**摘要:** 我公司于2013年年底制定了下一年度的战略目标——建立电力行业基建平台（IEPIP）产品。该产品系统包括台账、工作流、报表、权限、进度、通用办公、安全框架、数据库访问框架、页面框架等子系统。我在该产品建设中担任了系统架构师一职，主要负责整个系统的需求对接、架构设计以及详细设计等方面的工作。整个产品于2015年2月份开发完成，并发布了release1版本，以支撑应用层项目的搭建工作。 本文主要以企业信息集成的技术为核心，先分别描述了企业的内部集成和外部集成有哪些方面，并简单描述各方面适用的场景、以及其实现的方式。然后分别从产品的构件与构件之间的数据集成、工作流构件的业务流程实时构建和与外部企业或网站的组件集成三个方面来描述不同的业务需要采用不同的集成方式。在集成的过程中我们需要比较不同方案的优缺点，并择优选用。

我公司的主营业务是电力行业的工程信息化建设，公司经过多年在该业务方向的沉淀，于2013年年底制定了下一年度的战略目标——建立电力行业基建平台（IEPIP）产品。以更快捷、更经济、更优越的方式服务于该行业的客户。   
在该平台的建设中我担任了系统架构师一职，主要负责产品的需求对接，架构设计、以及详细设计等方面的工作。我们通过对已有项目的业务分析，找出了可以纳入到基础平台的业务构件有：台账、工作流、报表、权限、进度、通用办公、安全框架、数据库访问框架以及页面框架等。为了降低构件与构件之间的耦合，以及将来各构件之间能够灵活组装调用，我们总体上采用了独立构件的架构风格，即每个构件做为一个子系统进行单独的需求分析、设计、开发、测试等。在应用框架选用方面，我们考虑了将来可能会跨平台部署，并综合了现有技术人员的情况，选用了轻量级的J2EE的应用框架，以及MVC分层式的设计模式。在数据环境的选用方面，考虑到国家将在全国各行业逐步推进信息一体化建设的方案，我们采用了面向主题的数据库，即在一开始就做好数据的统一规划，这让通过我们产品进行信息化建设的电力行业的数据能够互连互通，为将来的数据挖掘奠定良好的数据支撑。  
本确立了产品的构建情况之后，我们选用了迭代增量与螺旋相结合的开发模型。先从关键需求做起，然后一个一个的实现构件，并在实现的过程中不断的尝试与其它构件的集成，以验证构件的可行性。信息的集成一般分为内部集成和外部集成两个方面。其中内部集成又包括界面集成、数据集成、控制集成和业务流程集成。界面集成的集成点在页面，通过整合页面，对外提供统一访问的界面方式，提交时按原渠道交给不同的应用进行处理，解决操作不便的问题；数据集成的集成点在数据访问层，通过中间件更新数据库的方式，保持数据一致；控制集成与业务流程的集成点都在程序的内部结构中，需要根据业务的实际情况，重组结构，重新开发。外部集成分为企业门户网站的集成和合作伙伴信息系统的集成。企业门户网站又包括仅显示企业信息的企业网站、增加信息交互的企业信息门户、企业知识门户和企业应用门户。在IEPIP产品的建设过程中我们使用了多种集成方法，下面将详细阐述。   
首先，构件与构件之间的交互我们采用了数据集成的方式来进行。以台账子系统与权限子系统的集成为例，系统管理员在台账构件中增加一个实际的台账信息，就会产生一组该台账的操作功能的权限数据，该数据需要放入到权限构件的基础配置表中供权限构件调用，并进行权限的分配。该业务存在于两个构件之间，耦合在任何一方都会破坏独立构件的风格，我们引入Redis的缓存机制，在增加台账时，把该台账相关的操作功能权限数据写入Redis，而权限子系统在进行权限分配时，分别从Redis和权限数据库表中读取数据，在进行权限分配时同时把Redis中的数据同步到相关表中，并移除Redis中该信息，以保持相关信息的一致性和持久性。这样通过增加缓存中间件的方式，解耦了两个独立构件之间的交互，并把两个构件集成在了一起。　　  
然后，构件与构件之间的交互我们还引入了控制集成的方式。以台账子系统与数据访问子系统为例，数据访问构件提供了公共的数据访问接口API，台账构件在与数据库进行交互时直接调用API，即可获得相关信息。在产品的构建过程中也引入了业务流程实时构建的集成方式。这一部分主要体现在工作流构件中。系统管理员可以在该构件中，在线创建流程、创建流转的内容，并设置流程的节点和业务流转条件等信息。系统管理员创建完成之后启动，即可在流程节点用户人员的界面看到该流程的内容，并可进行填报、提交然后会流转到下一环节。当流转一段时间后，用户发现问题，还可以上报系统管理员进行流程的调整、优化，直至成为客户实际业务需要的流程，这个过程一般需要迭代很多次。   
最后，产品中部分功能的实现也需要与其它企业的系统进行集成。该部分以界面框架的天气预报功能为例进行描述。在一开始设计该功能时，我计划通过调用公网上天气预报webservice 的方式来进行集成，即通过面向服务的方式来集成；而产品组中另一个架构师提出了采用前端界面通过Ajax异步请求的方式来进行集成。后来我们经过分析、调研、预研，发现该功能更适合采用界面集成的方式，因为调用WebService 只能拿到相关的数据，我们需要按照第三方的约定来解析数据，然后再设计自己的天气预报信息展示页面，这个过程相对来说要更为复杂、需要消耗更多的时间，也会带来更多的不确定性和Bug；并且在开发的过程中需要引入大量的Web Service的Jar包，会给后端的调试、开发带来更多的不稳定性，相对来说增加了开发和运维的成本；而通过界面集成只需要在前端引入第三方已经实现好的页面，不需要后端引入Jar包来进行支持，也不需要前端进行细项的解析、页面设计；而且第三方已经花费过大量的时间来研究、实现，并做为一个整体的控件提供大家使用，技术上来说是成熟的、稳定的、可靠的，具体的维护相对来说也容易很多；如果后期有其它第三方实现效果好的，我们只需要修改前端的脚本调用地址，即可引入新的实现机制；而WebService 市面上有大量的不同的标准与技术，想要切换是非常难的一件事情。　  
在产品的实施过程中，我们通过以上几方面的集成，最终实现了构件与构件之间的灵活组装调用，并取得了比较良好的效果。企业在信息技术方面的集成过程中，常常会遇到有多种集成方案的情况来进行选择，此时我们需要进行较多的调研与预研过程，从可靠性、健壮性、扩展性等几个方面来进行横向比较，从而择优选择。这个时间上的花费是值得的，毕竟一个好的方案，将会为我们后续的开发、运维奠定一个良好的基础，提升软件的安全性、可靠性、可用性、健壮性等非功能质量性需求的同时，也可以提升软件的可测试性、可修改性和可移植性等。   
产品于2015年2月份开发完成，发布了release1版本，以支撑应用层项目的搭建工作。产品到目前为止已更新了三个版本，现在运行非常稳定，已有十几个项目在其基础上进行二次开发，并迅速搭建完成，为公司带来了很大的经济效益。