

浅谈通用软件架构设计概念的应用

徐铮弦 王 鸣

[中国烟草上海进出口有限责任公司]

【摘 要】 要确保信息系统达到可靠性、安全性、可维护性、可用性、可扩展性要求,开发中应用通用软件架构设计概念是比较理想的选择。

IEEE 标准 1471 中对架构是这样定义的:架构是在组件及其彼此间和与环境间的关系引导设计发展原则中体现的系统的基本结构。UML1.5 中对架构的定义是:架构是系统的组织结构和相关行为;架构可被重复分解为通过接口、互联部分的关系和结合部相互作用的部分;通过接口相互作用的部分分包括类、组件和子系统。

我们认为所谓架构目前存在着三个层次的概念。一是针对数据存储等处理(数据库处理)、基本事务处理等的基础级架构,即系统框架或者系统结构;二是以对于共性功能最大化抽象后封装成类库集合为特征的应用级架构;三是强调应用和应变的,注重内外部环境交互的面向服务级架构,即当前红透业界的“SOA”架构。

中国烟草上海进出口有限责任公司管理信息系统作为管理模式的支撑信息系统和主要实现手段,功能非常庞大和复杂。为此,我们采用了通用软件架构设计概念,期望获得可靠性、安全性、可维护性、可用性、可扩展性的成效,并且主要涉及了基础级架构层面和应用级架构层面。

采用通用软件架构设计概念尽管可能带来一定的风险,但为了企业信息化的进一步有效推进,我们非常有必要进行这方面的探索和尝试,并且努力向更高级的层次(比如 SOA 架构设计)迈进。

【关键词】 架构 框架/结构 可靠性 安全性 可维护性 SOA(面向服务级架构) 可用性 可扩展性

前言

中国烟草上海进出口有限责任公司是上海烟草集团旗下经营烟草制品进出口贸易的企业。公司自 1998 年起,逐渐加大夯实基础管理工作的力度,特别是 2004 年底初步建成“以现代管理理念为先导、以先进信息技术为后盾”,覆盖企业经营管理全过程的“运营、行政、财务、人力资源”进出口公司管理模式,公司的整体经营管理上升到了一个较高的水平。该管理模式是以信息系统的配套作为主要的实现手段,因此对系统的可靠性(Reliable)、安全性(Secure)、可维护性(Maintainable)、可用性(Customer Experience)等方面都提出了较高的要求。此外,为

了应对复杂且频繁的企业环境变化引发的需求变化,系统需要具备较好的可扩展性(Scalable/Maintainable)。要确保系统达到“ 五性 ”要求 ,特别是对于大中型系统而言 ,开发中应用通用软件架构设计概念是比较理想的选择。

第一章 什么是软件架构

早在 20 世纪 60 年代 ,诸如 E. W · 戴克斯特拉就已经涉及软件架构这个概念了。自 20 世纪 90 年代以来 ,软件架构这个概念开始越来越流行起来。卡内基梅隆大学和加州大学埃尔文分校在这个领域作了很多研究。卡内基 · 梅隆大学的 Mary Shaw 和 David Garlan 于 1996 年写了一本叫做《Software Architecture perspective on an emerging Discipline》的书 ,提出了软件架构中的很多概念 ,例如软件组件、连接器、风格等等。加州大学埃尔文分校的软件研究院所做的工作则主要集中于架构风格、架构描述语言以及动态架构。

架构(Architecture)会以各种形式展示自己 ,且大部分架构的定义是非常模糊的。IEEE 标准 1471 中对架构是这样定义的 :架构是在组件及其彼此间和与环境间的关系引导设计发展原则中体现的系统的基本结构。不难理解架构是在某种原则中体现出来的系统基本结构。UML 5 中对架构的定义是 :架构是系统的组织结构和相关行为 ,架构可被重复分解为通过接口 ,互联部分的关系和结合部相互作用的部分 ,通过接口相互作用的部分包括类 ,组件和子系统。虽然在某些方面定义有些区别 ,但我们可以看到大部分是相同的。例如 ,大部分定义都指出一个架构关注于结构和行为 ,仅关注于重要决定 ,可以与架构风格一致 ,受受众和环境的影响 ,体现基于原因的决策等等。

软件架构有时候是一个非常容易理解的概念 ,多数工程师(尤其是经验不多的工程师)会从直觉上来认识它 ,虽然要给出精确的定义很困难 ,特别是一般很难明确地区分框架设计(或者结构设计)和构架设计。如果你要求人们为你描述“ 架构 ” ,十分之九的人都会参照结构来解释。这在关于构建或其他土木工程结构(例如桥梁)中非常常见。虽然这些条目中的其他属性(例如行为 ,目的适当性和美学观念)也存在 ,但是结构的属性是最熟悉的和最经常被提到的。为你描述软件系统架构的人往往会给你展示一份系统结构方面的图表 ,无论这些内容是否是架构层 ,组件 ,或是分布结点。架构显然不和框架或者结构(Structure)等同 ,架构的内涵要丰富得多 ,事实上 ,框架或者结构是架构的基础属性。构架不仅是结构 ,IEEE Working Group on Architecture 把其定义为“ 系统在其环境中的最高层概念 ” [IEEE98]。构架还包括“ 符合 ”系统完整性、经济约束条件、审美需求和样式。它不仅注重对内部的考虑 ,而且还在系统的用户环境和开发环境中对系统进行整体考虑 ,即同时注重对外部的考虑。

从架构设计师的角度来看 ,架构就是一套构建系统的准则。通过这套准则 ,我们可以把一个复杂的系统划分为一套更简单的子系统的集合 ,这些子系统之间应该保持相互独立 ,并与整个系统保持一致。而且每一个子系统还可以继续细分下去 ,从而构成一个复杂的企业级架构。

当一名架构设计师在构建某个企业级的软件系统时 ,除了要考虑这个系统的架构以及其应具有的功能行为以外 ,还要关注整个架构的可用性 ,性能问题 ,容错能力 ,可重用性 ,安全性 ,扩展性 ,可管理维护性 ,可靠性等各个相关方面。有的时候一名好的架构设计师甚至还需要考虑所构建的系统架构是否合乎美学要求。由此我们可以看到 ,我们衡量一个好的架构设计并不能只从功能角度出发 ,还要考虑很多其他的因素 ,对任何一个方面的欠缺考虑都有可能为整个系统的构建埋下隐患。一般而言 ,架构有两个要素 :

1、它是一个软件系统从整体到部分的最高层次的划分。

一个系统通常是由元件组成的 ,而这些元件如何形成、相互之间如何发生作用 ,则是关于这个系统本身结构的重要信息。详细地说 ,就是要包括架构元件(Architecture Component)、联结器(Connectot)、任务流(Task - flow)。所谓架构元素 ,也就是组成系统的核心“ 砖瓦 ” ,而联

结器则描述这些元件之间通讯的路径、通讯的机制、通讯的预期结果,任务流则描述系统如何使用这些元件和联结器完成某一项需求。

2、建造一个系统所作出的最高层次的、以后难以更改的、商业的和技术的决定。

在建造一个系统之前会有很多的重要决定需要事先作出,而一旦系统开始进行详细设计甚至建造,这些决定就很难更改甚至无法更改。显然,这样的决定必定是有关系统设计成败的最重要决定,必须经过非常慎重的研究和考察。

软件,从本质上来说,就是现实世界在计算机中的模拟。在考虑应用软件系统架构的时候,实际上,考虑的问题主要在于:处理什么?怎么处理?如何应用及应变?因此,应用软件系统架构,需要关注的方面,概括起来主要包括以下三个大类:

- 处理的对象,也就是数据。
- 处理的方式,也就是如何来处理系统的逻辑,即应用服务层。
- 如何进行交互,这个交互包括用户(使用者),以及外部系统。

或者说,我们认为所谓架构目前存在着三个层次的概念。一是针对数据存储等处理(数据库处理)、基本事务处理等的基础级架构,即系统框架或者系统结构;二是以对于共性功能最大化抽象后封装成类库集合为特征的应用级架构;三是强调应用和应变的,注重内外部环境交互的面向服务级架构,即当前红透业界的“SOA”架构。

第二章 采用通用软件架构设计概念的预期成效

中国烟草上海进出口有限责任公司管理信息系统作为管理模式的支撑信息系统和主要实现手段,功能非常庞大和复杂。为此,我们采用了通用软件架构设计概念,期望获得如下成效:

1、可靠性(Reliable)。软件系统对于用户的商业经营和管理来说极为重要,因此软件系统必须非常可靠。

2、安全性(Secure)。软件系统所承担的交易的商业价值极高,系统的安全性非常重要。

3、可维护性(Maintainable)。软件系统的维护包括两方面,一是排除现有的错误,二是将新的软件需求反映到现有系统中去。一个易于维护的系统可以有效地降低技术支持的花费。

4、可用性(CustomerExperience)。软件系统必须易于使用。

5、可扩展性(Scalable/Extensible)。软件必须能够在用户的使用率、用户的数目增加很快的情况下,保持合理的性能。只有这样,才能适应用户的市场扩展得可能性。此外,在新技术出现的时候,一个软件系统应当允许导入新技术,从而对现有系统进行功能和性能的扩展。

采用通用软件架构设计概念不仅预期能够取得上述成效,而且对于生产商品化软件的专业公司而言,还可以期望获得如下成效:

1、可定制化(Customizable)。同样的一套软件,可以根据客户群的不同和市场需求的变化进行调整。

2、市场时机(Time to Market)。软件用户要面临同业竞争,软件提供商也要面临同业竞争。以最快的速度争夺市场先机非常重要。

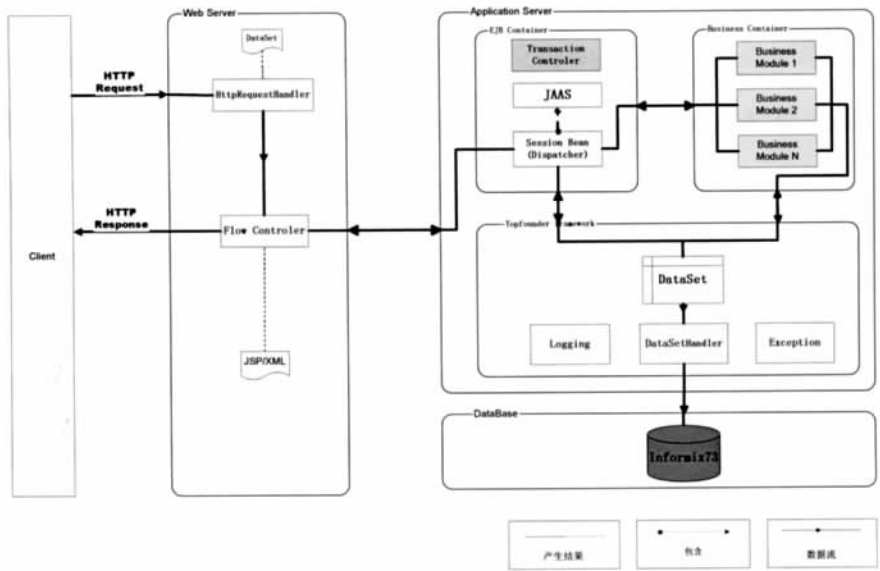
第三章 通用软件架构设计概念的应用

我们采用通用软件架构设计概念开发中国烟草上海进出口有限责任公司管理信息系统,主要涉及基础级架构层面和应用级架构层面。

3.1 基础级架构层面

在应用软件系统架构中,数据是处理的基本对象,程序总是以一定的数据结构来表现数据,并且,在使用面向对象语言开发的系统中,数据总是以类和对象的形式表现出来。另外一

方面 ,数据总是需要存储 ,对于大部分应用软件系统来说 ,通常会采用关系型数据库来保存数据。这样 ,由于数据在程序和数据库中表现格式的不一致 ,就必然要求在两者之间进行映射。应用软件系统架构 ,是软件工程的重要组成部分。设计一个好的架构主要目的 :尽最大的可能 ,提高软件开发的效率和软件质量 ,把不必要的工作和容易出错的工作 ,交给架构去处理。目前我们系统采用 JSP + Servlet 调用 EJB 的方式 ,最大限度地把 MVC 分离开来。这个架构底层实现主要依赖于两个模块 :JDBC. NET 和 taglibs。这两个模块也是可以脱离现有架构独立存在的 ,其中 ,JDBC. NET 模型在数据集(dataset)中存储数据并使用数据适配器读取和写入数据库中的数据 ,则 taglibs(标记库)是指在 Jsp 页面中人们为了某种特定的用途或者目的 ,将一些标记放到一起而形成的一种库。下图为本系统框架的整体流程图 :



其中 Web Server 的主要功能包括三部分 :接收数据 ,数据转换 ,数据验证 ,调用 EJB ,返回处理后的数据。

Application Server 的主要功能是 :接受 Web Server 传入的数据 ,由 EJB 进行业务流程设定的规则进行分发 ,执行各个业务逻辑模块 ,将结果保存到数据库 ,返回处理后的数据。

3.2 应用级架构层面

作为一个应用级的架构必须对整个系统的共性应用处理进行抽象 ,让有共性、有规律的东西由架构来统一处理 ,但架构的设计并不仅限于此 ,还应从架构设计的原则和涉及商业领域出发主动为应用系统提供更高的价值。本系统的应用级架构基于上述思想从安全性、可维护性、可用性、可扩展性几方面出发进行设计、实现。

安全性

包括数据的安全性及系统本身使用的安全性 ,数据库的安全性虽然是安全的重要组成 ,但其与本文主题关系不大 ,此处不做重要描述。本系统的安全策略是分为 :用户名及密码的安全、系统本身的访问权限的管理。

系统的访问权限 ,本系统在架构设计阶段充分了解分析了业界使用较多得安全策略管理机制 ,最终采用了‘角色’的概念 ,通过实现角色的概念 ,结合我公司实际情况将安全策略分为三种权限 :登录权 ,查看权 ,处理权。对同一角色赋予同样的权限 ,方便了使用中权限管理的设置工作量 ,也增加了系统安全性 ,实现了让合适的人看合适的信息 ,合适的人处理合适的业务。

可维护性

系统的维护性主要从两方面考虑,一种是由于系统开发本身存在的缺陷,在开发阶段没有发现出来;另一种情况是在系统使用中,由于外界的变化或本身管理要求、业务流程、操作要求的变化也会引发维护工作。

如何使投入使用的系统维护工作更方便是系统架构设计重要的考虑内容。本系统架构在设计时体现了下面的设计思路:

1、尽量将通用的处理抽象到高层次,通过高层次的逻辑实现提供应用级的统一处理(这些处理主要包括对单据的主要操作,如:新增、修改、删除、保存、发送审批、单据生效、作废等)。通过这样的设计实现了下面的效果,对于一种较简单的单据,在数据库表设计好的基础上,仅通过系统配置给它相应的操作功能,此单据就可在系统中运行,不需要其他的任何代码开发。

2、尽量将一个复杂的业务逻辑分为小粒度的逻辑单元,实现业务逻辑的插件结合形式。此做法一是可以将逻辑细分,当业务逻辑发生变化时可容易的将新的逻辑安装上去而不影响其他逻辑;另一个是当其中一个小逻辑需要屏蔽掉或暂时屏蔽掉时,仅需要将其从系统配置中注销掉相应的类就可实现,还有一个好处是将一些通用的工具可方便的在整个系统中共享,例如:在日常的业务使用中许多单据需要按照固定的规则产生单据的编号、日期+流水号,在本系统中有一个工具类,其作用就是实现此功能。当某个单据需要产生单据号时在系统的配置表中将这个类配置上去,就可实现,当不在需要时只需在配置表中去掉配置文件就可去除此功能。

本系统的架构不仅在维护工作开展上提供了方便的功能,也在问题、错误判断上采取了一些措施,以便于问题发生时快速的定位,方便的判断、分析产生问题的原因。在这方面本系统架构主要采取了两方面措施,提供了统一的异常处理、记录详细的操作日志。

异常处理包括:业务异常(业务异常主要是体现系统的使用性,此处不具体说明)和系统异常,系统异常主要是为了维护提供方便,记录系统产生的异常状况,主要是服务器异常,并记录在服务器端,以便分析问题产生的原因。

详细的操作日志:架构提供了一个整体的监控功能,记录下系统中各种操作,例如:什么单据在什么时间由谁做了什么操作。这些内容与系统异常记录进行结合分析,可快速地判断问题产生的原因,如果是业务流程中实现的问题也可快速定位到业务问题。

可用性

对于可用性的考虑首先要从界面风格的统一、操作风格的统一、操作功能的统一入手。一个好的系统在其架构设计阶段就应该考虑操作以及界面在本系统内部的统一,使用系统的用户都不希望每一个不同的业务单据都具有一种特殊的操作风格,都希望只要学会操作一种单据,就可以在本系统内方便地操作所有的类似单据。这也是微软开发的 Windows 为何能够如此风靡的原因之一。

本架构以任务的方式串接业务流程。在架构设计考虑到在系统中提供业务流程的设置功能,通过设置可清楚的知道,一项工作开始后它的后续环节有那些。当一个环节结束后系统通过流程设置判断出下一个环节是什么工作,需要由谁来完成。这主要是为了提高业务由多个用户处理时的效率。用户进入系统后,不需要等待其他人员的通知和提醒,系统会提供给你当前你有什么工作任务需要开展,任务完成的时间限制是什么。

系统从架构层提供了较好的、统一的系统提示功能,当业务人员在处理业务时违反了一些业务规则,系统通过统一的异常处理机制将预先设置的提示信息提供给使用者。

方便的数据共享功能,本系统的流程设置功能一方面是串接业务流程,实现任务提醒功能,另一方面也提供了方便的数据共享功能。例如:当卷烟分析单制作完成后,其后续单据如销售发票、出货报告(提货单)、投保单等,其实数据与分析单中没有太大的区别,在系统中不可能让客户在后续单据中将相同的内容重新录入一遍。架构在设计时使用下面的方法保证其共享性,除了设置任务流程外还提供一个数据对应维护的记录表,记录了前后环节数据的对应

关系。当用户通过系统得知某项任务需要开始时,用户开始操作后,系统通过设置的前后环节对应关系自动将前一环节的数据信息带到当前工作界面上,几乎彻底避免了当前环节的重复录入工作量。并且此数据对应维护表中的内容是可以通过架构中提供的工具进行维护的,当对应关系需要增加、修改时只要对其进行适当的维护,不需要任何代码就可实现新的对应关系。

可扩展性

可扩展性在架构设计时主要从两方面考虑的,一个是系统使用人员数量、次数增加以及数据容量扩大的扩展能力;一个是当前业务流程的扩展性以及未来业务发展的扩展性。

本系统的主要用户人员及数据量基本比较固定,其数据容量的扩展也很有限,所以针对扩展性更多的是对业务扩展的考虑和设计。

本系统在考虑通用功能抽象时,不仅仅是由架构设计师独立完成的,其中也包含了很多有经验的需求分析师的心血,技术人员与需求人员经过多次沟通,从业务角度对流程进行环节拆分,将线性的业务流程分解为一个一个的功能处理,并结合业务流程需要,最终确定了架构支持的通用处理功能。这些通用功能经过组合后可满足绝大多数业务单据类的流程要求,对于特殊的要求也从架构中设计了一定的接口规则,可以支撑其他形式的扩展。

系统架构提供的其他功能

数据初步验证功能:在进行信息系统开发时不可避免的要考虑数据录入时的校验功能,即判断用户录入的数据是否与数据库设置类型匹配、是否与数据库设置的长度要求符合等内容。本架构在设计时采用 taglibs 统一地实现数据验证功能,不需要每个开发人员单独的实现数据验证功能。

审批功能:在进行管理信息系统的开发中不可避免地要考虑管理中的审批程序在系统中的实现,审批功能是管理信息系统建设的一个重要组成部分,其功能也是我公司信息监管管理模式的重要组成部分。本系统的架构在设计时考虑到审批的通用性,但由于审批是功能的集合,不仅仅是单一的通用功能,所以本系统架构中将审批作为一个架构的工具实现,使其与系统架构有机的结合在一起,成为本系统架构的一个重要组成部分。

页面生成器:

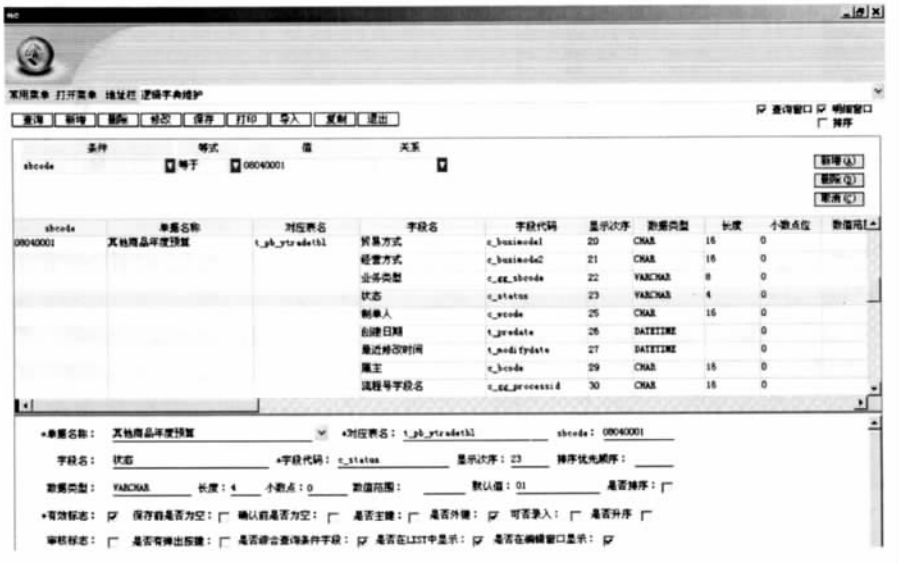


图 1:先通过预先开发好的工具设定需要生成页面的各项参数。

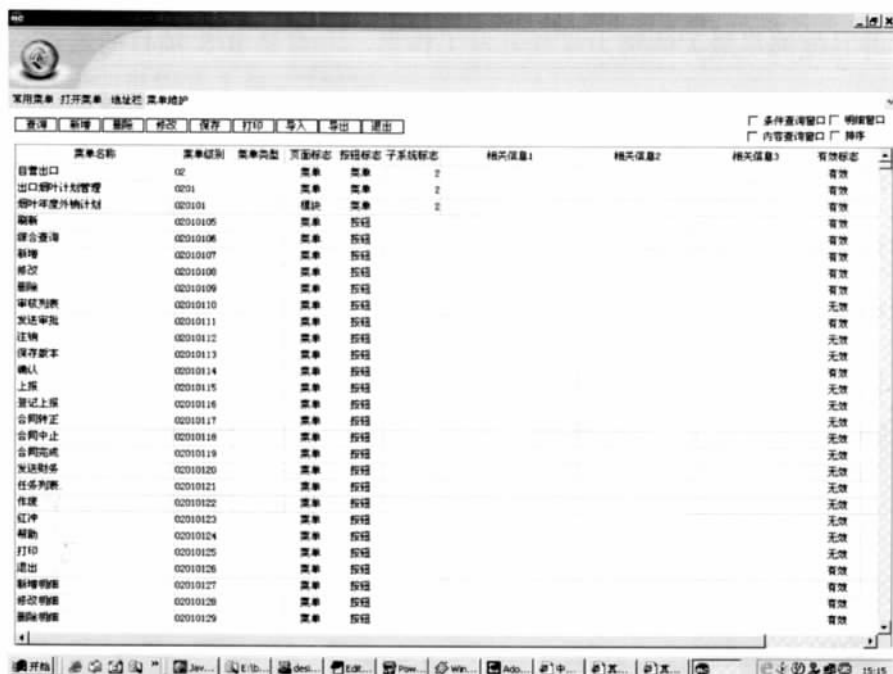


图 2 先通过预先开发好的工具设定需要生成页面的各项参数。

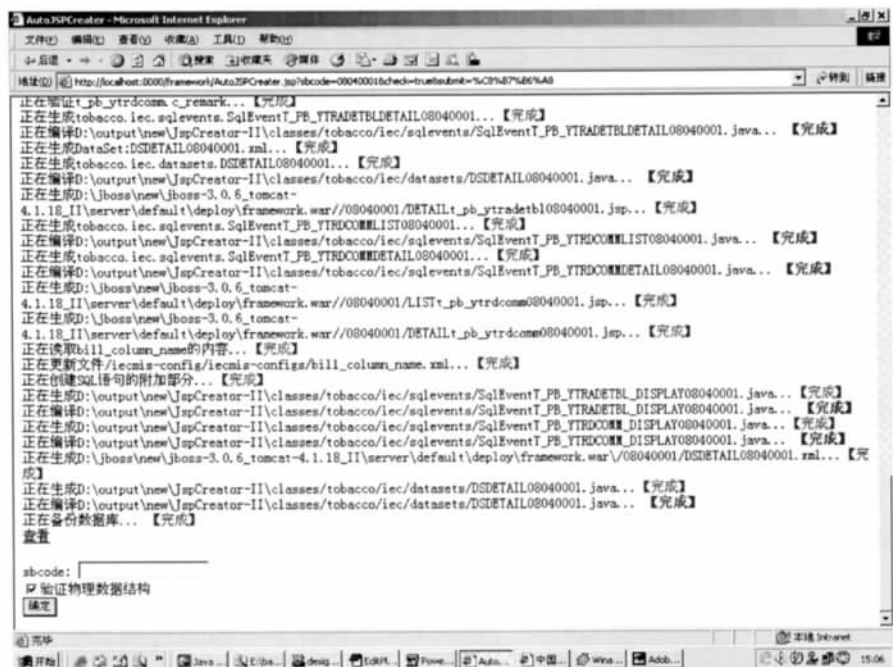


图 3 轻松点击页面生成器 完成页面生成。

我公司系统是基于 B/S 结构的,所以不可避免的有大量的页面开放工作要做。如何更充分的体现可用性、可维护性,并且能够尽最大的努力降低开发工作量,一直是 B/S 项目的难点,也是项目中重点思考的问题。基于以上的问题和困难,在架构设计阶段我们设计、开发了页面生成器工具(见图 1 - 图 3),由其来完成烦杂的页面代码编写。使得界面开发在完成设计的前

提下,由工具生成页面,大大提高了界面开发速度,以及开发的灵活性,进而降低了开发成本,并且为将来的维护工作提供了良好的基础。

发布工具 此工具实际是体现架构设计的可维护性,但由于其仅仅是提供发布的便利性,所以不能将其定位于架构,只能说是架构附属的一个工具。由于本系统的结构较复杂,在发布时涉及到框架类、业务类、功能类、页面代码、系统表、系统配置表等很多相关的信息,并且需要从开发环境、测试环境、正式发布环境顺序发布,如果凭借人工去完成这些工作,其工作量较大,并且极容易出现错误。本工具的主要目的是解决这些实际的问题,让工具去做这些固定的、繁杂的、极容易出现错误的事务可以有效地保证系统发布的正确性、高效性。

3.3 应用总结

在应用级架构的设计中抽象系统的业务逻辑处理,既是其核心部分也是难点。如何合理的构建业务逻辑、如何提供业务逻辑层的服务,以及表现层如何访问业务逻辑提供的功能,是应用软件系统需要重点关注的问题。在这个方面,业界已经发展了很多可供选择的范式,如契约式设计、SOA 架构(面向服务的架构)等。这些方法指明了设计的方向,有待在今后的实际开发中加以研究和应用。

第四章 通用软件架构设计概念应用的心得

4.1 采用通用软件架构设计概念的风险

如第一章已经讲到,当一名架构设计师在构建某个企业级的软件系统时,除了要考虑这个系统的架构以及其应具有的功能行为以外,还要关注整个架构的可用性、性能问题、容错能力、可重用性、安全性、扩展性、可管理维护性、可靠性等各个相关方面。这就需要架构设计师不仅应该是一个成熟的、具有实际经验的并具有快速学习能力的人,而且他还应该具有良好的管理能力和沟通能力。只有具备了必需的能力,架构设计师才能在关键的时刻作出困难的决定,这就是一名架构设计师应该承担的责任。从角色上来看,架构师不仅会负责端到端的服务请求者和提供者的设计,并且会负责对系统中非功能服务请求的调研和表述。对于任何一名经验丰富的架构设计师来说,不论他是采用基于传统的架构设计方法(基于 J2EE 架构或者 .NET 架构)还是采用基于 SOA 的架构设计方法来构建一个企业级的系统架构,具有相关商业领域的知识对于架构设计师来说都是必不可少的,架构设计师往往可以通过实际的工作经验积累以及接受相关的专项培训来获得这些商业领域的知识。除了具有相关商业领域的知识以外,一名合格的架构设计师必须具有较广泛的技术背景,这可能包括软硬件、通信、安全等各个方面的知识。但这并不意味着要成为一名架构设计师就必须熟悉每一门具体技术的细节,架构设计师必须至少能对各种技术有一个整体上的了解,能够熟知每种技术的特点以及优缺点,只有这样架构设计师才能在特定的应用场景下正确地选择各种技术来设计企业整体架构。

但是,架构带来的效益是丰厚的,在我们努力控制这种风险的前提下,非常有必要进行这方面的探索和尝试。此外,对于项目团队来说,统一的系统架构的意义不仅仅在于软件开发的本身。一个统一的系统架构,也是知识管理的重要组成部分。项目团队如果有一个明确的软件架构,那么,这些架构就可以成为凝结项目团队开发人员经验、智慧的载体,并且可以在不断的实践中加以充实和完善。由于项目团队的软件系统的架构比较统一,那么当某个项目成员更换或增加开发人员的时候,后来的人也能够比较容易接手,这对于项目团队的开发管理是具有非常重要的意义的。

4.2 对 SOA 的展望

我们目前对架构的应用还仅仅停留在应用级层面,对于红透业界的“SOA”还处于非常谨慎地探索阶段。

SOA 本身就是——一种面向企业级服务的系统架构,简单来说,SOA 就是一种进行系统开

发的新的体系架构,在基于 SOA 架构的系统中,具体应用程序的功能是由一些松耦合并且具有统一接口定义方式的组件(也就是 service)组合构建起来的。因此,基于 SOA 的架构也一定是从企业的具体需求开始构建的。但是 SOA 和其它企业架构的不同之处就在于 SOA 提供的业务灵活性。业务灵活性是指企业能对业务变更快速和有效地进行响应、并且利用业务变更来得到竞争优势的能力。对企业级架构设计师来说,创建一个业务灵活的架构意味着创建一个可以满足当前还未知的业务需求的 IT 架构。

利用基于 SOA 的系统构建方法,一个基于 SOA 架构的系统中的所有的程序功能都被封装在一些功能模块中,我们就是利用这些已经封装好的功能模块组装构建我们所需要的程序或者系统,而这些功能模块就是 SOA 架构中的不同的服务(services)。

因此,SOA 架构本质上来说体现了一种复合的概念,它不仅为一个企业中商业流程的组织 and 实现提供了一种指导模式,同时也为具体的底层 service 开发提供了指导。

结论：

要确保系统达到“五性”要求,特别是对于大中型系统而言,开发中应用通用软件架构设计概念是比较理想的选择。尽管这样做可能带来一定的风险,但为了企业信息化的进一步有效推进,我们非常有必要进行这方面的探索 and 尝试,并且努力向更高级的层次(比如 SOA 架构设计)迈进。

【参考文献】

1. IEEE (Institute of Electrical and Electronics Engineers)1471 – 2000

【点评】

本文在评细介绍通用软件架构内涵的基础上,介绍了进出口公司管理信息系统的基础级架构 and 应用级架构层面实现方式。主题明确,论点鲜明,有一定启发性和较高的价值。

(信息技术专业评审组)