目　录

[第1章 绪　论 4](#_Toc261339907)

[1.1　基于电子银行的个人财务规划平台概述 4](#_Toc261339908)

[1.2　市场前景与预期 4](#_Toc261339909)

[1.3　系统功能概述 5](#_Toc261339910)

[第2章　系统分析及技术选型 6](#_Toc261339911)

[2.1　系统分析 6](#_Toc261339912)

[2.1.1　限定问题 6](#_Toc261339913)

[2.1.2　确定目标 6](#_Toc261339914)

[2.1.3　调查研究，收集数据 6](#_Toc261339915)

[2.1.4　可行性分析 7](#_Toc261339916)

[2.1.5　提出备选方案和评价标准 7](#_Toc261339917)

[2.1.6　提交最可行方案 7](#_Toc261339918)

[2.1.7　解决方案 7](#_Toc261339919)

[2.2　技术选型 7](#_Toc261339920)

[2.2.1　Flex 7](#_Toc261339921)

[2.2.2　DataPush 8](#_Toc261339922)

[2.2.3　Cairngorm（MVC） 8](#_Toc261339923)

[2.2.4　BlazeDS 8](#_Toc261339924)

[2.2.5　Java 8](#_Toc261339925)

[2.2.6　Spring 9](#_Toc261339926)

[2.2.7　iBatis 9](#_Toc261339927)

[2.2.8　C3P0 9](#_Toc261339928)

[2.2.9　MySQL/Sybase 10](#_Toc261339929)

[2.2.10　SVN（subversion） 10](#_Toc261339930)

[2.2.11　Google Code 10](#_Toc261339931)

[第3章　需求分析 11](#_Toc261339932)

[3.1　对系统的综合需求 11](#_Toc261339933)

[3.1.1　功能需求 11](#_Toc261339934)

[3.1.2　性能需求： 11](#_Toc261339935)

[3.2　分析系统的数据要求 12](#_Toc261339936)

[3.2.1　银行系统信息安全数据： 12](#_Toc261339937)

[3.2.2　股指、汇率等频繁数据存储： 12](#_Toc261339938)

[3.2.3　规划方案数据存储： 12](#_Toc261339939)

[3.3　系统逻辑模型 12](#_Toc261339940)

[第4章　系统设计 14](#_Toc261339941)

[4.1　概要设计 14](#_Toc261339942)

[4.1.1　系统部署组织构建设计 14](#_Toc261339943)

[4.1.2　开发环境 15](#_Toc261339944)

[4.2　详细设计 15](#_Toc261339945)

[4.2.1　前端Flex页面显示设计 15](#_Toc261339946)

[4.2.2　业务中间件设计 16](#_Toc261339947)

[4.2.3　后端逻辑设计 17](#_Toc261339948)

[4.2.4　持久层设计 18](#_Toc261339949)

[第5章　编码实现 23](#_Toc261339950)

[5.1　Flex端的编码实现 23](#_Toc261339951)

[5.1.1　动态加载CSS样式 23](#_Toc261339952)

[5.1.2　Flex端模型层单利模式实现 23](#_Toc261339953)

[5.1.3　配置Cairngorm加载实现 24](#_Toc261339954)

[5.1.4　Flex配置业务信息服务中间件实现 25](#_Toc261339955)

[5.2　业务信息中间件配置方法 25](#_Toc261339956)

[5.2.1　BlazeDS配置 25](#_Toc261339957)

[5.3　Java端业务编码实现 26](#_Toc261339958)

[5.3.1　Spring 3.0.2搭建 26](#_Toc261339959)

[5.3.2　iBatis 3搭建 27](#_Toc261339960)

[5.4　数据推送实现 28](#_Toc261339961)

[5.5　邮件通知实现 28](#_Toc261339962)

[第6章　测试、运行及维护 29](#_Toc261339963)

[6.1　单元测试 29](#_Toc261339964)

[6.1.1　登录模块测试 29](#_Toc261339965)

[6.1.2　个人理财模块的测试 29](#_Toc261339966)

[6.1.3　数据推送模块测试 30](#_Toc261339967)

[6.2　集成测试 30](#_Toc261339968)

[6.3　系统测试 30](#_Toc261339969)

[6.3.1　浏览器测试 30](#_Toc261339970)

[6.3.2　系统运行测试 30](#_Toc261339971)

1. 绪　论
   1. 基于电子银行的个人财务规划平台概述

花旗个人理财系统，是作为2009年花旗银行举办的“花旗杯”金融与信息技术应用大赛获得第22名的参赛作品，由我带领的团队完成的整套系统，它是一套基于电子银行WEB模式的针对于个人财务的规划平台。本套系统主要致力于电子银行的服务质量以及产品价值，为广大用户提供针对于个人的一套较为完整的理财规划方案以及实施的一套理念。本项目定位为基于电子银行理财服务领域职能，利用RIA（Rich Internet Application富网络应用）模式技术来实现系统个人财务规划流程。以人机对话的方式，实现电子银行客户自助的理财分析，并且通过对客户理财目标的规划，与电子银行综合，完善电子银行的整套体系。

这套系统是作为电子银行的一个子模块的形式来设计并实现的。

* 1. 市场前景与预期

对于广大商业银行来说，如今的电子银行系统正在经历业务支付电子化到服务电子化的转型。而在中国理财领域，由于对客户需要比产品推销更重要的认同，理财规划服务正在兴起。然而，一方面，理财分析师的短缺，另一方面，银行通常针对百万千万资产的大客户开放个人对个人的理财规划，这样理财对于普通平民用户来说又是高额成本，导致了中国的理财市场仍然处在重视产品设计，产品营销，而忽视基于客户实际情况的理财分析，实际上导致了“个人理财”的“非个人化”。

本产品旨在解决个人理财个人化，做到个人理财程序化。现在中国商业银行的理财服务主要是针对大客户推行，门槛较高，很多工薪阶层的百姓因为没有能力一次性拿出十万元以上来做专项的理财基金，而无法成为银行服务的对象，但这部分人群对于专业的财富规划又有着强烈的需求。而且即使对于很多中高收入的人群来说，能享受到的理财服务也仅限于银行对理财产品的推荐，尚无法真正做到为个人制定一套行之有效的理财规划。另一方面，对于银行来说，集中存款的要求使他们需要工薪阶层积沙成塔的存款，但是由于专业理财师的稀缺，也使他们无力为这么大的人群提供服务。因此这个程序化理财产品，将更多的业务交给标准化程序来做，减轻银行工作量的同时帮他们吸引存款，也为更多的百姓服务。通过了解用户的资金状况和生活情况，在适当预期的基础上，为他们订立一套终身的理财规划建议，使他们切实得到专业理财建议，并且在日常通过对用户特殊账户的监控，了解其资金变动情况，随时做出调整。

我们正是把握着电子银行所经历变革，结合现如今国内理财市场的发展现状，将理财规划分析纳入电子银行的服务领域，力图实现客户低成本地，自助式地采用人机对话的方式，在网络平台上实现个人财务规划。并进一步结合挖掘电子银行的网上支付功能，使电子银行系统真正成为一个包含个人财务分析，理财目标规划，理财目标执行的一体的电子商务化理财平台。

完善电子银行的服务体系。使客户享受到集合个人财务状况分析，理财目标规划，理财规划执行，理财目标实施情况跟踪反馈为一体的电子银行理财服务。

实现低成本，电子商务化的个人财务规划，建立面向大众的，个人财务规划平台。

* 1. 系统功能概述
* 股指、汇率动态显示
* 金融新闻
* 个人财务现状分析
* 个人理财目标规划
* 利用网上支付实现个人理财目标规划的执行
* 监控理财目标实施情况

（具体请参见[第二章　系统分析章节](#系统分析)，本系统仅选部分功能实现）

1. 系统分析及技术选型
   1. 系统分析
      1. 限定问题

对于现实情况来说，国内暂时还没有任何一套服务是由银行针对于个人用户提供的理财规划，同时也没有相关的资料来进行借鉴，这样对于理想状态下的系统实现，提高很大的难度和要求，目前的所谓理财，都是由银行方的理财分析师针对银行的大客户，资产超过几百万千万的客户进行一对一的理财规划，所以我们需要做好充分的准备和资料来制定以及完善这套人机模式的理财系统。

另外的一点就是针对实时的数据变更，银行的股票指数、汇率等数值时时刻刻的变化，带来的技术问题，值得我们的注意。

* + 1. 确定目标

针对于系统的功能来看，制定了所需实现的功能目标以及预期达到的技术目标如下：

* 动态实时显示股指、汇率等即时数据
* 个人投资风险偏好测试
* 根据客户信息，智能计算个人理财规划方案
* 对理财进行监督以及帮助实现。
  + 1. 调查研究，收集数据

国内个人投资市场随着中国政府有关金融政策的相继放宽，社会保险制度的不断健全，社会保障基金的入市，政府基建项目私人投资者市场的最终开放，中国政府开始打破已往面对私人投资者的政策壁垒和相应的金融管制，已经全面开启投资的大门，向社会多渠道融集资金，资本多元化日益成为其鲜明特点，个人投资范围越来越广泛。证券，外汇，黄金，保险，以及衍生出的金融工具种类逐步丰富，为投资者提供了丰富的理财产品。并且随着金融市场监管体制的逐步完善，投资环境也相对平稳，为个人财富投资的稳中增值创造了条件，为“个人理财”创造了良好的外部环境。

而伴随着居民生活水平的日益提高，以及“理财”意识的深入人心。购房换屋，成为“有车一族”等需求需要满足带来了大量的理财需求，并且子为保证子女学业顺利完成走向社会，以及退休养老后的生活水平的保障，也要求人们去对当前财务现状做出分析，制定理财目标与规划，并通过储蓄，投资等手段达到目标。

* + 1. 可行性分析

对于我们，则需要制定出一套非常精确准确的体系方案，理财方法与合理的公式，来为广大用户提供服务，同时也可给银行带来一定的影响力和宣传作用，更会有直接的收益获取在这当中。

对于技术来说，我们则选择一种较新的模式，以及较新的技术来给大家提供更有没的界面，更舒适的用户体验，更灵活更健壮的技术来实现我们的产品，而Flex技术独有的数据推送，以及与Java端的完美结合，使得系统的可行性更加有说服力。

* + 1. 提出备选方案和评价标准

对于技术难点来说，我们在使用新技术上会有一定的风险，毕竟对于此种行业来说，还是“史无前例”的，而我们则会采用Ajax技术来作为备选方案。其特点和Flex技术类似，Ajax也实现了页面的动态，动感以及数据的无刷新等特点，不过Ajax在实质上并非是一种动态无刷新技术，只是通过前端计算来模拟实现，所以，我们考虑使用Ajax来作为备选方案。

* + 1. 提交最可行方案

我们最终确定通过使用Flex作为前段通信技术、通过Java后端成熟的框架体系来实现整套符合理财规则的针对于个人客户的理财规划平台，并以Ajax技术作为备选技术。

* + 1. 解决方案

相关解决方案在[下一节　技术选型](#技术选型)中一一分析。

* 1. 技术选型
     1. Flex

技术简介：Flex最初是由Macromedia公司在2004年3月发布的，基于其专有的Macromedia Flash平台，它是涵盖了支持RIA（Rich Internet Applications）的开发和部署的一系列技术组合。使用的是ActionScript和封装好的标签式语言。

使用原因：Flex是RIA模式领头技术，可以动态更新数据、友好的用户体验、对任何支持FlashPlayer的操作系统和任何浏览器都适用并且不会出现格式错位解析异常等问题。另外与Flex配套的还有本地的桌面RIA应用，也就是AIR，这是本地化的云项目，丰富了云模式的本地化处理。当然还有就是下面提到的数据推送技术。

* + 1. DataPush

技术简介：数据推送技术是一种新型概念，是一种新的思想，是以他更动态的方式来实现直接从后端像页面前段推送数据，而无需由页面前段发出请求来获取数据，这样则实现更加动态更加灵活的功能。

使用原因：数据推送技术虽然现在并没有成熟，但是他非常好的解决了由服务器端发生数据变化而让客户端数据也改变的问题，这也是Flex的独特技术，强大的服务器端处理并推送数据。

* + 1. Cairngorm（MVC）

技术简介：Cairngorm是Adobe为Flex技术推出的一套基于MVC模式的轻量级框架技术，从而使程序可扩展性、可维护性都大大提高，其本身并不是一个完整的企业应用，它只是提供了一个开发骨架，Adobe称之为体系。Cairngorm主要就是对开发Flex应用程序应用了一系列的设计模式，从而使开发出来Flex程序可扩展性，可维护性都大大提高。他很好的将Flex的各个层面很好的进行了分层化。

使用原因：当然，我们就是为了能够在每一个层次划分的更明确一些，耦合更少一些。

* + 1. BlazeDS

技术简介：BlazeDS是一个基于服务器的Java远程控制(remoting)和Web消息传递(messaging)技术，以LGPL（Lesser GNU Public License）公共许可证书发布。它能够使得后端的Java应用程序和运行在浏览器上的Adobe Flex应用程序相互通信。在Java应用服务器上，它以servlet的形式存在，因此可以在任何标准Java网络应用中运用它。

使用原因：Adobe提供了更为强大的LCDS（LiveCycle Data Service），而BlazeDS为简装的LCDS，他提供了很多基本的功能，我们的Flex和Java端连接就靠他了，他提供了与Spring等业务框架的连接方式以及数据推送的方法和类库等。

* + 1. Java

技术简介：Java自然是不用多说了，他拥有跨平台、动态的Web、Internet计算、以及开源等很多的有点，而他的开源方式，也让全世界的开发者来钻研，而产生了上百余种的应用型框架和服务等等。

使用原因：自然也不用多说了，他强大的松耦合框架，轻量级的技术，以及非常完善的业务逻辑设计，给开发带来了很多优势，使整个系统更加优美，更加健壮。

* + 1. Spring

技术简介：Spring是一个开源框架，它是为了解决企业应用开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。他可以理解为一个轻量级的IOC（控制反转）和AOP（面向切面）容器框架。本他系统则使用了刚刚正式发布的Spring3.0.2版本。

使用原因：我们使用它来合理的来创建了需要时所应用的对象和切面管理一定事件时的事物。

* + 1. iBatis

技术简介：是一个基于Java的轻量级持久性框架，相对Hibernate和Apache OJB等“一站式”ORM解决方案而言，iBatis 是一种“半自动化”的ORM实现。本套系统则使用了还没有正式发布的iBatis3版本。

使用原因：相对于Hibernate来说，iBatis和Hibernate各有各自的优点。对于本套系统，由于是一套银行系统，所以对数据库的操作会相对频繁，尤其是在汇率和股指处理的时候，都是毫秒级的操纵数据库。而Hibernate的优势是在于灵活的SQL语句，可以动态的配置数据库，但是对于银行的业务需求，较高的要求应该由我们自己的SQL来实现和优化。还有我们的个人理财系统项目是作为网上电子银行的子系统出现的，而这并不适用于Hibernate持久型框架。经过很多的综合因素，我们选择了iBatis持久型轻量级框架。

* + 1. C3P0

技术简介：C3P0是一个开放源代码的JDBC连接池，它在lib目录中与Hibernate一起发布,包括了实现jdbc3和jdbc2扩展规范说明的Connection 和Statement 池的DataSources 对象，连接池对于系统来说持有了连接数据库的连接，节省了系统开销，节省了连接，更好的管理连接，有效的减少了频繁连接数据库操作。

使用原因：虽然说大多数的时候C3P0数据库连接池都是和Hibernate搭配适用的，不过我们也可以与iBatis持久层框架一起适用，而C3P0的配置具有较为完善的连接池管理配置和更为庞大的管理机制，适用于较大型，要求较高的系统适用。

* + 1. MySQL/Sybase

技术简介：MySQL和Sybase都是非常实用的数据库管理系统。

MySQL名字的来历MySQL是一个小型关系型数据库管理系统，开发者为瑞典MySQLAB公司，在2008年1月16号被Sun公司收购。MySQL被广泛地应用在Internet上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了MySQL作为网站数据库。

美国Sybase公司研制的一种关系型数据库系统，是一种典型的UNIX或WindowsNT平台上客户机/服务器环境下的大型数据库系统。 Sybase提供了一套应用程序编程接口和库，可以与非Sybase数据源及服务器集成，允许在多个数据库之间复制数据，适于创建多层应用。系统具有完备的触发器、存储过程、规则以及完整性定义，支持优化查询，具有较好的数据安全性。Sybase通常与SybaseSQLAnywhere用于客户机/服务器环境，前者作为服务器数据库，后者为客户机数据库，采用该公司研制的PowerBuilder为开发工具，在我国大中型系统中具有广泛的应用。

使用原因：按正常的银行系统来说应该适用Sybase的DBMS，但是我们系统着重的强调了业务逻辑和系统架构以及编码设计，所以我们的DBMS只是当作了一个数据存储和管理的工具，并没有太重视数据库部分。

* + 1. SVN（subversion）

技术简介：SVN(subversion)是近年来崛起的版本管理工具，是cvs的接班人。目前，绝大多数开源软件都使用svn作为代码版本管理软件。

使用原因：使用SVN来进行代码的管理，方便了代码的开发、对比和版本管理等。

* + 1. Google Code

技术简介：Google Code 是 Google 的网站开发人员工具和 api 的技术资源。该站点包含在使用 Google 的开发人员工具和 api 的使用 Google 的开发人员产品的开发人员包括讨论组和博客的文档。

使用原因：使用Google Code 作为远程的代码存放服务，方便代码的管理，他支持SVN以及其他的一些版本管理工具，给广大开源开发者提供了很多方便。

1. 需求分析
   1. 对系统的综合需求
      1. 功能需求
         1. 关于银行用户的登录：

银行用户登录，就是用户通过个人的银行卡号或者密码对用户的身份进行验证，从而进入本电子银行系统并执行相应的功能操作，并且需要对个人信息进行保密，详情稍后介绍。

* + - 1. 关于个人投资风险偏好测试：

在每个人的理财规划之前，系统将对每个人对所谓的“花钱”做一个较为详细的测试，来了解使用者对平常的财务使用情况或者说投资风险的承受能力是如何的，然后系统会根据相应的风险类型来对用户做进一步的理财规划，从而达到较为合理的理财规划。

* + - 1. 关于理财规划模式：

本系统是以个人理财为目的，我们的系统大体分为以下几种理财模式，有“个人购房理财规划”、“个人购车理财规划”、“子女教育财务规划”、“养老财务规划”和“寿险财务规划”五种模式，本系统实现仅以“个人购房理财规划”为例。

* + - 1. 关于规划执行阶段：

系统需求的执行方法是采用通过邮件的通知方式来督促用户来完成规划出来的这份理财方案，解决方案是采用通过存入数据库的预期规划方案，每日的凌晨用户访问量少的时候，随系统服务启动的线程服务来遍历查询数据库的数据，若有与当前日期相同的日期，则执行发送邮件通知功能。

* + - 1. 算法分析：

作为一套个人理财系统，算法起到了很大的作用

* + 1. 性能需求：

对于RIA模式的系统来说，本身的体系模式就是将由原来负责数据处理的服务器端的繁重任务转移到现在的客户端来进行计算，相当于新的一种“云”概念，但也不完全是“云”计算，我们是通过一些在本地的计算来提高系统的运行速度。

当然Flex也有他自己的一些性能方面的需求，本事Flash的运行体系就是要讲整个系统先下载到本地，这样如果Flex编译出来的swf体积过大也会影响系统的加载速度和运行速度，所以我们尽量的来优化Flex端的代码设计，使得Flex编码优化，大大减小内存开销和编译出的swf文件体积。

* 1. 分析系统的数据要求
     1. 银行系统信息安全数据：

作为银行系统，与财务有关，银行就应该为客户的个人财产等信息进行保密，乃至银行的工作人员也无法知道用户的一些个人信息，如密码等。我们的解决方案则是采用直接通过前端Flex技术在页面就进行加密，通过Sha1算法对密码的字符串进行运算加密，然后通过BlazeDS传递给Java端，并存入数据库中。

* + 1. 股指、汇率等频繁数据存储：

由于股指和汇率不是由银行来控制，并且在现实生活中，这些随时变动的数值是不定期，有可能是在一秒钟之内由某官方组织发送若干数据，也有可能几秒钟都不发生变化，而我们作为系统操作端，需要使用这些数据，并且需要存储，我们则采用单独设计一张表来存储这些随即变动的大量数据。如果需要，备用方案是单独的来分离数据库来存储这些大批量的数据。

* + 1. 规划方案数据存储：

我们的系统最终是要给用户一套较为合理的理财规划方案，而我们将采用存储有效的数据来进行存储，而在预定预期的时段来给用户一定的指导和帮助措施。

* 1. 系统逻辑模型

整套系统的逻辑模型如图 3‑1　个人理财系统逻辑顺序图。

这个图将会很好的表述我们这套系统的整个的业务流程的顺序。



图 3‑1　个人理财系统逻辑顺序图

1. 系统设计
   1. 概要设计
      1. 系统部署组织构建设计

图 4‑1　个人理财系统架构图，描述了整套系统的系统架构和流程模式。



图 4‑1　个人理财系统架构图

* + 1. 开发环境

系统开发IDE：Flex Builder3.0.2 plugin on Eclipse3.4.2；

Flex端开发SDK：Flex SDK 3.5、Datavisualization\_sdk3.5；

Java端开发SDK：JDK6 Update 20；

数据库：MySQL 5.1.36\_X64；

系统服务器中间件：Apache Tomcat 6.0.24；

系统业务中间件BlazeDS 3.2；

后台业务逻辑应用框架：Spring3.0.2；

后台数据持久层引用框架：iBatis 3 Beta 10；

开发环境操作系统：Windows 7 Enterprise X64；

Flex端框架：Cairngorm 2.2.1；

Java端业务逻辑层框架：Spring 3.0.2；

Java端持久层框架：iBatis 3 Beta 10；

数据连接池框架：C3P0 0.9.1；

* 1. 详细设计
     1. 前端Flex页面显示设计
        1. 页面引用功能设计

采用标准的模块独立划分式，让页面之间的耦合尽量的减少，我们采用的是以一个mxml文件管一个子模块活页面，通过SDK拥有的Flex组件和我们自己定义出来的Components自定义组件来实现系统的界面部分。

* + - 1. 页面布局设计

我们的页面布局设计则使用广大网站标准的布局样式，当然我们也会发挥出我们Flex特有的一些特色东西。

布局采用上面的应用条和下面的主界面布局样式，应用条左侧使用Logo显示，中间则是简易的股指显示LineChart图，右侧显示系统时间等。主页面布局则是左面是导航部分，左下显示当前打开窗口列表，右面是主页面显示部分。

* + - 1. 页面加载方式设计

为了实现动态的加载页面，那么我采用了通过Flex的反射机制，其实也就是ActionScript3.0的反射技术来实现页面的加载，将若干个主要页面的文件配置到xml文件当中，然后通过Flex来动态的加载页面，由此实现。

* + - 1. 股指汇率显示设计

首先，股指汇率等大批量数据在数据库中存储，然后在系统加载的时候，通过RPC（远程过程调用）来调用Java端的执行方法，查询出数据库中的最后若干条数据用来页面的初始化数值显示，然后通过Flex强大的数据推送技术来实现由服务器端主动向客户端推送股指数据动态的在用户页面上显示，如图 4‑2　股指显示LineChart式样图所示。

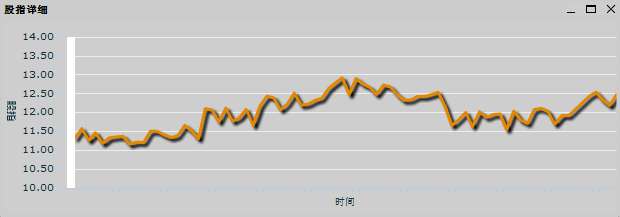


图 4‑2　股指显示LineChart式样图

* + - 1. 前端样式实现设计

Flex这技术对“灵活”和“动态”表现的都非常的好，充分的体现了他的页面无刷新简易加载等RIA网络模式的特性。

我们的前端样式实现，采用动态加载CSS样式表，但是正常的CSS文件只是一种纯文本文件，Flex的SDK就提供了将CSS进行编译成为swf文件，然后通过Flex程序可以动态的加载我们的样式表，并且可以在不刷新页面的情况下对样式进行更换加载等。

* + 1. 业务中间件设计
       1. 数据推送模块设计

实际上的数据是由某官方组织来发布的，我们作为使用者也作为提供者，所以我们要先接受数据，然后再转发，那么我们采用的方式就是模拟组织来进行数据发送，然后我们将推送的部分单独的卸载一个Servlet里面，加载再Web的项目的web.xml配置文件当中，由人为来通过指令来控制数据是否进行推送或者是否进行停止操作。

* + - 1. Flex端模型层设计

对于Flex的对象使用来说，模型层是Flex端的重点，因为Flex注重的是客户端的数据处理，而客户端的数据都是在模型层当中，所以我们尽量的采用单利的模式来使内存中只有一个模型对象，从而节省内存的开销和灵活的使用数据源，具体如图 4‑3　Flex端模型层设计类图所示。



图 4‑3　Flex端模型层设计类图

* + - 1. Flex端总体架构设计

我们采用的是Adobe官方的基于MVC模式的Cairngorm框架来实现，采用的是2.2.1版本。当然针对于系统的应用，还是对Cairngorm框架进行了一定的优化和修改，我将原本的操作类\*Command.as和通过RPC调用的\*Delegate.as合并在了一起。并且使用了继承方式使所有的\*Command都共同拥有连接Java端的RemoteObject对象持有Service。

* + - 1. Flex与Java端连接设计

采用了Adobe提供的与Java端连接的BlazeDS这项业务中间件技术，BlazeDS提供了数据推送服务以及与Spring业务层连接等一系列的功能。

* + 1. 后端逻辑设计
       1. Java端业务逻辑

Java端的业务逻辑层实现则是采用Spring轻量级开源框架，我们采用的版本是Spring3.0.2，是最新发布的Spring版本，与之前的2.5.x有了很大很大的区别，首先从Jar包上，Spring去掉了原来拥有的Spring.jar总需要Spring的jar包，而是分离开来，我们选用Spring来控制Flex端对Java端调用时所需要的一些类、对象等等，通过Spring的工厂来动态的灵活创建对象。

* + - 1. Java端持久层分离

Dao实现设计，使得我们在更换持久层操作变得更加灵活和简单，只需实现提供的接口即可，具体如图 4‑4　Java端Dao也Service设计类图所示。



图 4‑4　Java端Dao也Service设计类图

* + 1. 持久层设计
       1. Java端持久层实现

我们的持久层选择通过iBatis轻量级持久层框架来实现，版本我们选择了Apache还没有正式发布的iBatis3的测试版iBatis3Beta10版本。

iBatis3在总体的应用上与之前的iBatis2有了很大的区别，比如在xml配置文件的配置上，执行语句的调用上等，而目前，由于iBatis3还没有正式发布，所以Spring还并没有支持iBatis3，我们将使用自己写的SqlSessionTemplate来实现数据库的JDBC操作。

* + - 1. 数据库设计

首先，我们的数据库设计并非是真正的银行数据表结构，我们也不清楚庞大的银行数据表系统是怎样的结构，所以我们以下的数据库设计只是针对本理财系统的需要来设计的一些表和相关字段。

其实对于数据库设计来说，又很多相关注意的事项，比如说在一个功能复杂的系统上，要根据实际的需求和数据的存储量来设计表与表之间的关系等，具体归纳为一下：

* 根据数据操作是否频繁分表；
* 根据数据与数据的对应关系分表；
* 根据范式理论对依赖和主外键分表；
* 根据需求来设计适当的数据类型和存储空间长度；
* 根据相应需求来设计表字段的非空、唯一等约束；
* 根据数据特性来设计是否使用索引等；
* 根据业务需求合理设计表名字段名。

|  |  |
| --- | --- |
| CITI\_RateData | |
| RD\_ID | INT 非空 自增 主键 非负 |
| RD\_CODE | CHAR(2) 非空 |
| RD\_StockIndex | DOUBLE 非空 |
| RD\_StockTime | DATETIME 非空 |

表 4‑1　银行股指汇率数据表

表 4‑1　银行股指汇率数据表是用于存储每时每刻不间断的股指汇率数据的表。

RD\_ID字段表示该表每条数据的ID；

RD\_CODE字段表示银行某种股指或者货币对汇率的类型嘛，如01纳斯达克、02道琼斯等；

RD\_StockIndex字段表示股指数值；

RD\_StockTime字段表示股指的发送时间。

|  |  |
| --- | --- |
| CITI\_StanData | |
| SD\_ID | INT 非空 自增 主键 非负 |
| sd\_depo\_Rate | DOUBLE 非空 |
| sd\_CRED\_RATE | DOUBLE 非空 |
| sd\_prod\_Rate | DOUBLE 非空 |
| sd\_prod\_Dline | DOUBLE 非空 |
| sd\_mort\_Dline | DOUBLE 非空 |
| sd\_mort\_Rate | DOUBLE 非空 |

表 4‑2　银行基本信息表

表 4‑2　银行基本信息表是用于存储银行应用的一些基本信息，本表的设计只是为了模拟虚拟银行，并不是真实性的。

SD\_ID字段表示该表的主键ID；

SD\_DEPO\_RATE字段表示银行的存款利率；

SD\_CRED\_RATE字段表示银行的贷款利率；

SD\_PROD\_RATE字段表示银行的公积金利率；

SD\_PROD\_DLINE字段表示银行的公积金期限；

SD\_MORT\_RATE字段表示银行的按揭利率；

SD\_MORT\_DLINE字段表示银行的按揭年限。

|  |  |
| --- | --- |
| CITI\_USERINFO | |
| UI\_INFOID | INT 非空 自增 非负 索引 唯一 主键 |
| UI\_BANKID | CHAR(16) 非空 唯一 |
| UI\_NAME | CHAR(15) 非空 |
| UI\_ID | CHAR(18) 非空 |
| UI\_PASSW | CHAR(128) 非空 |

表 4‑3　个人账户信息表

表 4‑3　个人账户信息表是记录银行的账户的一些个人基本信息。

UI\_INFOID字段表示该个人信息表的主键，也为其他依赖表提供引用服务；

UI\_BANKID字段表示该用户的银行账号；

UI\_NAME字段表示用户的姓名；

UI\_ID字段表示用户的身份证号；

UI\_PASSW字段表示用户的账户密码；

|  |  |
| --- | --- |
| CITI\_PURC\_ACCOUNT | |
| PA\_INFOID | INT 非空 非负 索引 唯一 主键 外间 |
| PA\_PURC\_FUND | DOUBLE 非空 非负 |

表 4‑4　购房账户表

表 4‑4　购房账户表购房账户表是对银行用户的个人购房账户分配的表。

PA\_INFOID字段表示个人银行的账户表ID，属于引用个人账户表的外键；

PA\_PURC\_FUND字段表示用户的购房账户里面的资金。

|  |  |
| --- | --- |
| CITI\_USERINFO\_ADD | |
| UA\_INFOID | INT 非空 非负 索引 唯一 主键 外间 |
| UA\_AVG\_MICOM | DOUBLE 非空 非负 |
| UA\_AVG\_MPOUT | DOUBLE 非空 非负 |

表 4‑5　个人信息附加表

表 4‑5　个人信息附加表是根据账户需求来分配的个人的附加信息表。

UA\_INFOID字段表示个人银行的账户表ID，属于引用个人账户表的外键；

UA\_AVG\_MICOM字段表示个人月平均收入；

UA\_AVG\_MPOUT字段表示个人月平均支持。

|  |  |
| --- | --- |
| CITI\_ACCOUNT\_FUNDS | |
| AF\_INFOID | INT 非空 非负 索引 唯一 主键 外间 |
| AF\_FUNDS | DOUBLE 非空 非负 |

表 4‑6　账户资金表

表 4‑6　账户资金表是用户的银行账户中的存储金额的表。

AF\_INFOID字段表示个人银行的账户表ID，属于引用个人账户表的外键；

AF\_FUNDS字段表示用户的当前账户资金额度。

|  |  |
| --- | --- |
| CITI\_PURC\_PROG | |
| PP\_INFOID | INT 非空 非负 索引 唯一 主键 外间 |
| PP\_ID | TINYINT 非空 非负 唯一 |
| PP\_DPAY | INT 非空 |
| PP\_MFUND | DOUBLE 非空 |
| PP\_MCOUNT | TINYINT 非空 |
| PP\_PURC\_DATE | DATE 非空 |
| PP\_SQUARYM | INT 非空 |
| PP\_MPRICE | DOUBLE 非空 |
| PP\_FUNDS | DOUBLE 非空 |

表 4‑7　购房规划推荐表

表 4‑7　购房规划推荐表是系统做出了个人的购房规划推荐之后将数据存入到数据库中的表。

PP\_INFOID字段表示个人银行的账户表ID，属于引用个人账户表的外键；

PP\_ID字段表示该推荐计划的ID号；

PP\_DPAY字段表示购房规划首付金额；

PP\_MFUND字段表示购房计划每月要存入的资金数；

PP\_MCOUNT字段表示购买房子共要还的总共月数；

PP\_PURC\_DATE字段表示预计的购房时间；

PP\_SQUARYM字段表示预计购房的平米数；

PP\_MPRICE字段表示购房预计每平米价格；

PP\_FUNDS表示表示购房预计总额度。

|  |  |
| --- | --- |
| CITI\_INVE\_PREF | |
| IP\_INFOID | INT 非空 非负 索引 唯一 主键 外间 |
| IP\_PREF | TINYINT非空 |

表 4‑8　个人投资偏好表

表 4‑8　个人投资偏好表是用来存储用户个人的理财偏好类型。

IP\_INFOID字段表示个人银行的账户表ID，属于引用个人账户表的外键；

IP\_PREF字段表示用户的风险偏好投资类型；

|  |  |
| --- | --- |
| CITI\_USER\_NOTE | |
| UN\_INFOID | INT 非空 非负 索引 唯一 主键 外间 |
| UN\_NOTE | BLOB 非空 唯一 |
| UN\_UPDATADATE | DATE 非空 |
| UN\_DATE | DATE 非空 |

表 4‑9　个人记事表

表 4‑9　个人记事表是用来存储个人用户的一些临时记事数据表。

UN\_INFO字段表示个人银行的账户表ID，属于引用个人账户表的外键；

UN\_NOTE字段表示临时存储的记事内容；

UN\_UPDATADATE字段表示插入表的时间；

UN\_DATE字段表示记事的内容时间；

根据这些字段的关系创建了如上这些表。

* + - 1. SQL语句设计

插入股指数据向表 4‑1　银行股指汇率数据表SQL语句

insert into

Citi\_Ratedata(RD\_STOCKINDEX,RD\_STOCKDATE)

values

(#{stockIndex},#{stockTime});

查询最后若干条股指数据SQL语句

select \*

from (

select RD\_STOCKINDEX,RD\_STOCKDATE

from CITI\_RATEDATA

order by

RD\_STOCKDATE

desc limit #{1}

) as stockOrdertemp

order by

RD\_STOCKDATE;

1. 编码实现
   1. Flex端的编码实现
      1. 动态加载CSS样式

在Flex程序加载的时候就去执行加载CSS编译好的swf文件，具体实现是在mx:Application的标签里面加入initialize="init()"，这个意思就是初始化事件触发时调用init()方法，init()方法如下：

private function init() : void

{

StyleManager.loadStyleDeclarations(viewConfig.cssfiles.cssfile[0]);

}

而其中的viewConfig其实是程序读取XML配置文件的id，读取配置文件实现如下：

<mx:Model id="viewConfig" source="configs/viewsConfig.xml" />

XML配置文件如下（仅举其中一例）：

<?xml version="1.0" encoding="UTF-8"?>

<configs>

<!-- 视图反射显示配置 -->

<views>

<label>风险偏好测试</label>

<className>com.citipf.liyunpeng.views.iPrefTest.IPrefTestMain</className>

</views>

</configs>

* + 1. Flex端模型层单利模式实现

使用单利模式来实现我们的模型层，节省内存的开销并且作为整套前段Flex的数据存储空间，实现如下：

package com.citipf.liyunpeng.models

{

import flash.errors.IllegalOperationError;

[Bindable]

public class Model extends StockModel

{

private static var instance : Model = new Model();

/\*\*

\* 单例类构造方法

\* @param cons 参数为私有类

\*/

public function Model()

{

if(instance != null) {

throw new IllegalOperationError(

"This is a singleton class,you can call the getInstall() method!" +

"--这是一个单例类,您可以调用getInstance方法获取实例！");

}

}

/\*\*

\* @return 返回单例实现

\*/

public static function getInstance() : Model{ return instance ; }

}

}

* + 1. 配置Cairngorm加载实现

Cairngorm的服务和控制层需要在RIA程序加载的时候就加载运行，所以我们需要配置，具体标签配置如下：

<!-- 服务 -->

<business:CitiPFServices />

<!-- 控制层 -->

<controls:CitiPFControls />

<!-- 主页面 -->

<views:MainPage />

其中的business、controls和views是通过配置命名空间来实现调用的，命名空间的实现如：

<mx:Application xmlns:views="com.citipf.liyunpeng.views.\*" />

* + 1. Flex配置业务信息服务中间件实现

我们的Flex是通过RemoteObject方式来连接业务中间件服务的，当然我们还可以通过WebService、HttpService或者其他服务等来实现后台。RemoteObject配置如下：

<mx:RemoteObject id="citiPFService" destination="citiPFService" />

* 1. 业务信息中间件配置方法
     1. BlazeDS配置

其实BlazeDS的实现是他给了一个关于各种服务的类库和执行文件，然后配置到Web.xml文件当中，然后Web程序的服务在启动的时候就自动的加载了，然后BlazeDS的服务就这样的被实现了，那么我们的BlazeDS在web.xml中的配置如下：

<servlet>

<servlet-name>MessageBrokerServlet</servlet-name>

<servlet-class>flex.messaging.MessageBrokerServlet</servlet-class>

<init-param>

<param-name>services.configuration.file</param-name>

<param-value>/WEB-INF/flex/services-config.xml</param-value>

</init-param>

<init-param>

<param-name>flex.write.path</param-name>

<param-value>/WEB-INF/flex</param-value>

</init-param>

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

</servlet>

这段代码其中加载了配置文件services-config.xml文件，而service-config.xml配置了关于信息服务相关的内容，配置如下（仅举例说明）：

<channel-definition id="my-amf" class="mx.messaging.channels.AMFChannel">

<endpoint url="http://{server.name}:{server.port}/{context.root}/messagebroker/amf" class="flex.messaging.endpoints.AMFEndpoint"/>

</channel-definition>

而在services-config.xml里面也配置了一个remoting-config.xml文件，他的作用就是对Flex端的remoteObject对象和Java端的对象进行连接的。remoting-config.xml具体配置如下：

<default-channels>

<channel ref="my-amf"/>

</default-channels>

<destination id="citiPFService">

<properties>

<factory>springFactory</factory>

<source>citiPFService</source> <!--此处注入IOC-->

</properties>

</destination>

其中的citiPFService则是在Spring中注入的Bean。

那么Spring的注入，其实也是在service-config.xml中的，代码如下：

<factories>

<factory id="springFactory" class="com.citipf.liyunpeng.factories.SpringFactory" />

</factories>

* 1. Java端业务编码实现
     1. Spring 3.0.2搭建
        1. Spring监听配置

首先，我们要在web.xml当中来配置Spring的监听，这样在我们整个的Web项目当中就可以直接使用Spring的依赖注入控制反转了。而我们数据推送的实现不是通过Web项目服务调用的，我们也有另外的方法来调用Spring，先说配置监听如下：

web.xml

<context-param>

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

<param-value>

classpath\*:applicationContext\*.xml

</param-value>

</context-param>

以上代码将Spring的配置文件统一随系统服务加载了进来。

* + - 1. 数据源和数据连接池配置

我们采用的是通过加载配置文件来动态的配置数据源，具体实现如下：

applicationContext-Dao.xml

<bean id="propertyConfigurer" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="locations">

<list>

<value>classpath\*:mainConfig.properties</value>

</list>

</property>

</bean>

以上代码是通过Spring自身提供的类，通过注入来加载配置文件mainConfig.properties的。

我们采用了C3P0数据库连接池来管理数据库的链接，使用的是C3P0 0.9.1版本的jar包，同样，我们也是通过Spring的注入将C3P0的数据源配置好，具体实现如下：

applicationContext-Dao.xml

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

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

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

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

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

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

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

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

</bean>

以上代码就是我们将数据源的属性通过C3P0的ComboPooledDataSource来配置的。

* + 1. iBatis 3搭建

iBatis3的配置在与之前iBatis2的版本相差了很多，其中包括OR的映射问题。

现在，因为Spring3暂时还没有支持iBatis3，所以，我们通过自身的配置来实现的。具体的Java端配置以由本文4.2.3（2）Java端持久层分离描述，那么我们本部分仅介绍iBatis对ORM的实现配置，具体如下（仅举例说明）：

applicationContext-Dao.xml

<bean id="sqlMapClient"

class="com.citipf.liyunpeng.dao.iBatis.SqlSessionFactoryBean">

<property name="configLocation" value="classpath:ibatisConfig.xml"></property>

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

</bean>

以上是将iBatis的配置文件通过Spring注入到iBatis的SqlSession创建工厂。

iBatisConfig.xml

<typeAliases>

<typeAlias alias="StockVO" type="com.citipf.liyunpeng.valueObject.StockVO"/>

</typeAliases>

<mappers>

<mapper resource="com/citipf/liyunpeng/valueObject/stockVO.xml"/>

</mappers>

以上是对iBatis3中具体值对象映射的配置，与iBatis2不同的是，typeAliases从原来的映射xml移到了配置文件中。

再举例说明一下具体映射文件的xml配置，如下：

stockVO.xml

<resultMap id="resultMap\_stock" type="StockVO" >

<result property="stockIndex" column="RD\_STOCKINDEX" javaType="Double"/>

<result property="stockTime" column="RD\_STOCKDATE" javaType="Date"/>

</resultMap>

<select id="selectListStock" resultType="list" resultMap="resultMap\_stock" parameterType="int">

select \* from (

select RD\_STOCKINDEX,RD\_STOCKDATE from CITI\_RATEDATA order by RD\_STOCKDATE desc limit #{1}

) as stockOrdertemp order by RD\_STOCKDATE;

</select>

其中对SQL语句参数的格式也有了变化，iBatis2中式两个#号中间来放参数的。

* 1. 数据推送实现

数据推送是Java利用BlazeDS提供的Jar文件，来实现的数据派发，具体实现应该是通过长轮训的方式对数据进行派发，实现代码如下：

msgBroker.routeMessageToService(msg,null);

其中的的msg是BlazeDS提供的AsyncMessage类型，将要发送的数据打包。

* 1. 邮件通知实现

邮件通知是通过Java的Timer计时器来实现的，我们在配置文件中设置什么时间查询数据库和每隔几个小时来查询一次数据库，然后启动线程来执行发送邮件的方法。

具体Timer实现如下：

timer.scheduleAtFixedRate(new PlanNoticeListenerThread(), \_date,

Integer.parseInt(CitiPFService.p.getProperty("selSeptumTime"))\*60\*60\*1000);

而监听还是在web.xml里配置，具体如下：

<servlet>

<servlet-name>NoticeListenerServlet</servlet-name>

<servlet-class>com.citipf.liyunpeng.noticeListener.NoticeListenerCmd</servlet-class>

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

</servlet>

其中的load-on-startup表示在服务器启动的时候就执行此Servlet。这样启动服务的时候就开始计时，每日的特定时间查询数据库，根据符合条件的用户来发送邮件通知。

1. 测试、运行及维护
   1. 单元测试
      1. 登录模块测试

对本系统各个板块都分别进行来的测试，测试用例如表 6‑1登陆功能单元测试用例表：

|  |  |
| --- | --- |
| 测试用例 | 测试结果 |
| 正确用户名和密码 | 测试通过，正常登陆 |
| 错误用户名和密码 | 测试通过，提示错误 |
| 用户名或密码其中一个错误 | 测试通过，提示错误 |
| 用户名和密码正确并且为特殊符号 | 测试通过，正常登陆 |

表 6‑1　登陆功能单元测试用例表

* + 1. 个人理财模块的测试

个人理财模块的测试，是通过运算的数值来测试的，具体用例如表 6‑2　个人理财规划阶段测试用例

|  |  |
| --- | --- |
| 测试用例 | 测试结果 |
| 对理财规定不合理的数值输入 | 测试通过，正常规划 |
| 大于规定边界值数值 | 测试通过，提示错误 |
| 小于规定边界值数值 | 测试通过，提示错误 |
| 大于小于组合边界值数值 | 测试通过，提示错误 |

表 6‑2　个人理财规划阶段测试用例

* + 1. 数据推送模块测试
* 数据推送功能是个新技术，具体测试过程如下：
* 将系统服务启动
* 将数据推送功能开启
* 通过多个不同的电脑，同时访问本系统
* 测试推送的数据是否能够接受并且准确
  1. 集成测试

对系统各个模块组装起来进行了系统的集成测试，测试结果通过，各模块之间的数据交互和使用无任何问题。

* 1. 系统测试
     1. 浏览器测试

Flex的特点之一就是可以夸浏览器，因为他编译出来是swf文件，所以同样的swf文件在各浏览器中解析应该是相同的，我们对本套系统进行了浏览器测试，具体测试用例如表 6‑3　浏览器测试用例

|  |  |
| --- | --- |
| 浏览器 | 测试结果 |
| IE 7、8、9 Platform Preview | 测试通过，正常浏览 |
| Google Chrome 3、4、5 | 测试通过，正常浏览 |
| FireFox 3.5、3.6 | 测试通过，正常浏览 |
| Opera 10 | 测试通过，正常浏览 |
| Safari 3、4 | 测试通过，正常浏览 |

表 6‑3　浏览器测试用例

* + 1. 系统运行测试

Flex不仅仅是可夸浏览器，而且还可以夸操作系统，其实主要是支持FlashPlayer的操作系统都可以正常运行，具体测试用例如表 6‑4　系统运行测试用例

|  |  |
| --- | --- |
| 操作系统 | 测试结果 |
| Windows XP、Windows Vista、Windows 7 | 测试通过，正常浏览 |
| Windows Server 2003、2008、2008 R2 | 测试通过，正常浏览 |
| Ubuntu 9.10 Ubuntu 10.04 | 测试通过，正常浏览 |

表 6‑4　系统运行测试用例

以上系统运行测试时通过在Windows 7系统中安装了VMWare 虚拟机而对各系统进行虚拟化安装来进行测试的。

1. 结　论

在一套业务较为复杂的系统中，尤其是对于金融、财务的方面的对稳定性和准确性较高的系统中，系统的设计显得尤为重要，而架构设计是较软件设计更为深奥的一个层次。

其实“设计”一次最早就是源于艺术，慢慢的，在软件中，融入了“设计”的思想，其目的其实也是让事物变得更美，从这个想法出发，使得架构的设计，软件的设计变得更为优美，更为牢固，更为健壮，散发出更多的优点。做出好的软件，就要知道“软件架构的艺术”。

本套系统，运用了软件架构的设计和软件设计的思想，对《花旗个人理财规划平台》进行了解决方案分析和系统设计。