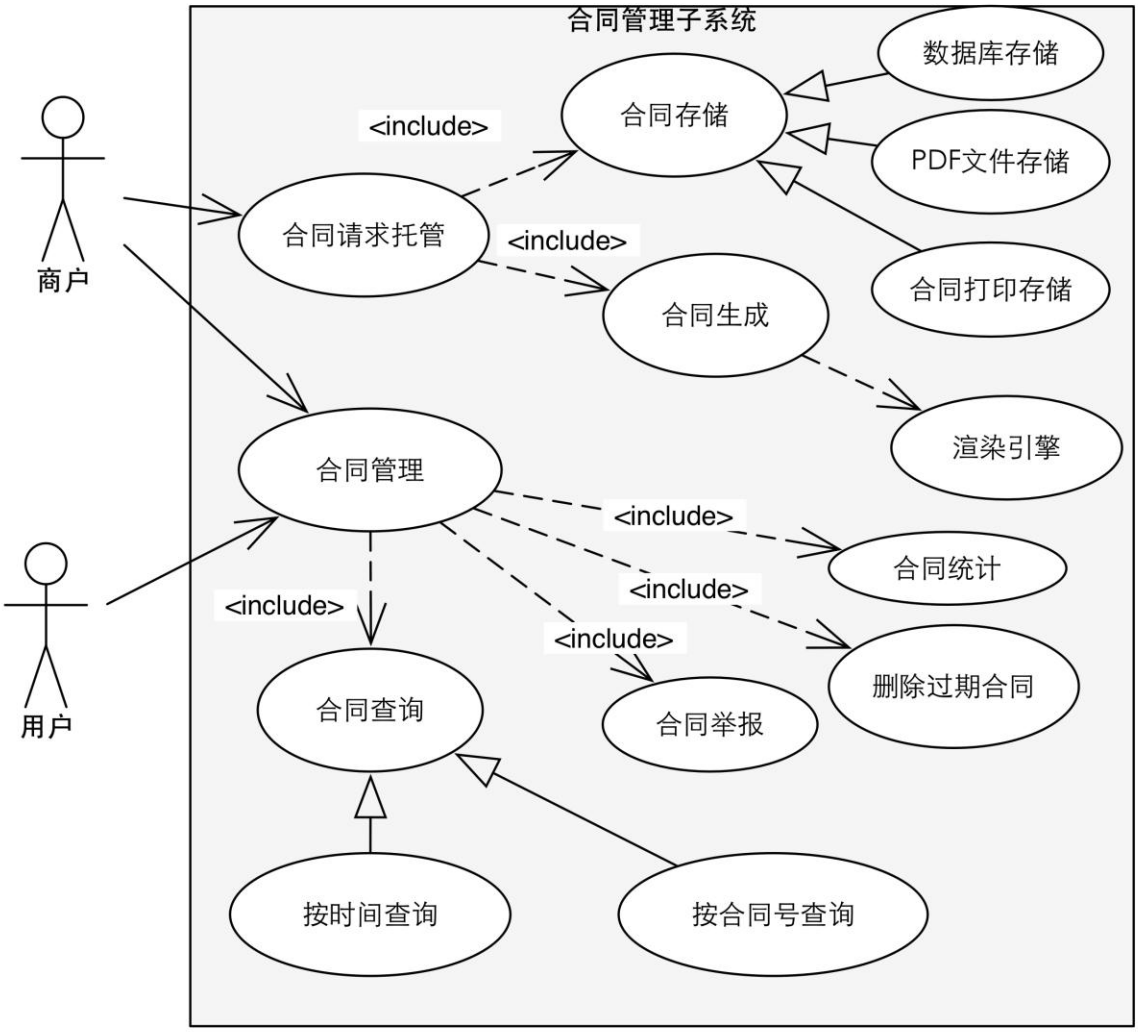


图 3.7　合同管理用例图

合同管理用例包含合同统计、合同查询、合同举报和删除过期合同。合同查询有

两种形式的扩展按时间查询和按合同号查询。

合同生成引擎的设计与开发是系统的重要核心。该引擎主要负责将商户发送过来的 HTML 文本数据进行正确的解析处理，需要将 CSS 代码所定义的样式正确地与 HTML 元素正确地匹配。然后调用 PDF 文件生成包，生成相应的 PDF 文档。在解析 HTML 文档时需要解决如下几个问题：

1) HTML 文档不严谨的问题，虽然浏览器能够正确地显示文档内容，但是商户如果发送过来的合同文本，有标签没有闭合的情况系统就发生错误，并抛出异常；
2) CSS 渲染问题，例如 CSS 中块（Block）元素和内联（Inline）元素和可变元素如何按照样式属性设置，正确的渲染在 PDF 文档中。例如，一般合同文档中不会有 form 标签元素，但不保证商户发来的合同数据中没有 form 元素，form 这个块元素比较特殊，它只能用来容纳其他块元素，渲染器必须解决这类元素的渲染问题；
3) HTML 在浏览器和 PDF 文档中显示格式差异比较大的问题。如何正确和美观的显示是一个需要解决的问题。

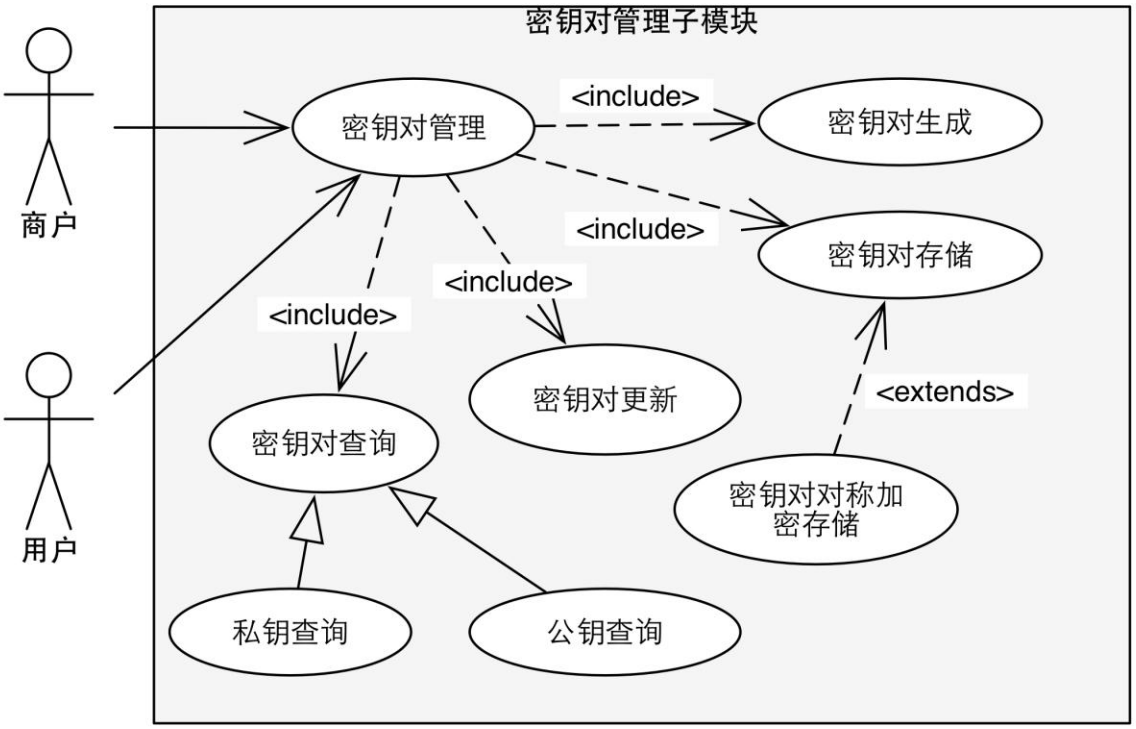


图 3.8　密钥对管理用例图

密钥对管理是系统安全的重要组成部分，如图 3.8 密钥对管理用例图，密钥对管

理用例包含密钥对生成、密钥对存储、密钥对查询、密钥对更新几个用例。密钥对生成接口调用 JDK 提供的 keytool（Java 密钥和证书管理工具）提供的算法，使用 genkey 命令生成私钥（secret key），linux/Unix 环境下代码片段如下：

```
StringBuffer cmd = new StringBuffer();
cmd.append("cd $JAVA_HOME/bin");
cmd.append("keytool -genkey -v -alias " + alias + " -keyalg " + algorithm + " -keysize "
+ keyLength + " -validity " + days);
cmd.append("-keystore $JAVA_HOME/jre/lib/security/cacerts");
cmd.append("-keypass " + password + " -storepass " + pwd);
cmd.append("-dname \"CN=" + name + ",OU=cn,O=cn,L=cn,ST=cn,C=cn\"");
try {
    Process ps = Runtime.getRuntime().exec(cmd.toString());
} catch (IOException e) {
    logger.error(e.getMessage());
}
```

然后使用 RSAUtils 工具类导出证书（certificates）和公钥（public key）。密钥对存储分别以.keystore 文件和关系型数据库表存储。密钥对存储用例包含密钥对加密用例，系统必须对从证书信息中提取出来的公私钥对，存储在数据库中时，利用 DES 算法做更进一步的加密。密钥对查询用例可扩展为公钥查询用例和私钥查询用例，用户利用私钥签名或者加密信息，发给其它人，其它人需要使用其公钥解密，如果其它人使用用户的公约签名或者加密信息，用户需要使用自己的私钥解密信息。密钥对更新用例，主要是为用户添加新的公私钥，并对历史公私钥进行迁库备份，使用旧密钥对签名加密的信息，仍然使用对应的旧密钥对进行解密。

为了方便商户调用系统，系统还必须设计和实现稳定、强大的接口管理功能。接口管理功能应该进行接口抽象，利用模板模式、工厂模式或适配器模式等设计模式进行接口封装，对外提供 API 接口，方便未来动态扩充接口，而不需要修改主体代码。

3.4 安全性需求分析

安全保密是合同托管生存的根本，合同托管业务最核心的数据是合同文本数据，其与用户签名数据皆属于机密信息，用户和商户注册信息也属于敏感信息，必须确保这些信息绝对安全，系统设计必须保证合同信息只有合同拥有者才能够接触。

电子合同和纸质合同不同，电子合同伪造容易，危害极大，所以需要采用电子合同数字签名技术来防止合同伪造，因此需要设计一个可靠的签名验证系统和签名流程。可靠的签名验证系统是电子合同安全认证的核心。电子签名与书写签名以及盖章认证

具有相同的法律效力。作为一个可靠的电子签名应满足如下条件，电子签名的制作数据必须为电子签名所有人专有；电子签名人实际控制签署电子签名制作数据[27]。通过数字证书及其数字签名实现交易各方的身份验证；

在统一的接口管理系统之外添加身份认证功能能够保证系统对外的安全性，在调用托管接口的环节，攻击者盗用商户和用户账号、密码等信息，以合法身份伪造合同数据，干扰系统正常运行，信息在传递过程中被篡改和泄露。这要求每次交互过程，参与合同托管的各方，均需要向彼此亮明身份。参与合同托管的各方均不能否认其所做的操作（不可逆性）。

应用安全保障概括而言包括合同托管各方身份真实性、信息的秘密性、信息的完整性和托管各方不可抵赖性四方面的需求。具体而言包括通过加密技术实现平台端敏感数据保护，比如账号、密码、密钥对和合同文本等数据。

安全的文件存储系统主要目标是安全的存储商户和用户的合同文档数据。合同数据以文件的形式存储在服务器上，安全的存储系统，允许授权用户访问存储服务器上的 PDF 文件，采用非关系型数据库存储合同二进制数据，这就涉及非关系型数据库的安全优化问题。

系统部署过程中同样必须考虑安全需要，内网要与外网隔离，内网必须设置防火墙（Firewall），需要对企业级路由器进行正确的设置，系统使用到的云服务器，必须购置安全服务，例如云盾保护。

3.5　非功能性需求分析

本系统的非功能性需求主要是确保用户在商户平台和合同托管平台身份信息一致，商户的密钥对要定期更新。本系统主要运行在商业公有云和公司内部搭建的服务器上，面向各种互联网金融平台后者其它平台提供合同托管服务，应具有多用户、多线程并发处理的能力，系统应具备在正常网络情况下中迅速响应的能力。具体的非功能性要求如下：

1) 服务器宕机率控制在千分之一下，现阶段可以采用比较成熟的商业云服务解决方案；
2) 合同文件存储器至少有两套互为备份；
3) 服务器维护和数据备份等工作，安排在系统每日凌晨 3 点钟进行；
4) 合同托管响应请求操作不超过 30 秒；
5) 各个服务器能够长时间稳定运行；
6) 支持多用户并发，支持多平台同时访问；
7) 消息服务畅通，保证队列里的消息至少被消费一次；

8) 引擎渲染速度要快，支持更多并发要求，充分利用多服务器多处理器特性；
9) 系统必须记录完整的日志信息，所有请求需要有日志可查，便于快速定位 bug 位置。

3.6 本章小结

本章具体分析了电子合同托管业务的形成过程，介绍了两种渲染方案的优缺点，以及由托管平台进行渲染的优势。从业务形成出发归纳和总结出合同托管业务总体需求，并进行了详细的阐述，然后借助系统界面菜单列表和核心功能的用例图，详细分析了系统的功能性需求，简述了系统的安全性需求和提出了系统的非功能性需求，为后续系统设计和实现打下了重要基础。

第四章 系统设计与实现

系统总体采用基于 Web 的 B/S 架构设计部署，利用 Maven 管理项目 jar 包依赖，利用 Nexus 搭建多人协同开发需要的 Maven 私服，使用分布式项目代码托管工具 git 管理项目代码，使用 Idea for Mac 作为开发环境，系统整体遵循了 MVC 设计模式，表示层、逻辑层和数据层业务分明，根据项目特点，采用 Maven 多模块（Multi Module）工程来划分功能模块，搭建项目骨架，分派开发任务，利用消息队列（MQ）实现各个子项目的松耦合，即插即用，使用 quartz 进行跑批处理，使用 Mybatis 作为 ORM 映射，采用 RUP 迭代开发策略，使用 JIRA 管理项目任务进展，利用 Tower 驱动项目知识和文档。

4.1 系统设计原则

合同托管系统设计过程中应遵循如下的原则：

1) 技术先进性：采用先进且稳定的的技术架构，本系统采用了 Spring 这一成熟高效的轻量级非侵入开发框架，吸收互联网开发的先进经验，采用稳定的消息队列技术（MQS），作为消息中间件，帮助系统功能间解耦合，适合功能迭代开发，同时满足大数据量的业务需求；
2) 快速迭代特性：采用 Scrum 敏捷开发策略；
3) 可扩展性和可伸缩性：使用模块化设计原则，用户根据需要选择不同组件，建构不同规模的应用系统，新业务和新功能的增加不影响系统正常运行；
4) 开放性：系统支持与其它系统的互联互通，接口设计必须设计良好；
5) 系统安全性：确保用户信息和合同数据一致准确，不被非法入侵者破坏和盗用，每一次交互都必须签名，确保不能被抵赖；
6) 易操作、易管理：用户界面交互必须简单友好，兼顾美观；
7) 合规性和统一性：系统的软件、硬件在充分满足业务需求的前提下，均应符合相关的业界标准，不同厂商开发的系统应保持统一的业务功能和衡量标准。必须确保遵守相关法律法规。

4.2 系统的总体设计

系统的总体设计主要针对系统总体架构和部署进行了设计和实现。

4.2.1 总体架构

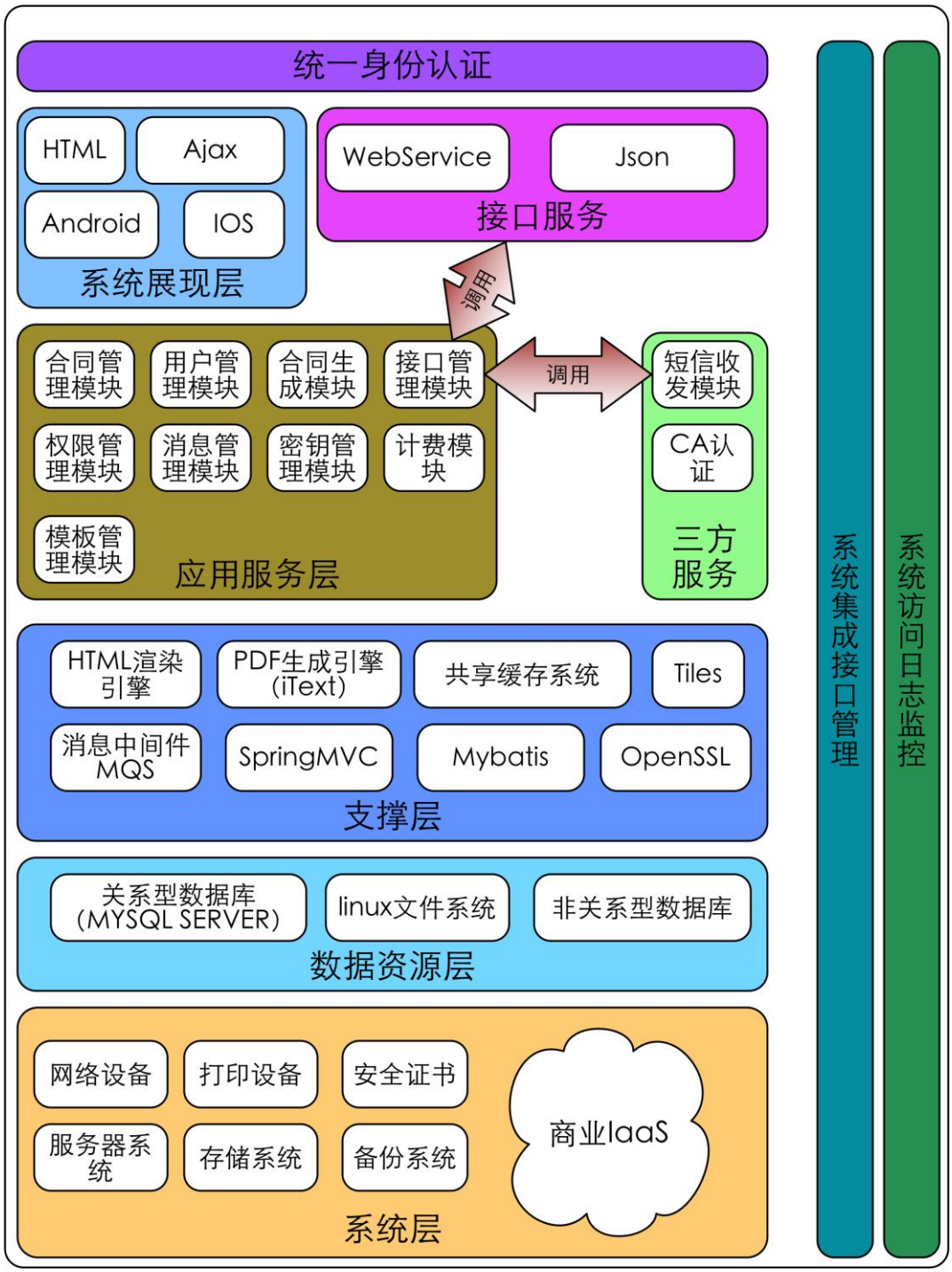


图 4.1　系统架构图

如图 4.1 系统架构图所示，合同托管系统采用基于消息中间件的分布式架构，具有松耦合、易扩展、分布式和包容异构系统的特性。消息中间件一个重要的作用是保

证商户调用一定会被响应。消息中间件支持 String 类型的数据 put 和 get 操作。考虑到不同模块需求的数据不同，所以为了交互方便，系统必须定义数据交互格式，目前主流的数据交互格式主要有 XML 扩展标记语言和 JSON。为了提高数据交换解析效率，系统规定所有信息以 JSON 格式发送和读取。JSON 是一种更轻量级的数据交互格式，具有更高的数据交互效率[28]。

系统层是整个架构最底层，提供基础服务，主要由服务器系统、存储系统、备份系统、网络设备、打印设备和安全证书以及商业 IaaS 构成。

数据资源层，定义了存储、访问、维护和修改数据的设施，同时管理和满足中间层的数据请求。关系型数据库主要存储用户注册，合同存储地址等信息。Linux 文件系统主要用于存储 PDF 格式的文件。非关系型数据库适合存储合同的文本信息。

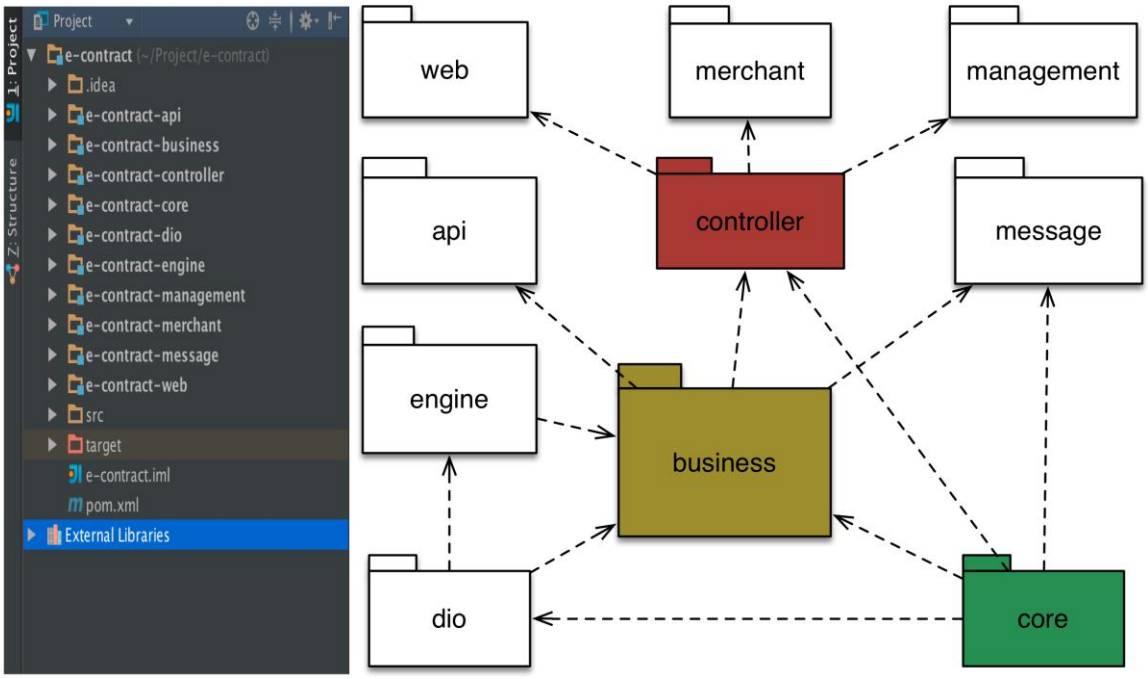


图 4.2　项目开发包关系图

支撑层包括 HTML 渲染引擎、PDF 生成引擎、Mybatis 和工作流引擎、消息中间件，共享缓存系统，Tiles 布局框架等核心组件。其中 Mybatis 提供 ORM 映射，能与 Spring 很好的结合，提供了强大的动态 SQL 功能。共享缓存可以共享多个服务器用户登录的 session 信息和其它需要不同 server 间共享的数据信息。

应用服务层规范了各个业务模块，负责业务逻辑。托管平台由合同管理模块、合同生成模块、用户管理模块、接口管理模块、权限管理模块，消息管理模块、密钥管理模块和计费模块组成。其中接口管理模块负责管理调用第三方服务的接口和提供对外服务的接口。调用第三方提供的短信服务和安全证书服务可以更好地完成业务需要。

接口服务有两种模式一种是传统的 WebService，另一种是更轻量更高效的 JSON 字符串信息和 Http 结合的接口服务，系统支持这两种数据形式的调用，以对接采用相应技术架构的网贷平台。

系统展现层主要提供 web 页面和移动端服务，普通用户、企业用户和管理员可以通过浏览器和 App 访问系统。

如图 4.2 项目开发包关系图，逻辑上对系统进行了模块化的分解，使得项目开发更加条理有序。项目主要由 web 包、merchant 包、management 包、controller 包、api 包、message 包、business 包、engine 包、dio 包和 core 包组成。

web 包主要负责用户服务端视图代码和展现。merchant 包主要负责商户服务端视图代码和展现。

management 包主要负责系统管理端系统视图代码和展现。

controller 包负责存放各种控制器、过滤器、监听器和拦截器等，负责处理由在 web.xml 中配置的 DispatcherServlet 分配的 url 映射。

api 包专门负责存放和管理系统对外提供的 WebService 或者 JSON 接口服务，其包含所需的 servlet 和 filter，并运行在内嵌的 Jetty 容器之上。

message 包主要放置一些监听消息队列消息的监听器（listener），主要负责发送和接受消息队列中的消息，驱动业务发展。

business 包是应用的业务包，此包非常重要，而且根据业务特性划分为众多子包，包括 device 包、bi 包、schedule 包、service 包、util 包等，后期随着业务发展，需求更新，此包需要进一步分解。其中 device 包主要负责维护发送邮件的模块、发送短消息的模块、发送系统消息的模块、调用打印设备模块、接受短消息的模块、生成二维码模块等，所有抽象输入输出设备都放在此包下。其中 bi 包提供业务接口，方便 api 包、controller 包和 message 包调用。业务功能非常多的情况下，为了适应业务的变化，使用调度器任务包是必要的，所有调度器和批量处理任务类都放在 schedule 包下，util 包仅放置 business 包需要使用到的工具类等。

engine 包是 HTML 渲染引擎包，所有与渲染有关的类都放置在此，其输入是渲染命令，由 service 包调用，service 包主要放置所有业务应用，并按照业务特性分类构建一定数量的 service 类。

dio 包，主要负责存放数据库操作（dao）、输入和输出设备等，engine 包调用输出设备，生成 PDF 文档，service 包调用 dio 包存取数据。

core 包存放各个项目都会用到的基础元数据、参数、通用的工具类、数据模型类、核心算法类、自定义异常类等。

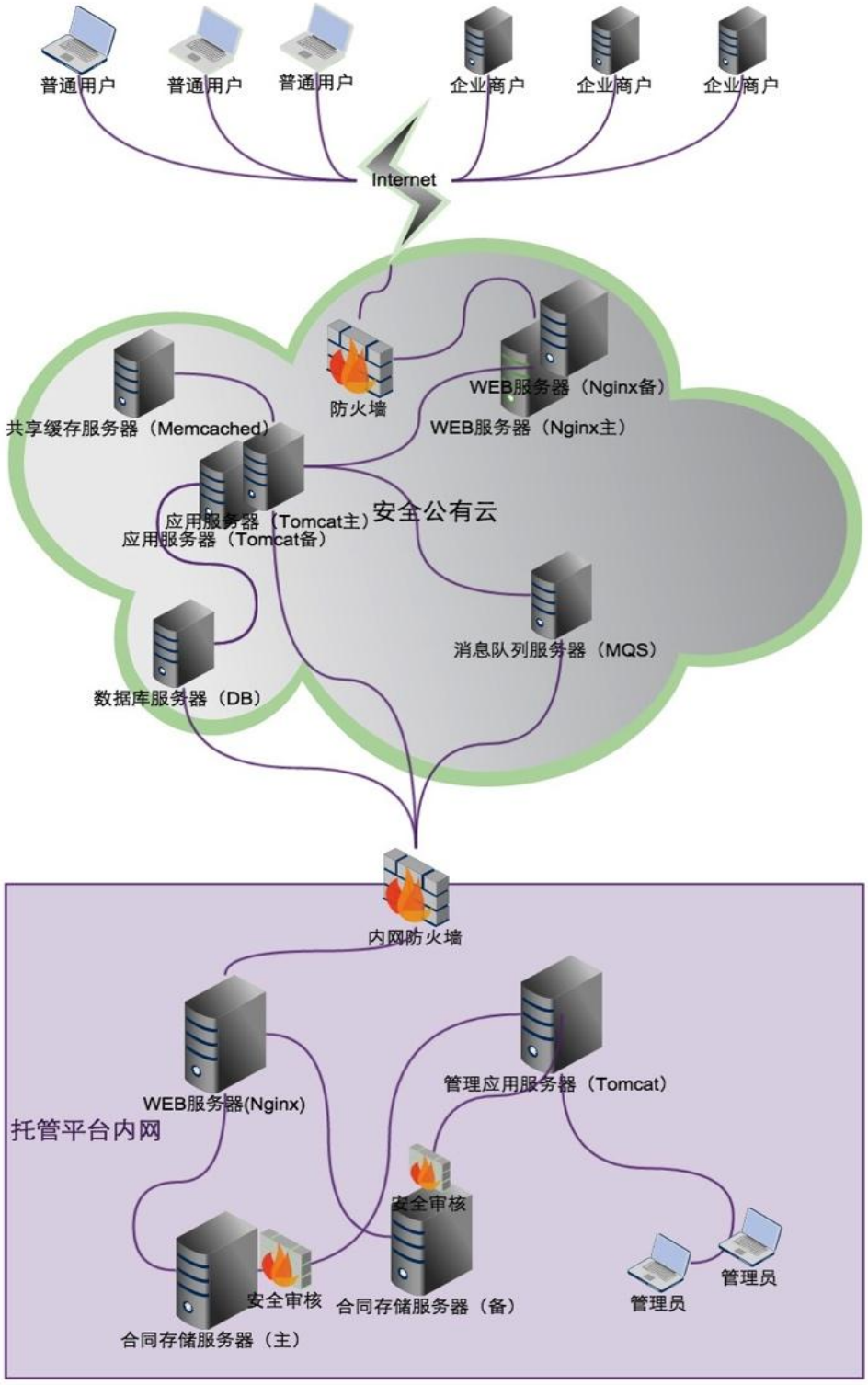

普通用户
普通用户
普通用户
企业商户
企业商户
企业商户
Internet
防火墙
WEB服务器（Nginx备）
WEB服务器（Nginx主）
共享缓存服务器（Memcached）
应用服务器（Tomcat主）
应用服务器（Tomcat备）
安全公有云
消息队列服务器（MQS）
数据库服务器（DB）
内网防火墙
WEB服务器(Nginx)
管理应用服务器（Tomcat）
托管平台内网
安全审核
安全审核
合同存储服务器（备）
管理员
管理员
合同存储服务器（主）

图 4.3　系统详细部署图

4.2.2 部署设计与实现

系统部署设计必须遵循安全和可靠第一的原则。所以在设计中应该将合同原数据保存在本地服务器上，将面向用户的信息部署在公有云端。数据存储采用主备模式，保证主服务器宕机情况下，及时切换到备机提供服务，保证数据更加安全。系统部署也必须兼顾提高系统服务效率和降低系统服务成本的原则进行部署设计，如图 4.3 系统详细部署图所示的实施方案进行。

系统服务端安全级别较低的部分部署在商业化公有云上，主要因为服务端主要面向普通用户和企业商户提供安全级别较低的服务，寻求这类服务的用户往往要求更快速、直接的响应。使用商业化公有云可以减少了搭建服务器集群的工作量，其网络质量往往更高，可以降低了部署和维护成本，提高 IT 系统的效率和安全性，并且系统具有更高可伸缩性、可扩展性，可以根据业务需求进行快速调整。安全公有云一般都能够为部署的应用提供 WEB 攻击防火墙、能进行 DDos 防护、漏洞监测和主机拦截 SQL 注入，XSS 跨站等类型的 WEB 攻击入侵防护等。总之，公有云为用户提供基于新一代的云平台远程部署系统的业务，已经成为互联网公司和开发者的首选。

托管平台使用 Nginx 做为 WEB 服务器。Nginx 是一种强大的 WEB 和反向代理服务器，由 Igor Sysoev 创建，最早发布在 www.rambler.ru 网站上，是当前最流行的 WEB 服务器之一。托管平台利用 Nginx 提供静态文件、虚拟主机、负载均衡、缓存策略、支持 SSL 等服务。

实践中公有云虚拟主机也存在宕机的可能性。因此为了保证服务连续提供，在实际搭建中，系统利用 Nginx 主从两台虚拟主机上做互备（HA），Nginx 服务器本身非常稳定，加上主从互备，基本就能保证用户访问不间断。Nginx 的配置和启动则比较简单，这里不做赘述。HA 功能需要借助 Keepalived 来实现。Vrrp 主要目的就是解决静态路由单点故障的问题。两者最大不同在 state 和 priority 上。其中一台为 Master，其它为 Backup。当 master 宕机时，系统根据 Vrrp 协议能极快的切换到优先级较高的备机上。master 和 salver 的 keepalived.conf 配置不同之处表现在 vrrp 实例配置上，如代码：

```
#vrrp_instance VI_100
vrrp_instance VI_1{
state MASTER
# state BACKUP
interface eth1
virtual_router_id 50
priority 100
# priority 50
……
```

}

应用服务器使用 Tomcat 动态服务器和 Jetty 内嵌服务器。Tomcat 是一个免费开源的 Serlvet 容器，Jetty 是一个提供了 Http 客户端和服务端的开源 Servlet 容器，都属于轻量级服务器。Nginx 和 Tomcat 的组合再优化配置后能够胜任互联网应用的需求。业务逻辑代码部署在应用服务器上，应用服务器通过 Nginx 服务器做集群。代码更新时，保证停掉一台服务进行更新时，其它 Tomcat 服务器能够继续提供服务。通过修改 Tomcat 的配置文件，来提升 Tomcat 的性能。采用了 Nginx 来作为请求分发服务器，后端多个 Tomcat 共享 session 来协同工作。

加大 Tomcat 占用内存。修改 bin 下的 catalina.sh 文件，

配置如下：

```
JAVA_OPTS='-Xms1024m -Xmx2048m -XX:PermSize=256M -XX:MaxNewSize=512
m -XX:MaxPermSize=512m'
```

加大 Tomcat 连接数和开启缓存，需要修改 tomcat/conf/context.xml 配置文件：

```
<Connector port="9027" protocol="HTTP/1.1"
maxHttpHeaderSize="8192"maxThreads="1000"minSpareThreads="100"
maxSpareThreads="1000" minProcessors="100" maxProcessors="1000"
enableLookups="false"compression="on"compressionMinSize="2048"compressableMimeTyp
e="text/html,text/xml,text/javascript,text/css,text/plain" connectionTimeout="20000"
URIEncoding="utf-8" acceptCount="1000" redirectPort="8443"
disableUploadTimeout="true"/>
```

在系统对外接口服务开发搭建过程中，没有采用开源项目，而是利用了 PrintWriter 类封装 XML 格式数据和 JSON 格式数据服务，用户可以使用 Ajax 或者其它 Client 按照要求请求接口并处理从接口获取的数据，该项目作为独立的 Maven 子项目，置于 api 包下，将 Jetty 作为内嵌服务器使用，通过调用 Server 类和 ServletContextHandler 类实现一个轻量级的接口服务。

数据库服务器使用 Mysql Server 服务器，版本号为 5.6.23。数据库引擎主要有 MyISAM 和 InnoDB。MyISAM 是默认搜索引擎，适合大量查询操作，而合同托管服务以查询操作为主，所以将数据库引擎使用默认搜索引擎。对 mysql 响应进行优化，修改 my.cnf 配置文件，限制最大连接数:

```
set-variable = max_connections=100
```

针对比较固定的业务查询，又修改比较少的数据库字段添加数据库索引，譬如根

据合同编号查询合同存储地址，以提高数据库检索效率。

使用 MQS 这款商业消息队列（目前可以免费使用）作为消息队列服务器。这是一款基于 HTTP GET/POST 协议，并遵循 RESTful 接口设计原则的 Queue，它简单易用，是一款对平台无依赖的轻量级简单消息队列服务，能提供高效、可靠、安全、便捷、可弹性扩展的分布式消息队列服务，并支持普通队列、延迟队列、优先级队列等多种队列模式。支持多个生产者和消费者并发访问同一个消息队列，并能确保某条消息在取出之后的特定时间段内，无法被其他消费者再次获得。MQS 能够帮助应用开发者在他们应用的分布式组件上自由的传递数据，构建松耦合系统。在消息有效期内，确保消息至少能被成功消费一次。这一队列使用配置简单，但需要根据 SDK 和说明文档，按照实际需求开发监听端（listener）和客户端（client）。监听端监听时需要不断轮训，开发过程中应该权衡每秒轮训次数，这里默认设置每 1 秒轮训一次。

系统将核心数据例如合同文本数据，部署在内网两台独立文件服务器上，进行主从备份，这两台文件服务器在物理上应该是独立的，他们占有不同的端口。管理服务器使用 Tomcat 因为内网用户并发访问量不是很大，所以不需要 nginx 分发 http 请求，管理端系统代码部署在该服务器上，内网管理员可以通过该服务器访问管理端提供的服务，按照管理员权限进行相应的业务操作。

文件服务器使用 linux 服务器，暂时使用 linux 默认的文件系统。短期搭建一个专属的文件系统太过复杂，可以作为长期目标。由网贷平台传递过来的合同文本原始二进制数据通过 MongoDB 非关系型数据库存储，其是一个基于分布式文件存储的数据库开源项目，是一个文档导向型的数据库[29]，具有高性能、易部署、易使用、存储数据非常方便的特点。

4.3 数据库概念模型设计

数据库概念建模是必要的，严格遵循数据库设计范式，增加数据库概念模型设计的难度，在实际开发中。因此在实际开发过程中，使用的是通用的 BIGINT 类型的 id 作为每张表的自增主键，以提升数据库的查询效率，通过冗余字段减少连表操作，通过业务和事务控制保证数据一致，E-R 模型作为实际开发中重要参考依据。

概念建模使用 Mysql Workbench 中提供的 E-R 建模工具对数据库进行建模，部分核心实体关系模型设计如图 4.4 所示。系统中的用户表有三类：普通用户表(user)、商户表(merchant)和管理员表(admin)。为了便于管理，单独建表。所有用户共有用户名、密码和角色 id 等属性。不同用户具有不同的信息属性。

在权限管理方面，定义角色表（role），包含角色名。定义权限表（permission），包含权限字符串和权限描述，定义角色权限表（role_has_permission），定义某个角色

所具有的权限。

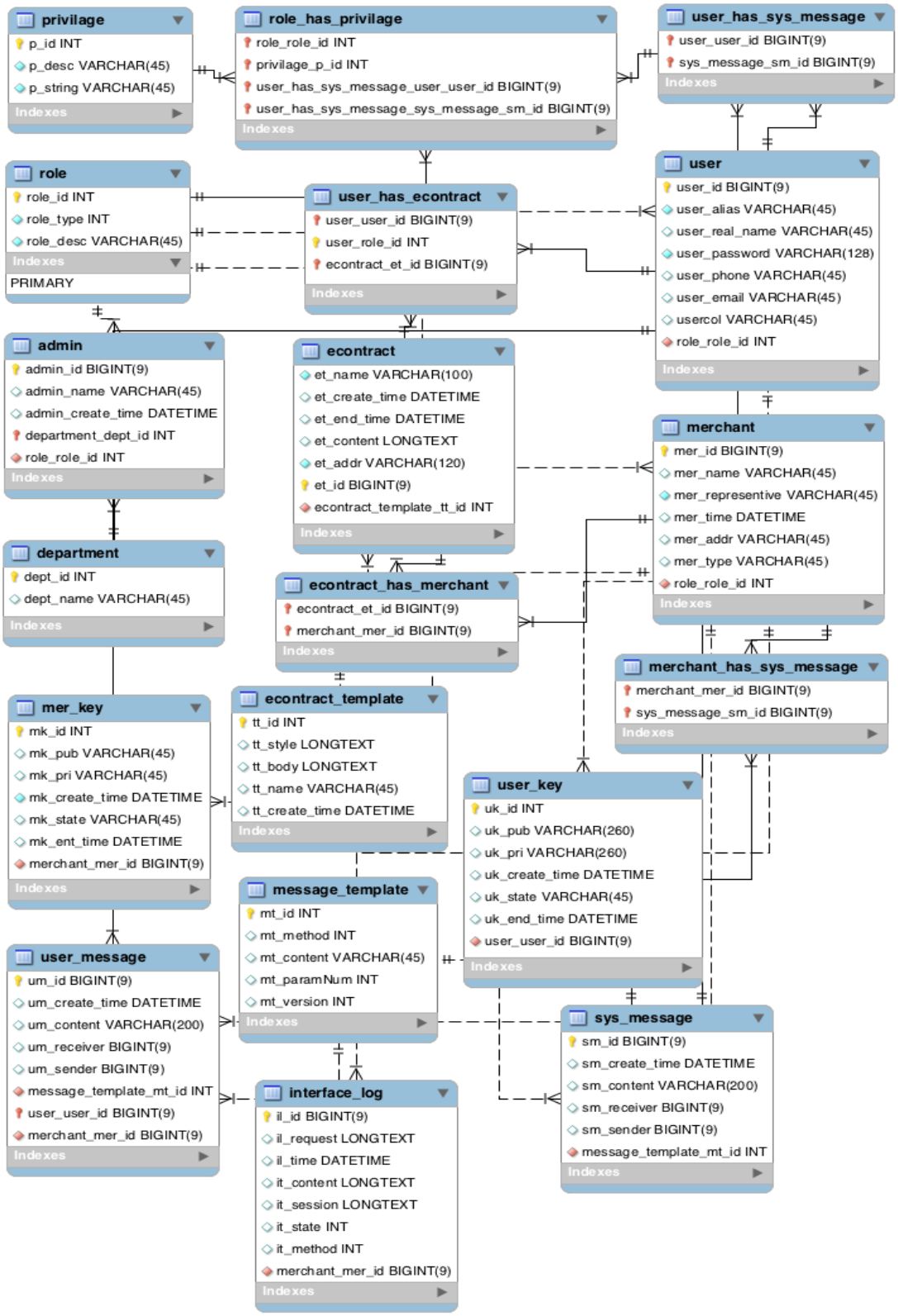


图 4.4　数据库部分 E-R 图

在合同管理方面，定义了合同表（contract），合同模板表(contract_template)，用户合同表(user_has_econtract)，用户合同表记录了合同信息及所属用户。一个合同仅有唯一对应的商户，所以商户和合同是一对多的关系，用户和合同则是多对多的关系。

在消息管理方面，模块需要用到一张消息模板表（message_tempale），系统消息表(sys_message)和用户消息表（user_message），用户消息具有私有特性。系统消息表和商户是多对多的关系，所以追加一张商户系统消息表（merchant_has_sys_message）。

在密钥对管理方面，需要普通用户密钥对表（user_key）和商户用户密钥对表(merchant_key)，商户和用户的密钥分开存储，在经过 DES 算法加密之后存储在专门数据库中，即便数据库被攻击，拿到密钥对信息，也无法还原原始密钥。

系统日志管理需要接口日志表（interface_log），渲染日志表（renderer_log），访问日志表（visit_log）等。记录所有接口请求和响应的结果，以备处理突发情况。渲染日志表记录渲染引擎使用的信息，作为开发人员进一步优化引擎的重要依据。访问日志表记录用户登录访问信息，用户每次登录系统就会生成一条记录。

4.4 HTML 渲染引擎设计与实现

HTML 渲染的基本工作流程，如图 4.5 所示，渲染引擎解析 HTML 文档并把标记转换成内容树中的 DOM 节点。解析 Style 元素和外部文件中的样式数据，然后用样式数据和 HTML 中的显示控制组件协同创建另一棵渲染树。渲染树按照显示顺序包含一系列带有颜色、尺寸等显示属性的矩形。构建渲染树之后，确定每个元素在文档中显示的位置，最后遍历渲染树，将每个元素绘制出来。该原理与 Webkit 基本一致。

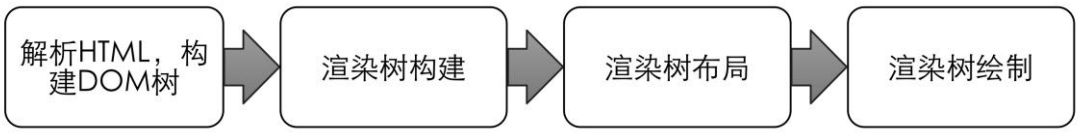


图 4.5　渲染基本流程图

首先需要抽取解析 HTML 文档信息，构造 DOM 树。但是如果 HTML 输入文本不严谨，不符合 XML 规范要求，将导致转换（transform）失败。flying saucer 中使用 XMLResource 这个类进行转换，该类使用 Java JDK 提供的 XMLparser 解析器。转换不成功时会抛出 Failed when to configure SAX to DOM transformer 异常，因此本项目中无法使用 flying saucer 原生提供的 setDocument 方法，因为该方法通过 loadDocument 方法调用 XMLResource 类。该类的功能比较弱，而且解析效率不高，所以需要对原项目进行完善。

DOM 树的构建目前可行的方案需要通过严格的 XML 文档进行构建，那么首先必须将不严谨的 XML 代码转换为严谨的代码。策略是边读取，边检查，边修复，边转换。这样就可以大大提高转换效率。幸运的是 JTidy 这个开源项目提供了转换需要的功能，JTidy 也提供了 HTML 代码检查功能。因此，本项目使用 JTidy 提供的 Tidy 类的 parseDOM 方法进行转换。具体实现是在 Renderer 中引入 Tidy 变量，而后增加设置 Tidy 的 setTidy 方法。

```
private void setTidy(){
    tidy.setInputEncoding("UTF-8"); //  暂时只支持 UTF-8
    tidy.setQuiet(true);
    tidy.setOutputEncoding("UTF-8");
    tidy.setShowWarnings(false); //不显示警告信息
    tidy.setIndentContent(true);
    tidy.setHideComments(true); //是否隐藏注释
    tidy.setBreakBeforeBR(true); //是否 br 在一行中显示
    tidy.setSmartIndent(true);
    tidy.setIndentAttributes(false);
    tidy.setEncloseBlockText(false); //是否用 p 标签包括文字
    tidy.setUpperCaseTags(true); //是否把大小的标记转换成小写
    tidy.setWraplen(1024); //多长换行
    tidy.setPrintBodyOnly(false);
    tidy.setXHTML(true);
    tidy.setErrout(new PrintWriter(System.out));
}
```

HTML 引擎的核心是对 CSS 的处理和渲染树的布局。CSS 样式可能不多，但是计算量大，是引擎的一个重要瓶颈。CSS 规则由两个主要的部分构成：选择器，以及一条或多条宣言。选择器通常是需要改变样式的 HTML 元素，每条声明由一个属性和值组成。

一个文档能够正确的按照规定样式展现，依赖于 CSS 样式脚本。CSS 能被任何结构化的文档格式使用，例如 XML 标记语言，实际上 XML 依赖的样式脚本多于 HTML。渲染器本身并不知道 XML 文档遵循的标准，渲染器本身只知道 XML 文档，它不知道一个 XML 文档可能还包括 CSS 样式信息等。因此需要 NamespaceHandler 类确定文档遵守的规范标准。元素（Element）是 HTML 文档结构的基础，在 CSS 中，每个元素对应一个 Selector，并渲染生成了一个包含了元素内容的框（box，也译为“盒子”）。但是不同的元素显示的方式会有所不同，例如 div 和 span 就不同，而 strong 和 p 也不一样。在文档类型定义（DTD）中对不同的元素规定了不同的类型，例如浏览器会根据 img 标记的 src 属性的值来读取图片信息并显示出来，而如果查看(X)HTML 代码，则看不到图片的实际内容，又例如根据<input>标签的 type 属性来决定是显示输入框，还

是单选按钮等。

W3C 规定 DOM2 的样式规则包括两个主要部分，选择器（selector）和宣言（declaration）。h1 标记就是一个选择器，color:green 就是一个宣言。所以一个 CSSStyleSheet 对象由 CSSStyleRule 对象和 CSSSytleDeclaration 对象组成。CSSStyleSheet 对象定义的是所有 CSS 样式表，包括外部样式表和使用“<style type="text/css"></style>”标签指定的嵌入式样式表。CSSStyleSheet 同样构建于其他的 DOM2 CSS 对象基础之上，而 CSSStyleRule 对象表示的则样式表中的每条规则。

每个 CSSStyleSheet 对象包含一组 CSSStyleRule 对象，这些对象包含着这样一条规则：

```
body{
font:verdana;
background:#c7600f;
color:#1a3800;
}
```

CSSStyleDeclaration 对象表示一个元素的 style 属性，与 CSSStyleRule 对象类似，CSSStyleDeclaration 具有下面属性：cssText 和 parentRule。

本引擎使用 StyleSheetFactory 这个类处理样式脚本。该类实现了工厂模式，调用 parse 方法处理样式脚本，处理 CSS 样式脚本首先必须构建 CSSParser 处理器。为了计算样式找到匹配元素的 property-value 值，一种高效的方式是构建 Hash 模型。引擎使用 RuleSet 来组织数据，RuleSet 包含了两个表，属性声明表和选择器表，代码如下:

```
private List props =new LinkedList();//属性声明表
private List selectors = new LinkedList(); //选择器集
```

其中，属性声明表集合由 CSSName 和 PropertyValue 两部分组成。

ruleset的词法规则，遵循巴科斯范式（BNF: Backus-Naur Form）。这样在处理时就可以正确解析，规则如下：

```
ruleset
  : selector [ ',' S* selector ]*
    '{' S* declaration [ ';' S* declaration ]* '}' S*
  ;
```

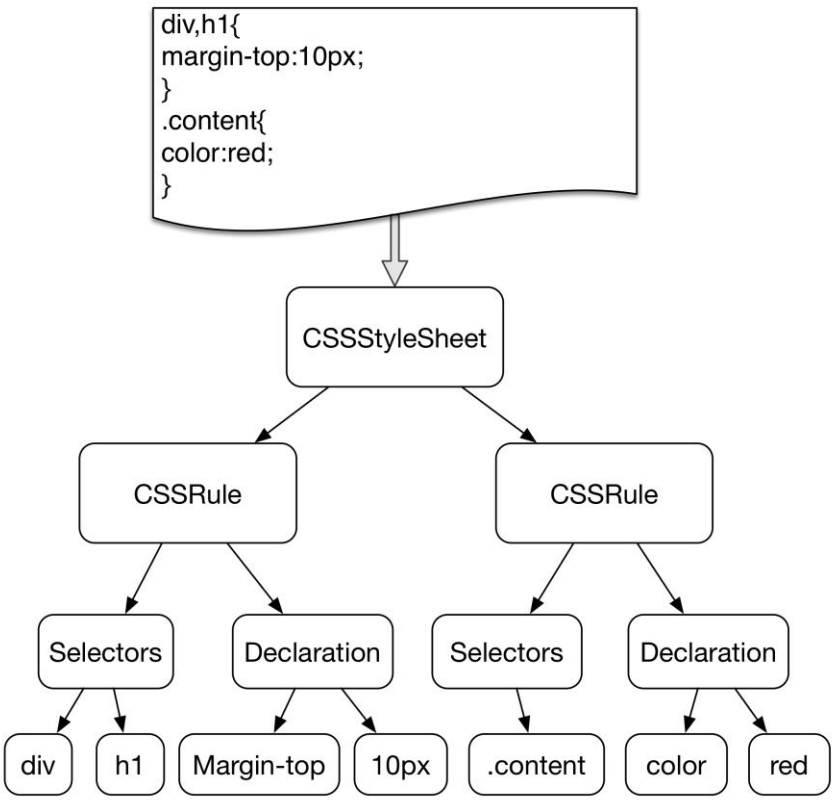


图 4.6　CSS 解析原理示意图

按照图 4.6 示意图所示原理，逐层剥离归类，对 CSS 样式进行正确解析。引擎要正确实现解析，还必须借助词法解析器（Lexer Scanner），通过词法解析器对样式字符串浏览解析出相应的元素信息，其算法非常复杂，这里使用了开源算法。解析出来的样式结果以 StyleSheet 类保存在 Matcher 中的 HashMap 中，称之为 CSS 树。CSS 树和 HTML 的 DOM 树会被合成一棵渲染树（Render Tree），用于计算每个可视元素的布局。同时，它也会作为绘制过程的输入参数，用于绘制屏幕上的每个像素点。实现方法是遍历后重排序生成一棵 CSS 样式规则组成的渲染树（具体实现利用 java.util.TreeMap 类，TreeMap 与 HashMap 的不同之处体现在所有元素保持某种固定顺序）。

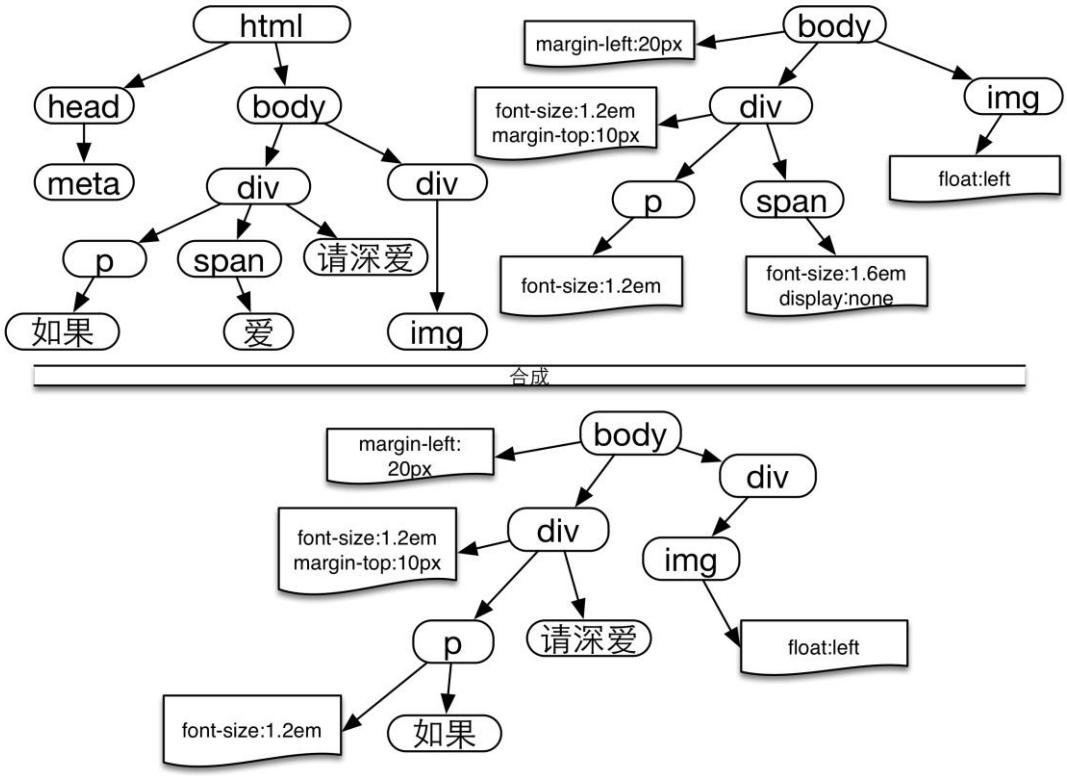


图 4.7　渲染树构建模型图

渲染树的构建如图 4.7，具体构建做法是从 DOM 树的根节点开始，遍历所有的可视节点，忽略不可见元素（比如脚本标签、元数据标签和主题标签等），将 CSS 样式隐藏掉的元素一同忽略掉，比如上图中的 span 元素。由于该元素上显式地定义了属性“display:none”。从 CSS 树中寻找对应的样式规则，并付诸节点，输出可视的节点，计算每个节点的样式，最终形成一棵包含可视信息和相应样式的渲染树。在渲染树构建过程中，针对 img 标签做了特殊的优化处理。从引擎整个工作过程看，工作任务分为两类，渲染文本任务和下载图像任务（不是必须），前者是 CPU 密集型任务，后者是 IO 密集型任务，为了优化引擎速度，在构建渲染树阶段就针对 img 标签启用独立的线程池去下载远程图像，创建专门的 ImageTask 类负责管理任务状态和结果，充分利用绘制过程之前的处理器时间，提升系统效率。

接下来利用渲染树信息和 Java AWT 包及扩展类集布局文档结构进行渲染树布局。布局阶段的输出结果称为“盒模型”（box model）。盒模型精确表达了窗口中每个元素的位置和大小，而且所有的相对度量单位都被转化成了 PDF 文档中上的绝对像素定位。具体做法是从渲染树的根元素开始，逐层处理计算 BlockBox 在绝对定位中的起点、

高和宽等信息。然后计算和设置 Box 的属性信息，例如：设置轮廓，BasicStroke 类定义了图像外轮廓渲染的属性信息，该类有多个属性信息，其中 width 定义了轮廓的宽度，end caps 定义不同的线端，Join 定义了两个线段如何交互，Miterlimit 定义了斜接限制，Dash 定义了虚线分割的数组 dash_phrase 等属性，部分属性有限定值。对于可替代元素，如图像 image 元素要通过缓冲输入流读取调用 paintImage 方法绘制。对于 input 根据类型（type）最终替换成不同的元素。每个 BlockBox 都通过剪切（clip）手法在 Document 中独占一块区域。通过 BoxRenderer 渲染器类进行渲染。每个 Document 都有唯一的一个 BoxRenderer 类。最后需要根据 Box 元素总计估算出一个 document 需要的 page 数目，并存入上下文中，将同一页的元素放入同一个 PageBox 中。布局过程中将重要信息放入上下文中，以便绘制 PDF 文档时使用，如图 4.8 所示。

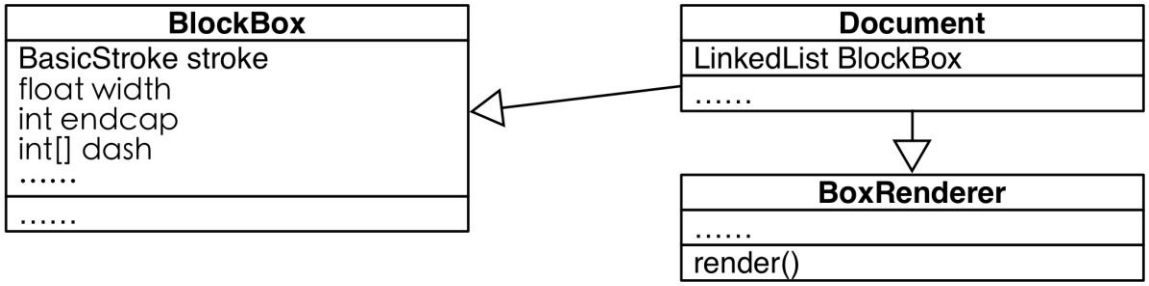


图 4.8　布局模块核心类图

最后一步是绘制 PDF 文档。利用 iText 提供的 Document 类（该类在 com.itextpdf.text 包下）逐页绘制，对每页的 Box 集合，迭代遍历。通过虚拟的输出设备，根据不同的 BOX 调用不同的接口实现。Document 设置的尺寸是标准 A4 纸的纸张大小（宽 21.0 cm 宽，高 29.7cm）。可以设置文档的版本号和加密方式。然后通过 PdfWriter 输出。

```
if (null!=pdfVersion) {
  writer.setPdfVersion(pdfVersion); //设置 pdf 版本
}
if (null!=pdfEncryption) {
writer.setEncryption(
pdfEncryption.getUserPassword(),pdfEncryption.getOwnerPassword(),pdfEncryption.get
AllowedPrivileges(),pdfEncryption.getEncryptionType());      //设置文档加密
}
```

为了适应大规模用户请求和改善了文档输出效果，对整个流程和代码部分地方进行了一些优化。渲染每次都很耗时，如果等待渲染完成之后，再通知用户和合同文档已经渲染，将导致用户体验不佳。在渲染生成任务前，首先将这次任务记录到数据库渲染日志中，在任务完成之后在渲染日志中再次记录，主要因为插入记录比更新记录在多线程下使用事务管理更为简单。通过数据库记录，系统可以及时发现中断或出错

的任务，进行紧急处理。

每次任务都由一个新线程来处理，适当的增加线程能够提高系统的吞吐量，大量线程的创建将消耗大量的计算机资源。因此，引入 Java 的 Executors 多线程框架技术，如果是在单机情况下，可以考虑使用 JDK 提供的 ExecutorService 类，创建一个 Callable 类负责下载图像任务，并提交到 ExecutorService，返回一个描述下载图像执行情况的 Future 类，当主任务完成之后，等待 future.get()方法调用的结果，理想状态下，不用等待，继续执行绘制任务，本项目中因为可以充分利用多台服务器以及 Spring 框架的优点，利用 Spring 提供的多线程框架类 TaskExecutor，其效果等同于 java.util.concurrent.Executor 类，发起线程不必须关心异步线程执行的结果。Spring 中配置 TaskExecutor，设置核心线程数为 15，最大线程为 30，通过依赖注入的方式使用该类，配置方法如下：

```
<bean id="taskExecutor"
class="org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor">
<property name="corePoolSize" value="15"/><property name="maxPoolSize"
value="30"/></bean>
```

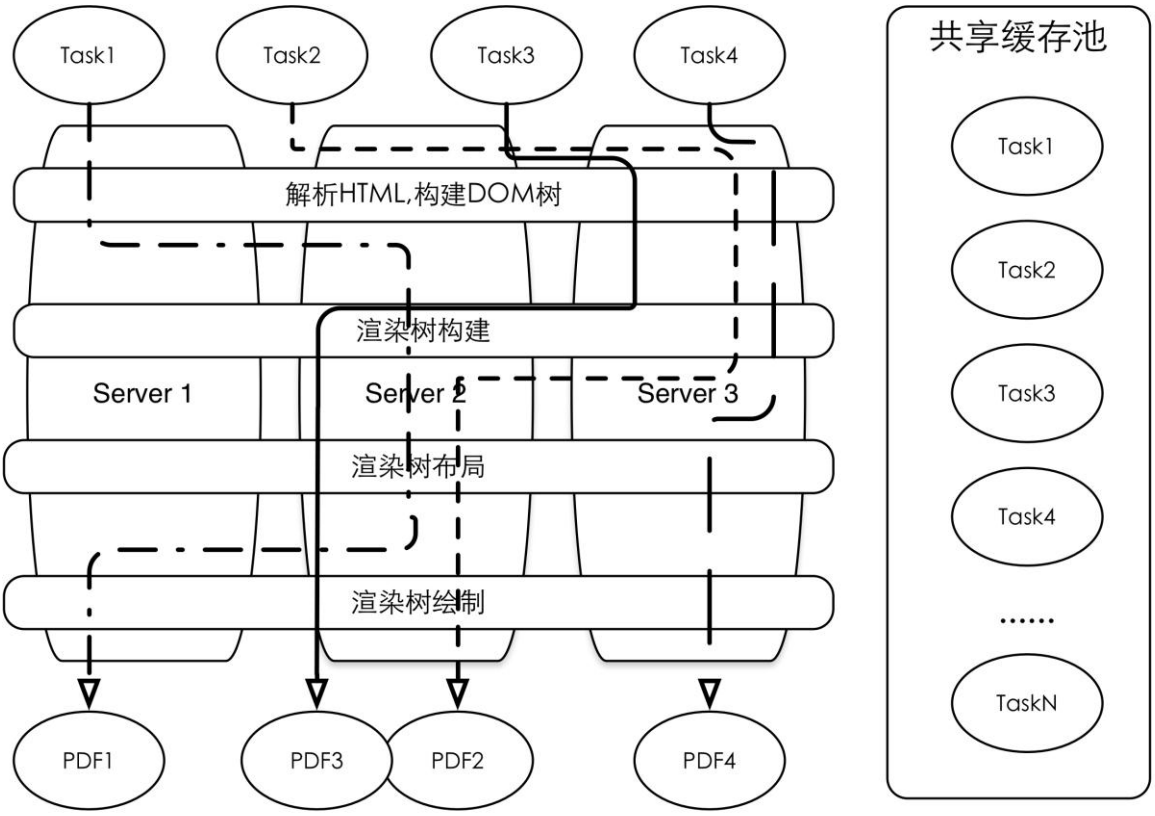


图 4.9　线程优化示意图

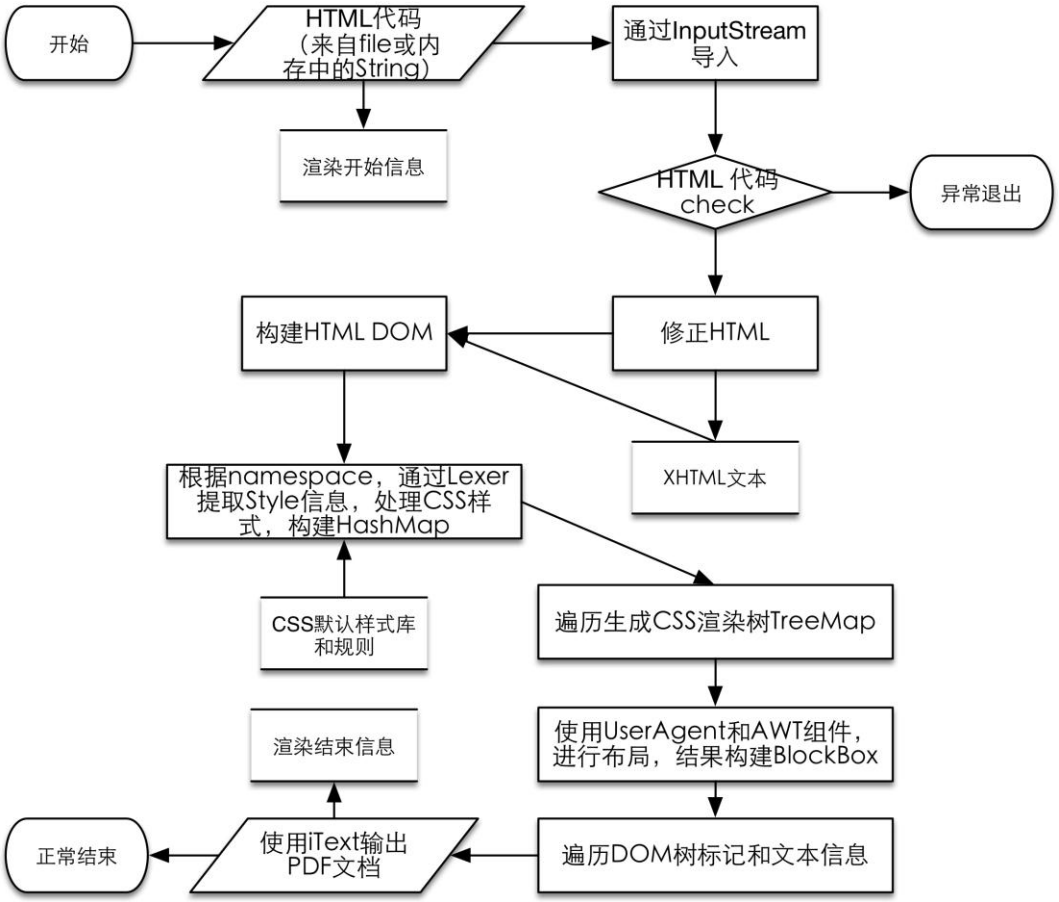


图 4.10　引擎工作流图

渲染流程的四步难以纵向分割，如图 4.9 线程优化示意图所示，每一步的结果都是下一步的原料，所以定义了 Task 类，针对每次任务创建一个对象实体，封装了共享上下文类（SharedContext），该类实现了序列化接口，可以通过网络传递，每次渲染任务分成四步，根据已经执行的步数（step），调用相应下一步方法执行。通过分布式共享缓存 Memcached 缓存每一步的执行结果，这样任务就可以在不同的 Server 上协作执行，任务按照优先级顺序优先分配给不忙的 Server 完成。服务器一般都是千兆网卡，因此 memcached 的 get 和 set 操作的延时非常少。如果一个请求非常着急，那么可以在该步执行完后，提升任务的优先级（privilege），直接在当前线程（current thread）继续执行下一步操作。Task 接口类核心代码如下：

```
private int step =0;//执行到第几步
private int privilege;//任务优先级
private SharedContext sharedContext; //共享上下文
public void parseTree(){};//第一步：处理 HTML 和 CSS
```

```
public void renderTree(){};//第二步：渲染构建
public void layout(){};//第三步：布局
public void toPDF(){};//第四步：绘制 PDF
```

使用 Task 类将流程最核心的四大块，通过异步线程和共享缓存技术实现解耦。提高引擎在多任务情况下的渲染速度，同时将一些通用的组件通过引入共享缓存，方便多个服务器上的多个线程取用资源，进一步提高了渲染效率。

引擎实际运行非常复杂，依据上面内容的阐述，经过归纳总结得出引擎创建一个任务（Task 对象实例）之后的工作流程图，如图 4.10 所示。

4.5 交互接口设计

接口设计是托管平台能够正常提供服务的重要组成部分。网贷平台与托管平台托管合同过程中需要按顺序调用两次接口分别是签名请求接口和托管请求接口，与托管平台发生两次交互操作。接口设计采用 RESTful 风格规范。通过 CMDID 和 Version 来定义接口操作，方便接口的扩展和升级。

接口统一为：http://www.xxxxxx.com/merchant/publicRequests，支持 POST 请求。数据编码方式统一用 UTF-8 格式编码。

表 4.1 罗列了系统提供的核心接口，包括签名请求接口、托管请求接口、用户开户接口、用户登录接口、浏览合同接口，下一步还会增加查询类接口。

表 4.1 接口列表

接口名称	CMDID	接口说明	接口类型
签名请求	Verify	当商户需要托管合同时，调用该接口。商户将需要发送的合同数据以及使用的合同模板编号后者合同文本的 HTML 数据发送过来。	实时后台 POST 请求，返回 JSON 字符串
托管请求	Desposit	当商户拿到托管平台签名之后，用户用商户的公钥签出的信息比对无误之后。用户可以通过浏览器跳转调托管平台合同显示界面。	浏览器跳转
用户开户	Account	如果用户没有在托管平台开过户，需要调用此接口开户。	浏览器跳转
用户登录	Login	如果用户已经开过户，可以调用次接口，跳转到登录页面	浏览器跳转
浏览合同	Find	如果用户想要登录托管平台查询某一份合同，商户平台可以通过传递指定合同标识信息，跳转到相应合同页面	浏览器跳转

表 4.2 罗列了签名请求的具体请求信息，包括操作名称、版本号、商户号、订单号、订单日期、后台回调路径、请求类型、合同数据、附加数据和签名结果。

表 4.2　签名请求 request 表

参数名	参数中文名	参数类型	必须	参数说明
CMDID	操作名称：Verify	变长 String	是	指定每一种操作的类型。
Version	接口版本	定长 2 位	是	指定所调用的操作版本。随着托管系统不断完善，接口也会随之调整，为了方便不能及时升级的商户，所以版本号要不断更新。当前版本为 10。
MerID	商户号	定长 10 位	是	商户在托管平台注册成功后分配的商户唯一编号
OrdID	订单号	定长 20 位	是	订单后有商户负责生成，必须保证唯一
OrdDate	订单日期	定长 8 位	是	订单生成日期，例如：20141212
BgReturnURL	后台回调 URL	变长 128 位	否	托管平台通过后台异步请求方式返回结果数据。商户都应该在应答接收页面输出 RECEIVE_ORDER 加订单号。这样的字符串说明商户已经接受结果数据。注意：使用时不能包含中文字符，和其它禁止的字符，必须是商户的外网地址
Type	请求类型	定长 1 位	是	目前有两种方式：D 和 I 两种。D 表示直接传递文本数据。I 表示传递合同本文参数
MerText	合同数据	变长 String，无限制	是	因为合同文本数据量不固定，所以这里不限制合同文本大小，考虑到合同文本一般大小，所以接口只能用 POST 请求，GET 请求最多只能传递 1024 字节数据。
AddData	附加	变长	否	附加数据，根据用户需要，可以自己附加一些数据

	数据	256 位		
CheckValue	签名结果	定长 256 位	是	CMDID+Version+MerID+OrdIDOrdDate+BgReturnURL+Type+MerText 方式将字符串拼接起来，不能有空格，然后使用提供的插件 jar 包中的 SecureLink 提供的 signMsg 方法进行签名（使用私钥）

表 4.3 罗列了签名请求的返回结果信息，包括后台回调路径、托管订单号、商户签名信息和签名结果等。

表 4.3　签名请求 response 表

参数名	参数中文名	参数类型	必须	参数说明
CMDID	操作名称：Verify	变长 String	是	指定每一种操作的类型。
Version	接口版本	定长 2 位	是	指定所调用的操作版本。随着托管系统不断完善，接口也会随之调整，为了方便不能及时升级的商户，所以版本号要不断更新。当前版本为 10。
MerID	商户号	定长 10 位	是	商户在托管平台注册成功后分配的商户唯一编号
OrdID	订单号	定长 20 位	是	订单后有商户负责生成，必须保证唯一
OrdDate	订单日期	定长 8 位	是	订单生成日期，例如：20141212
BgReturnURL	后台回调 URL	变长 128 位	否	托管平台通过后台异步请求方式返回结果数据。商户都应该在应答接收页面输出 RECEIVE_ORDER 加订单号。这样的字符串说明商户已经接受结果数据。注意：使用时不能包含中文字符，和其它禁止的字符，必须是商户的外网地址。

AddData	附加数据	变长 256 位	否	附加数据，根据用户需要，可以自己附加一些数据
SingInfo	商户签名信息	定长 256 位	是	托管平台对文本数据进行拼接，并用公钥签名，返回，商户在接受回调信息后应该对使用商户的公钥签名产看是托管平台拼接正确。
MerOrdID	托管订单号	定长 16 位	是	该订单号唯一，由商户对每笔请求单独生成。
CheckValue	签名结果	定长 256 位	是	按照 CMDID+Version+MerID+OrdIDOrdDate+BgReturnURL+SignInfo+MerOrdID 方式将字符串拼接起来，不能有空格，然后使用提供的插件 jar 包中的 SecureLink 提供的 signMsg 方法进行签名（使用私钥）

表 4.4 罗列了托管请求的具体请求信息，包括订立用户号、其它用户号、签名信息和签名结果等。

表 4.4　托管请求 request 表

参数名	参数中文名	参数类型	必须	参数说明
CMDID	操作名称：Verify	变长 String	是	指定每一种操作的类型。
Version	接口版本	定长 2 位	是	指定所调用的操作版本。随着托管系统不断完善，接口也会随之调整，为了方便不能及时升级的商户，所以版本号要不断更新
MerID	商户号	定长 10 位	是	商户在托管平台注册成功后分配的商户唯一编号

UserCustID	订立用户号	定长 16 位	是	该合同是和谁订立的，此接口只支持一位用户的情况。
OtherCustID	其它用户号	变长 256 位	否	该合同的其它参与者，使用 JSON 格式传递其它用户信息。
OrdID	订单号	定长 20 位	是	订单后有商户负责生成，必须保证唯一
OrdDate	订单日期	定长 8 位	是	订单生成日期，例如：20141212
BgReturnURL	后台回调 URL	变长 128 位	是	托管平台通过后台异步请求方式返回结果数据。商户都应该在应答接收页面输出 RECEIVE_ORDER 加订单号。这样的字符串说明商户已经接受结果数据。注意：使用时不能包含中文字符，和其它禁止的字符，必须是商户的外网地址。
FtReturnURL	前台回调 URL	变长 128 位	是	托管平台前台返回路径，返回结果以表单数据提交。
SignInfo	签名信息	定长 256 位	是	托管平台签名信息，商户必须校验签名信息的正确，并返回。表示商户确认托管合同正确无误。
AddData	附加数据	变长 256 位	否	附加数据，根据用户需要，可以自己附加一些数据
CheckValue	签名结果	定长 256 位	是	按照 CMDID+Version+MerID+OrdID OrdDate+UserCustID+OtherCustID+FtReturnURL+BgReturnURL+SignInfo 方式将字符串拼接起来，不能有空格，然后使用提供的插件 jar 包中的 SecureLink 提供的 signMsg 方法进行签名(使用私钥)

表 4.5 罗列了托管请求返回结果信息，包括前台回调路径、托管状态、用户公钥列表等。

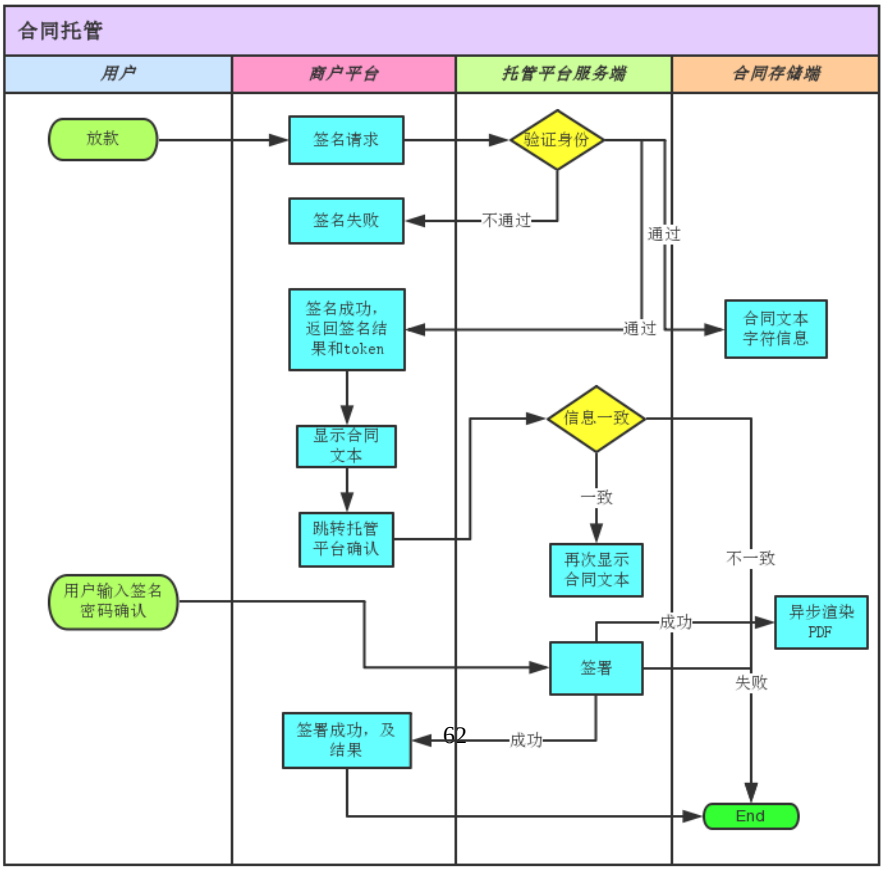
表 4.5　托管请求 response 表

参数名	参数中文名	参数类型	必须	参数说明
CMDID	操作名称：Verify	变长 String	是	指定每一种操作的类型。
Version	接口版本	定长 2 位	是	指定所调用的操作版本。随着托管系统不断完善，接口也会随之调整，为了方便不能及时升级的商户，所以版本号要不断更新
MerID	商户号	定长 10 位	是	商户在托管平台注册成功后分配的商户唯一编号
UserCustID	订立用户号	定长 16 位	是	该合同是和谁订立的，此接口只支持一位用户的情况。
OtherCustID	其它用户号	变长 256 位	否	该合同的其它参与者，格式传递其它用户信息。
OrdID	订单号	定长 20 位	是	订单后有商户负责生成，必须保证唯一
OrdDate	订单日期	定长 8 位	是	订单生成日期，例如：20141212。
BgReturnURL	后台回调 URL	变长 128 位	是	托管平台通过后台异步请求方式返回结果数据。商户都应该在应答接收页面输出 RECEIVE_ORDER 加订单号。这样的字符串说明商户已经接受结果数据。注意：使用时不能包含中文字符，和其它禁止的字符，必须是商户的外网地址。
FtReturnURL	前台回调 URL	变长 128 位	是	托管平台前台返回路径，返回结果以表单数据提交。

DepState	托管状态	定长 1 位	是	如果用户登录托管平台后接受托管，返回 Y，如果用户拒绝托管返回 N。
UserPubKeys	用户公钥列表	变长	是	返回用户和其它用户的公钥地址。
AddData	附加数据	变长 256 位	否	附加数据，根据用户需要，可以自己附加一些数据。
CheckValue	签名结果	定长 256 位	是	按照 CMDID+Version+MerID+OrdIDOrdDate+UserCustID+OtherCustID+FtReturnURL+BgReturnURL+DepState 方式将字符串拼接起来，不能有空格，然后使用提供的插件 jar 包中的 SecureLink 提供的 signMsg 方法进行签名(使用私钥)

接口后端连接一个名为 Nume 的队列系统，这个队列保证接收到的请求至少被处理一次，对于异常情况，激活系统预警设置，以加强系统的健壮性。

后台回调出口前置一个名为 Bame 的队列系统，这个队列保证返回信息，至少被

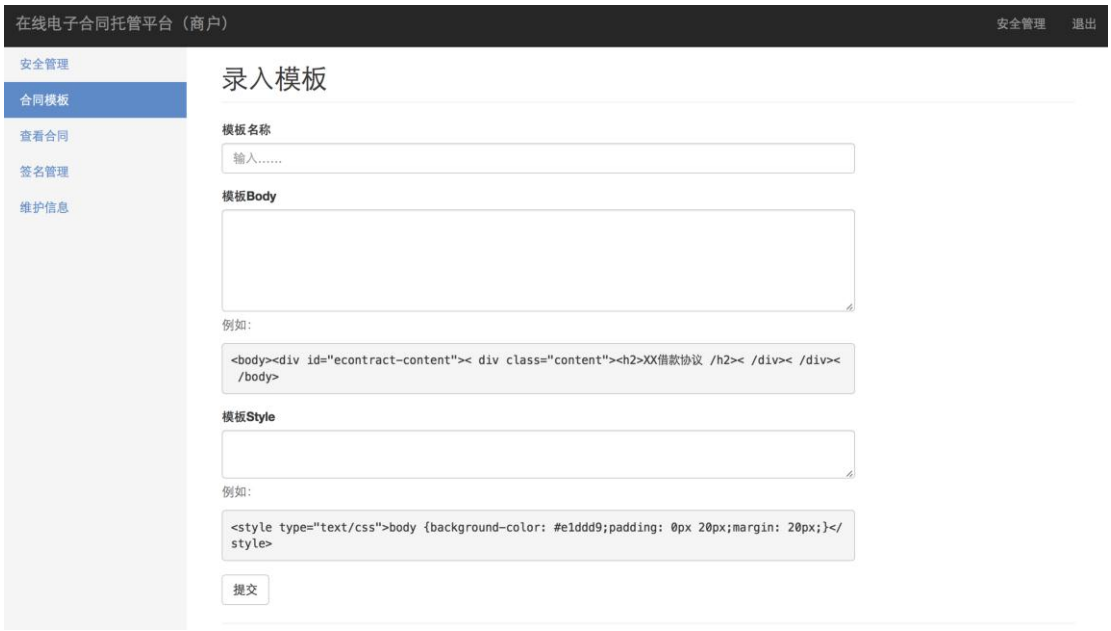


商户系统接收一次，对于超过 24 小时没有响应的回调信息，启动系统预警设置，及时告知商户。

图 4.11 合同托管流程图

4.6 关键业务设计与实现

系统主要业务是为用户和商户提供互联网合同托管服务。用户可以通过托管平台实现自动存储、高效管理属于自己的电子合同；托管平台也为商户（平台）提供电子合同生成和托管接口，商户（平台）可以调用该接口提供的服务，生成合同、管理合同等。主要服务对象是用户和商户（平台）。主要业务包括：合同文本的生成和存储、用户查询合同、商户查询合同、用户账户管理、商户账户管理和消息通知等功能。下面对关键业务托管业务和消息业务进行说明。

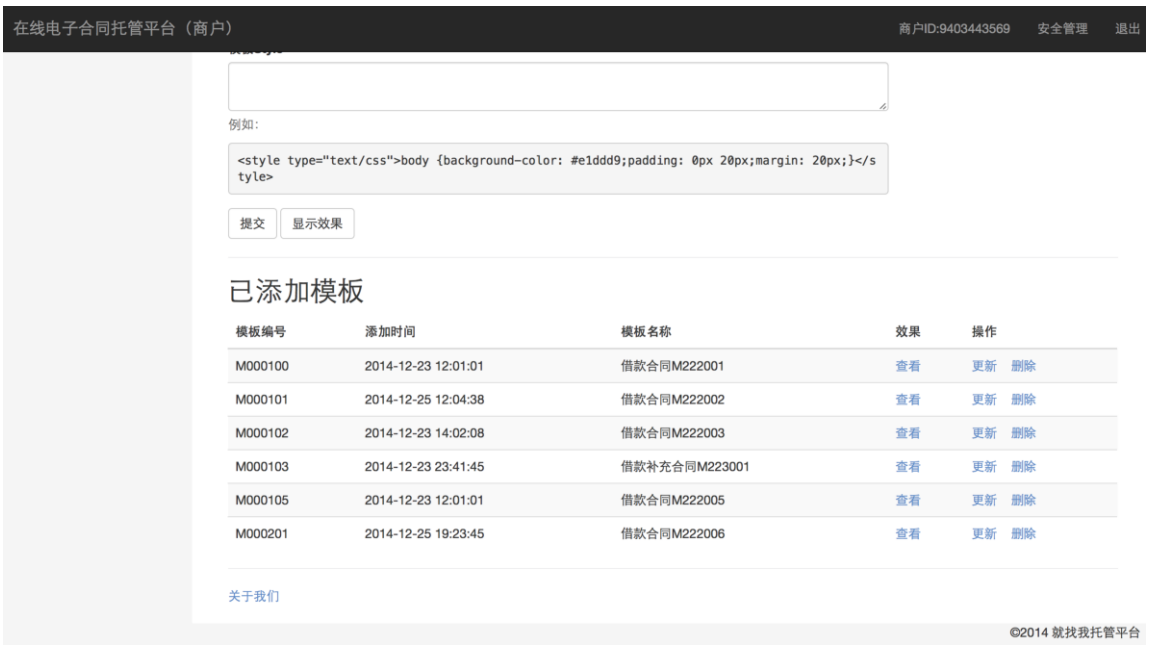


4.6.1 托管业务

图 4.12 模板录入界面

如图 4.11 合同托管流程所示，合同托管业务由用户发起，用户在满标之后，做确认放款操作，启动合同托管流程。商户平台接收到放款请求之后，合成合同文本字符串（HTML 形式），按照“签名请求”接口规定的形式，使用签名 jar 包进行签名操作。将签名数据通过后台实时 Post 请求发送到托管平台服务端，服务端根据 MerID 从数据库中查找商户的公钥，用商户的公钥将商户发来的信息，按照约定规则组合进行解密，将解密的数据与原始数据比对，看看是否一致，如果哦不一致，返回失败结果，如果

一致，验证身份通过，根据 Type 字段确定商户发送过来的是合同文本还是合同数据，如果是合同数据，需要根据合同数据找到相应合同模板，合成合同文本，然后将合同文本信息存储在合同存储的非关系型数据库中，根据合同编号做好键值映射，以便之后的查询操作。同时，对合同文本和商户公钥组合用托管平台自身的私钥进行签名操



作。将签名结果以及随机生成的 token 返回给商户，商户保存 token，以便用于托管合同请求。用户点击放款，商户可以在同步做完这些操作之后，在返回用户合同页面，让用户浏览合同信息后采取手动跳转，也可以直接跳转到托管平台浏览合同信息，托管平台的合同信息不支持隐藏属性。用户访问界面并登录成功后，呈现合同文本信息，在合同的底部有一个签名按钮和一个取消签名按钮，用户点击取消签名则返回签署失败的结果，商户根据返回结果设计应答界面。用户确认合同无误，并意欲签名，则点击签名按钮进行签名，这时跳出输入签名密码的页面，需要用户输入签名密码，用户密码通过验证，就表示用户已经授权系统进行签名。系统通过密码，使用 DES 算法解密密钥信息，并调用用户私钥对合同文本以及商户公钥信息组成的字符串进行签名，签名结果表明用户已经认可某商户的电子合同。同时，系统后台创建新 Task，将合同文本数据渲染成 PDF 文档，并将签名信息生成二维码，附加在电子合同的末尾，保证不会因为难以理解的签名哈希值干扰用户的视线。关于合同模板的生成商户可以选择通过商户管理平台录入和生成。

图 4.13 模板列表界面

为了减少开发人员的工作量和减少商户平台和托管平台之间的数据交互量，面向商户的管理系统中，提供了模板管理的功能模块。用户可以在模板管理模块录入合同

模板，并且可以查看已提交生效的合同列表。在录入模板模块，用户需要提交模板名称，此名称由商户自己定义，通常可以使用合同本身的名称。模板 Body 指用户提交合同 HMTL 文本中 Body 定义的部分，模板的 Style 部分指合同文本中用户使用到的内部样式。商户登录系统之后，选择合同模板，进入录入合同模板界面，如图 4.11 所示，用户只要按要求将样式表和 body 体分别录入，确定之后点击提交按钮就可以生成一个模板，并生成一个模板序列号，用户可以通过模板列表界面查询刚生成的模板号，如图 4.12 所示。商户在使用模板生成合同时，只要提交模板的序列号和数据就可以生成合同，用户发来的模板样例，见附录。

4.6.2 消息业务

为了实现系统业务之间的解耦，所有消息包括系统消息、用户消息以及模块之间通信等都经过消息中间件进行分发和消费，消息队列在整个系统的构建过程中起到了桥梁的关键作用。项目初始过程中，系统使用 ActiveMQ 作为消息中间件，但其相对太重、效率较低、维护成本高，因此换成更轻量级、更迅速的 MQS 作为消息队列。其可以设置不同队列名称，镜像出多个消息队列，开发者不用关心队列服务器内部运作机制。在设计过程中，充分利用 P2P 模式和发布者订阅者两种模式，设计开发消息队列系统和服务。

消息在数据库中以 Json 格式存储：

```
{"index":2,"version":1,"name":"签署成功
","pId":"SYE20150311111523DEP","params":"签署时间##@##12:20"}
```

消息发送者（Sender）通过 sendMessage 方法发送到消息队列中。

消息接受者（Receiver）不断轮询中间件的监听端口，获取消息。消息模块单独作为一个程序，使用 shell 编写启动脚本，独立运行。通过 open 参数作为钩子，控制系统的启动和关闭，代码如下：

```
private void listen() {
// 开始监听
  while (open) {
// 获得消息
                final String message = MQSUtils.receiveMessage(queueName);
                // 如果有消息那么一直消费，负责等待 1 秒继续消费
                if (GlobalMethod.isStrNonEmpty(message)) {
                    taskExecutor.execute(new Runnable() {
                        @Override
                        public void run() {
```

```
                                System.out.println("start handle:" + message);
                                consumer.consume(message);
        }
                        });
                    } else {
                        try {
                            //睡眠 1 秒
                            Thread.sleep(1000);
                        } catch (InterruptedException e) {
logger.error(e.getMessage());
        }
                    }
                }
            }
```

消息处理框架整体采用职责链模式设计，定义一个消息类并且实现序列化，消息类定义了目标处理方法，职责链根据目标处理方法调用相应的 Handler 进行处理。处理消息的 Handler 具体有 SMSHandler、NoticeHander、EmailHandler、UMHander 四种实现，将四种实现注入 MessageHandler 类中，分别用来处理短信消息、系统通知、邮件通知和用户消息。

4.7 本章小结

本章从系统设计的原则出发，概述了系统的架构、部署设计与实现，数据库的 E-R 模型设计，然后介绍了系统核心——渲染引擎的设计与实现，关键接口的设计，最后详述了系统的关键业务的设计与实现，详细介绍了托管业务和消息业务的设计与实现。

第五章 系统测试

5.1 功能测试

软件测试是软件开发的重要阶段[30]。功能测试就是对产品的功能进行测试，根据功能列表，编写测试用例，进行测试。面向对象语言中，单元测试的基础单元是类。首先对核心代码函数，编写单元测试用例，测试其核心性能是否满足要求。

本项目的核心功能是从商户端向服务端接口发送生成测试合同文本的请求，测试服务端能否成功接受并达到预期的渲染效果。通过逐步修改代码和不断调试，最终得到较为满意的输出效果，其中测试文本基本包含了合同文档一般需要的属性元素。

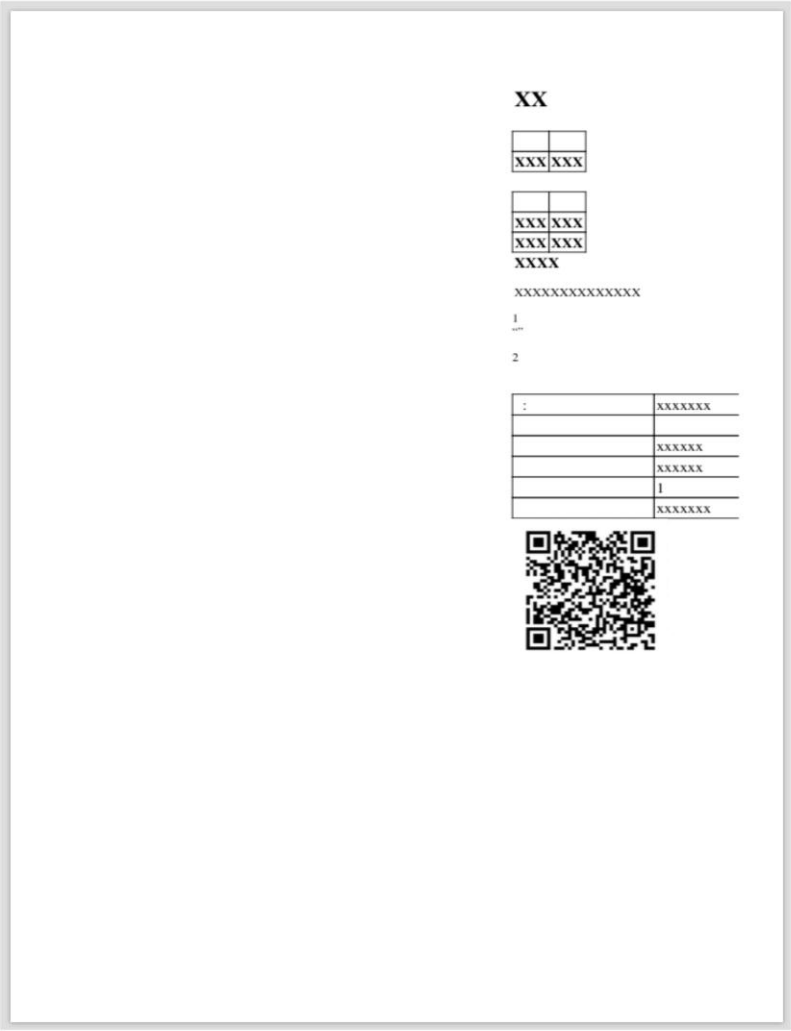
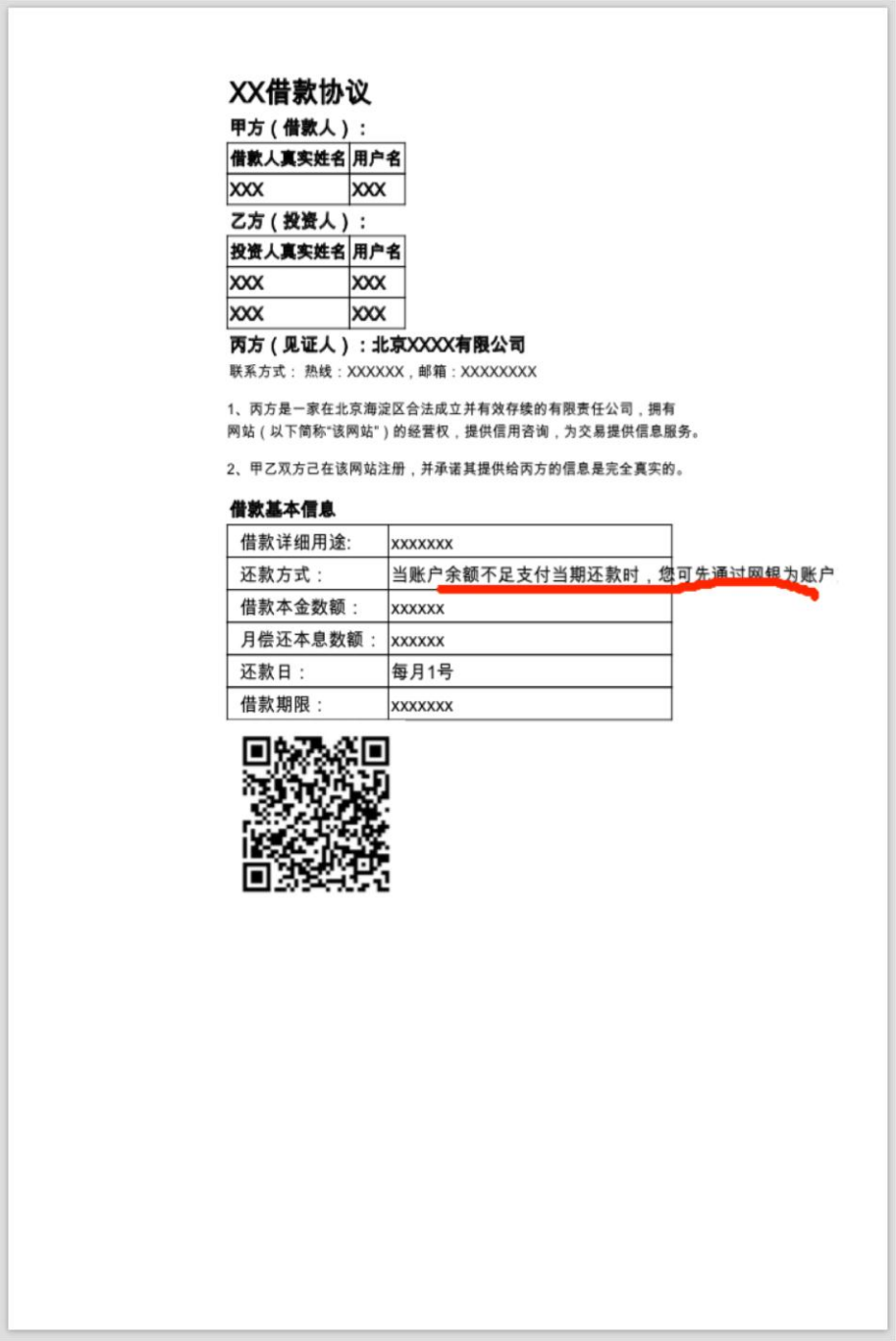


图 5.1 测试输出结果 1

最初状态下生成的合同文档中文是乱码，如图 5.1 所示，通过引入 itextpdf-Asian 中文字体包，在 HTML 处理过程中前置一个代码纠正器，为每一份 body 追加<body style="font-family:'Arial Unicode MS '">（网贷平台实际开发中，开发人员不必再添加）。而且显示的位置不对，通过追加样式过滤器对输入样式进行调整，如图 5.2 所示。



XX借款协议

甲方（借款人）：

借款人真实姓名	用户名
XXX	XXX

乙方（投资人）：

投资人真实姓名	用户名
XXX	XXX
XXX	XXX

丙方（见证人）：北京XXXX有限公司

联系方式： 热线：XXXXXX，邮箱：XXXXXXXX

1、丙方是一家在北京海淀区合法成立并有效存续的有限责任公司，拥有网站（以下简称"该网站"）的经营权，提供信用咨询，为交易提供信息服务。

2、甲乙双方已在该网站注册，并承诺其提供给丙方的信息是完全真实的。

借款基本信息

借款详细用途:	xxxxxxx
还款方式：	当账户余额不足支付当期还款时，您可先通过网银为账户
借款本金数额：	xxxxxx
月偿还本息数额：	xxxxxx
还款日：	每月1号
借款期限：	xxxxxxx

图 5.2　测试输出结果 2

表元素中文存在文字不能自动换行问题。通过修改符号位置计算参数值和条件判

断的方法、进一步调试得到了目标效果如图 5.3 所示效果。

XX借款协议

甲方（借款人）：

借款人真实姓名	用户名
XXX	XXX

乙方（投资人）：

投资人真实姓名	用户名
XXX	XXX
XXX	XXX

丙方（见证人）：北京XXXX有限公司

联系方式： 热线：XXXXXX，邮箱：XXXXXXXX

1、丙方是一家在北京海淀区合法成立并有效存续的有限责任公司，拥有 网站（以下简称"该网站"）的经营权，提供信用咨询，为交易提供信息服务。

2、甲乙双方已在该网站注册，并承诺其提供给丙方的信息是完全真实的。

借款基本信息

借款详细用途:	xxxxxxx
还款方式：	当账户余额不足支付当期还款时，您可先通过网银为账户充值。 充值完成后，您可以自行点击还款。
借款本金数额：	xxxxxx
月偿还本息数额：	xxxxxx
还款日：	每月1号
借款期限：	xxxxxxx

图 5.3　测试输出结果 3

```
        long total = 0;
        System.out.println("开始 1 ");
        for (int i = 0; i < 20; i++) {
            long start = System.currentTimeMillis();
            CSSParser p = new CSSParser(errorHandler);  //创建CSS处理器
            p.parse(null, 0, new StringReader(longTest.toString()));
            long end = System.currentTimeMillis();
            total += (end-start);
        }
        System.out.println("第一次平均时间 " + (total) + " ms");
//
        total = 0;
        System.out.println("开始 2 ");
        for (int i = 0; i < 20; i++) {
            long start = System.currentTimeMillis();
            CSSParser p = new CSSParser(errorHandler);
            p.parse(null, 0, new StringReader(longTest.toString()));
            long end = System.currentTimeMillis();
            total += (end-start);
        }
        System.out.println("缓存平均时间" + (total) + " ms");

        CSSParser p = new CSSParser(errorHandler);

        total = 0;
        System.out.println("开始 3 ");
        for (int i = 0; i < 20; i++) {
            long start = System.currentTimeMillis();
//            for (int j = 0; j < 20; j++) {
//                p.parse(null, 0, new StringReader(test));
//            }
            p.parse(null, 0, new StringReader(test));
            long end = System.currentTimeMillis();
            total += (end-start);
        }
```

图 5.4　CSSParser 单元测试代码示例图

5.2 性能测试

完成了系统设计研发阶段工作之后，接下来的工作是进行系统性能测试，高质量的测试工作既能确保系统正常上线，又保证了软件的工程质量。软件系统随着信息技术的快速发展，也日益复杂，单纯功能测试并不一定能保证系统的稳定性和功能正确性。通过进一步的性能测试，优化和提升系统整体处理能力。

进行性能测试之前，首先使用 Jtest 工具进行静态代码检查，进行代码优化和测试。系统多处使用了多线程和输入输出流。使用 Jtest 测试系统部署在服务器长期运行时能否保持稳定，正确释放资源。Jtest 是 Parasoft 公司推出的一款针对 Java 语言的自动化代码优化和测试工具，它通过自动化实现对 Java 应用程序的单元测试和编码规范校验，从而提高代码的可靠性以及 Java 软件开发团队的开发效率。使用 Jtest 进行静态代码分析，登录 Parasoft 公司官网申请试用版授权，在 Intellij IDEA 中安装。创建 Jtest 实例，并导入项目代码。运行 Jtest，根据结果修改源码，查找内存可能的泄露点。以进行有针对性的修改。

针对 CSS 渲染非常耗时的问题，专门对 CSSParser 处理性能进行了多次测试。如图 5.4 单元测试代码所示，分别测试了利用不同解析器初次渲染，利用缓存渲染和使用同一解析器渲染的结果如图 5.5 所示，初次解析耗时 207ms，通过缓存解析耗时 5ms，性能大幅提升，使用同一解析器，而不是每次解析都创建，耗时仅 6ms，同样大幅提

升解析性能。系统针对相同合同文本使用缓存和使用同一解析器处理能够大大提升系统的性能数据。

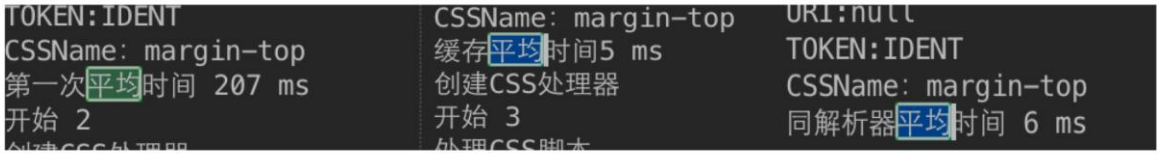
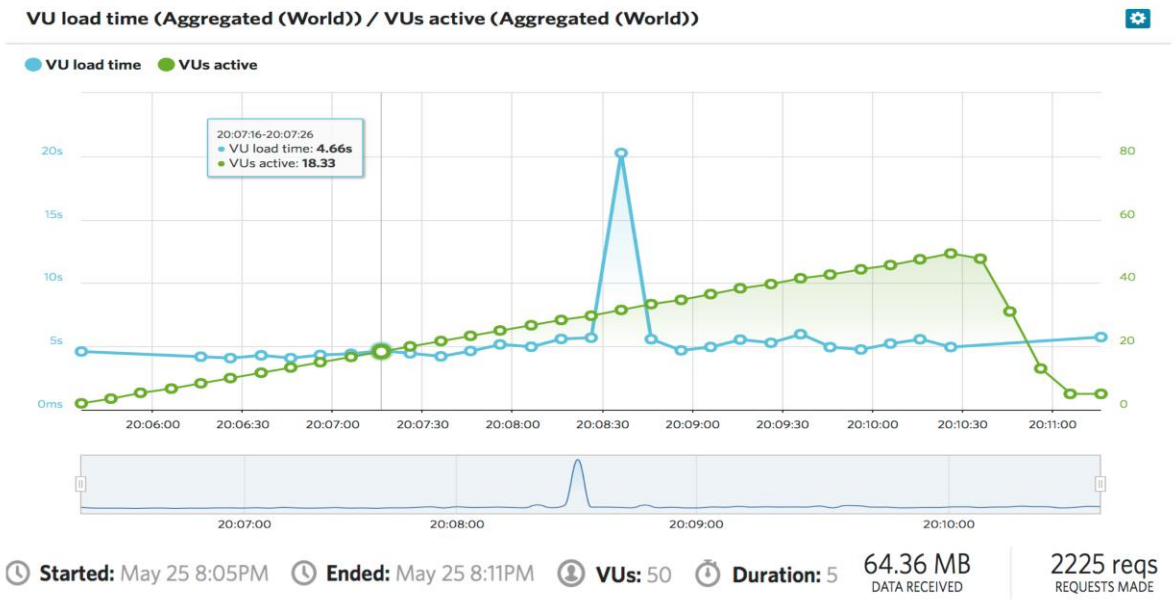


图 5.5　CSSParser 处理输出日志图

最后进行系统并发性能测试，了解系统的吞吐能力。测试使用第三方在线测试工具 loadimpact。系统模拟创建了 50 个虚拟用户（VU），在 5 分钟内一共进行了 2225 次请求操作，对随机多个模板字符串进行渲染，结果如图 5.6 所示，在虚拟用户数缓慢增加过程中，系统基本保持平稳运行。

图 5.6　系统吞吐能力情况图



第六章 总结与展望

6.1 总结

在线合同托管系统试图尝试变革传统的在线电子合同订立存管流程，明确了合同托管业务的服务主体是商户和普通用户。设计和构建了一个可以为绝大多数网贷平台提供服务的合同托管业务流程和部分实现。本平台的设计明确提出了合同托管系统的业务范围、业务流程、系统设计和具体实现，在本论文第四章关键业务的设计与实现中做了详细的描述。这一流程严格遵守了相关法律和规范的要求，特别是遵循了订立和存储分开的原则。

本系统在设计与开发过程中需要解决的核心问题是 PDF 的渲染问题。经过多种开源渲染技术尝试比较之后，最终寻求根据业务实际特点和需求，改进优化，使之成为一款适应需求的 HTML 渲染引擎，增强了它的功能，特别是多任务处理能力，并对主体功能输出效果进行了测试优化。

但是平台本身还有很多不足之处，业务功能模块需要根据真实业务做大量调整，需要重新梳理消息中间件在哪些业务中使用，可以起到更好的系统级优化效果，增强稳定性。对于系统安全，仍需做大量基础工作，譬如找出流程上可能出现的安全问题，并针对这些问题具体解决。重点进一步优化 HTML 渲染引擎，适应大规模并发需求，改进 Lexer 匹配算法，提高 CSS 处理和渲染树布局以及 PDF 绘制速度。目前来看，Lexer 匹配过程消耗了大量的处理器时间，过多的重复调用，需要改进。

6.2 展望

目前引入合同托管在网贷平台中仍属于个例，证据托管模式“形式大于意义”，电子数据存在着可以被无痕篡改的漏洞。由网贷平台委托的第三方证据托管机构，如果不能确保中立地位，不排除其联合不良网贷平台灭失或篡改证据，反而会增大消费者权益保护的难度。未来合同托管平台将会将重点放在如何满足合同审查，证据托管和维权触发三个核心诉求上，希望以此消除投资人对网贷平台合同安全性的顾虑和降低受害人实际维权的困难。

该项目正式运营后，能够大大降低众多网络平台合同模块的设计难度，降低开发人员的工作量。未来随着商业化运营，以及团队正规化建设，可以以此为基础，进一步完善系统的业务流程，开发诸如计费系统、数据分析系统等更多功能模块，开放更多接口，形成面向客户的、完善的在线电子合同托管系统。未来系统工作重点在优化

文件存储系统和渲染速度上，使之更加高效，创建更高效的合同生成引擎。合同托管业务会是一个规模很大、很有潜力、很有价值的市场，可以作为互联网金融安全保障的技术之一。目前很多厂商已经意识到合同托管的重要性。合同托管平台未来在电子证据、电子政务、电子档案、电子商务和互联网金融领域可大有作为。

参考文献

[1] 马文良. 互联网金融:掀起第三次“金融革命”[J]. 中关村, 2014, (02): 66-67.

[2] 谢平, 邹传伟. 互联网金融模式研究[J]. 金融研究, 2012, (12): 11-22.

[3] 闫淼. 中国 P2P 借贷平台模式、问题及对策研究[D]. 中国社会科学院研究生院, 2014.

[4] 田光宁. 互联网金融发展的理论框架与规制约束[J]. 宏观经济研究, 2014, (12): 42-48+111.

[5] 中华人民共和国商务部. 电子合同在线订立流程规范. 中华人民共和国商务部: 中国质检出版社 中国标准出版社, 2013: 12.

[6] Gu J, Zhu X. Designing and Implementation of an Online System for Electronic Contract Negotiation Based on Electronic Signature[M]. 9. 2014.

[7] Kwok T, Nguyen T, Lam L, et al. A Web-based and Email Driven Electronic Contract Management System[J]. e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on, 2007: 149 - 156.

[8] Jianzheng Y, Yunchao X. Design and Application of a Network Platform for Electronic Contract Making[J]. Advances in Intelligent and Soft Computing, 2012.

[9] Chatterjee K, Raman V. Assume-guarantee synthesis for digital contract signing[J]. Formal Aspects of Computing, 2014, 26(4): 825-859.

[10] 杨晨阳. 基于 EXT 和 AJAX 的富客户端合同管理系统的设计与实现[D]. 北京邮电大学, 2009.

[11] 段苏. 海尔公司合同管理系统的设计与实现[D]. 吉林大学, 2014.

[12] 黄建华, 赵长亮. 电子合同的法律效力问题及对策研究[J]. 电子科技大学学报, 2009, 38(z1): 117-119.

[13] 张燕妮. 电子商务合同若干问题研究[D]. 北京大学, 2005.

[14] 罗清彩. 基于电子签章技术的电子合同平台设计与实现[D]. 华东师范大学, 2009.

[15] 黄业文. 我国电子合同法律问题研究[D]. 华东政法学院华东政法大学, 2006.

[16] 张楚. 电子商务法教程（第 2 版）[M]. 清华大学出版社, 2011.

[17] 高富平. 在线交易法律规制研究报告[M]. 北京大学出版社, 2005.

[18] 计磊, 李里, 周伟. 精通 J2ee-Eclipse, Struts, Hibernate 及 Spring 整合应用案例 [J][D].

[19] 胡启敏, 薛锦云, 钟林辉. 基于 Spring 框架的轻量级 J2EE 架构与应用[J]. 计算机工程与应用, 2008, (05): 115-118+133.

[20] Zhang D, Wei Z, Yang Y. Research on Lightweight MVC Framework Based on Spring MVC and Mybatis[J]. Computational Intelligence and Design (ISCID), 2013 Sixth International Symposium on, 2013: 350 - 353.

[21] Vegh A: MySQL Database Server, Web Development with the Mac®: Wiley Publishing, Inc., 2010: 317-340.

[22] Welcome to Apache Shiro[EB/OL]. http://shiro.apache.org/documentation.html.

[23] 张周群. 第三方安全支付平台设计与实现[D]. 复旦大学, 2008.

[24] 王丽萍, 秦永平. 基于 iText 的 PDF 报表设计[J]. 电脑知识与技术(学术交流), 2007, (08): 492-493+574.

[25] Java.net. Welcome to Flying Saucer[EB/OL]. https://xhtmlrenderer.java.net.

[26] 刘家真. 文件保存格式与 PDF 文档[J]. 档案学研究, 2002, (02): 46-51.

[27] 徐震. 电子认证服务标准体系研究[J]. 电子商务, 2006, (01): 16-29.

[28] Lin B, Chen Y, Chen X, et al. Comparison between JSON and XML in Applications Based on AJAX[J]. Computer Science & Service System (CSSS), 2012 International Conference on, 2012: 1174 - 1177.

[29] Wei-Ping Z, Ming-Xin L, Huan C. Using MongoDB to implement textbook management system instead of MySQL[J]. Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, 2011: 303 - 305.

[30] Du Q, Zhang H. An Effective Design Method of System Function Test Cases[C]. Computer and Management (CAMAN), 2011 International Conference on, 2011: 1 - 4.

附录

测试模板样例之一

```
<html lang="zh-CN">
<head>
<style type="text/css">
        @page {
            margin-left: 5cm;
            size: 210mm 297mm;
    }
        html {
            padding: 5px;
        margin: 10px;
        }
        body {
            background-color: #ffffff;
            padding: 10px 10px;
        margin: 10px;
        }
        .title {
            margin: 2px;
   }
        p {
      font-size: 0.8em;
        }
        table {
            border-collapse: collapse;
            border: none;
            table-layout: fixed
        }
        td {
            border: solid #000 1px;
            /*text-overflow: ellipsis;*/
            /*overflow: hidden;*/
            /*white-space: nowrap;*/
          padding: 2px
        }
</style>
</head>
```

```
<body>
<!--合同细节-->
<div id="econtract-content">
<div class="content">
<h2 class="title">XX 借款协议</h2>
<h4 class="title">甲方（借款人）：</h4>
<table style=" background:#fff;  font-weight:bold; font-size:15px; border:1px">
<tr>
<td>借款人真实姓名</td>
<td>用户名</td>
</tr>
<tr>
<td>XXX</td>
<td>XXX</td>
</tr>
</table>
<h4 class="title">乙方（投资人）：</h4>
<table style=" background:#fff;  font-weight:bold; font-size:15px; border:1px">
<tr>
<td>投资人真实姓名</td>
<td>用户名</td>
</tr>
<tr>
<td>XXX</td>
<td>XXX</td>
</tr>
<tr>
<td>XXX</td>
<td>XXX</td>
</tr>
</table>
<h4 class="title">丙方（见证人）：北京 XXXX 有限公司</h4>
<p class="title">
联系方式：<span>热线：XXXXXX，邮箱：XXXXXXXX</span>
</p>
<p>1、丙方是一家在北京海淀区合法成立并有效存续的有限责任公司，拥有
网站（以下简称“该网站”）的经营权，提供信用咨询，为交易提供信息服务。</p>
<p>2、甲乙双方已在该网站注册，并承诺其提供给丙方的信息是完全真实的。</p>
<h4 class="title">借款基本信息</h4>
<table style="background:#fff; width:400px;">
<tbody>
<tr>
```

```
<td width="140px">  借款详细用途:</td>
<td>xxxxxxx</td>
</tr>
<tr>
<td width="140px">  还款方式：</td>
<td>当账户余额不足支付当期还款时，您可先通过网银为账户充值。充值完成后，您可以自行点击还款。</td>
</tr>
<tr>
<td width="140px">  借款本金数额：</td>
<td>xxxxxx</td>
</tr>
<tr>
<td width="140px">  月偿还本息数额：</td>
<td>xxxxxx</td>
</tr>
<tr>
<td width="140px">  还款日：</td>
<td>每月 1 号</td>
</tr>
<tr>
<td width="140px">  借款期限：</td>
<td>xxxxxxx</td>
</tr>
</tbody>
</table>
<img src="http://pic.baike.soso.com/p/20130307/20130307134358-1958736496.jpg" width="160" height="160"></img>
</div>
</div>
</body>
</html>
```

致谢

时光荏苒，不经意间从入学走到了毕业，三年的研究生生涯即将画上圆满的句号。在三年的时间里我苦练技术，立志成为有用之人。

在此衷心感谢我的导师李杰老师，李老师是我人生的良师益友，一直不厌其烦地指导我、鼓励我，为我答疑解惑。李老师治学严谨，诲人不倦，和蔼可亲，她和其学生都成为了很好的朋友。三年下来，总感觉对李老师亏欠太多，回报太少。在这里祝老师身体健康，祝老师女儿洋洋茁壮成长。

同时也要感谢在北大给予我帮助的其他老师，段莉华老师、孙圣力老师、黄雨老师和刘京老师等，段老师和蔼可亲，孙老师气宇轩昂，黄老师英明过人， 刘老师手有余香，在此祝各位老师事业顺心，身体健康。

感谢和我一起学习和玩耍的伙伴，刘漪多、于涛、吴晓萍、金海龙、董丹丹、宋兵宵、高强、戴威、李坤、陈智涵和蔡立婕等同学；我们经常互通讯息，交流心得；他们给了我莫大的帮助，让我不再感觉孤单。感谢台湾生，他们是谦和做人的典范，也特别感谢张勇大哥，富有感召力，让李老师的团队，跨越年级凝聚在一起。

感谢北京大学给了我更大的舞台，将来在社会工作中，我要自信地面对困难，解决问题，做一个合格且对社会有益的北大毕业生。

感谢我的父母，他们的付出永远不求回报。

最后非常感谢评审的各位专家，感谢能够不吝惜宝贵的时间，在百忙之中审阅本文。