Broadview
www.broadview.com.cn

从简单的对象封装开始，逐步到快速开发的平台。

『面向对象』

项目开发经验大成：基于.NET实现

牛树长 杨海波 俞丹琛 黄智洪 著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

『面对对象』

项目开发经验大成：基于.NET实现

牛树长 杨海波 俞丹琛 黄智洪 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书以.NET C#为实现环境，通过大量的“自定义”构件由浅入深地诠释了“面向对象”理念的完整实践。针对每个控件（或应用设计）翔实地解析了“需求分析→命题抽象→设计构思→设计实现（封装）→运行效果（截图、验证）”的全过程。题材选择经典、广泛、通用，构思清晰严谨，代码经过验证，很多题材都可直接用于项目设计。以此为基础可构造自己的“开发平台”，可提高编程效率达50%以上，设计复用度达到 70%以上。这些设计构思与设计理念基本与语言无关，在其他编程语言、数据库基础上同样可以实现且效果良好。

本书能快速提升技术资本价值，特别适合于编程、需求、架构、项目管理等从业人员借鉴，也可为企业的技术积累提供参考性方案。

图书在版编目（CIP）数据

“面向对象”项目开发经验大成：基于.NET 实现 / 牛树长等著. —北京：电子工业出版社，2009.4
ISBN 978-7-121-08398-3

I. 面… II. 牛… III. 面向对象语言－程序设计 IV. TP312

中国版本图书馆 CIP 数据核字（2009）第 025387 号

责任编辑：江 立
印 刷：北京天宇星印刷厂
装 订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1092 1/16 印张：55.75 字数：1322 千字
印 次：2009 年 4 月第 1 次印刷
印 数：4000 册 定价：98.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

序

推进工业化与信息化的融合，需要工业领域和信息领域的工作者共同创新，离不开每个行业、每个企业中专职人员的探索与实践，所谓的成果就是这些精英在解决具体问题时所迸发出的智慧之光。《“面向对象”项目开发经验大成：基于.NET 实现》一书则是中核集团核工业计算机应用研究所的专家们的智慧，是长期科研、实践中积累的开发思想、实践经验及失败教训的总结与升华出来。

中国核工业集团公司秉承集团化、专业化的运作理念，积极推进科研院、所的成果迅速转化成主导产业的生产力，是集团公司持续发展的重要策略之一。核工业计算机应用研究所是集团中唯一的信息化产业单位，为了有效实现各企业的应用系统研发，构建“构件化软件开发平台”则是我们科研的重点课题。经过几年的潜心努力，我们取得了一些卓有成效的成果，现出版的《“面向对象”项目开发经验大成：基于.NET 实现》则是该成果中的一部分。

面对全国核工业迅速发展的形势，中核集团核工业计算机应用研究所在引进国外先进、成熟的信息技术产品的同时，针对核工业的特殊性不断地提升自己在关键技术上自主的能力。在企业应用层面如何推动信息技术应用的课题上勇于探索、大胆实践并取得了长足的进展。“构件化软件开发平台”是我们自行研制、具有自主知识产权的高效开发工具，特别便于迅速搭建符合于企业自身特点的信息管理系统。通过该平台所开发的各应用项目，能够与核燃料企业的生产、管理过程密切贴合，便于应用并有利于普及。该成果现已经推广到其他的能源行业，同样体现出良好的应用效果并得到用户的高度评价。该平台不仅为中核集团信息化工程、核燃料生产企业的应用提供了有力的支撑，同时也在实践的过程中得到完善与深化。

盈利是企业的宗旨。对于信息技术服务企业来说，能否实现技术复用与应用模式复用应当是实现盈利的关键所在。《“面向对象”项目开发经验大成：基于.NET 实现》一书中为技术体系的建立、设计方案的有效复用、应用系统模型的有效复用的提供了思路和实现的具体方法。

读者可以本书来体会面向对象理念的具体实践，从而使软件架构设计不再神秘，面向对象不再抽象，软件开发不再是因人而异，软件工程管理不再是一团乱麻，软件明星与一般程序员在技术层面上有机融合，从而使整体团队的创造力更加协调、高效。

核工业计算机应用研究所长 [signature]

2009-2-19

前言

本人在软件行业磨练了很多年，历经的大大小小软件项目（管理类、ERP 类）不下几十个，其中自有一些酸甜苦辣的感受可以与读者交流。很多编程人员迫切希望提高应对实战的能力，但苦于市面上很难找到这方面的参考书籍，这种感慨在职业圈中普遍存在。身边同仁的热情邀请与一些善意蛊惑促成了本书的写作，说是个人体会也罢，说是经验之谈也罢，总归是十年磨一剑，香自苦寒来，目的是与大家分享一些个人的感悟而已。

读者群体

本书基于开发过程的各个环节，总结筛选了一批经常可以被“复用”的命题素材，并将其创造成极具代表性的设计样例。以此来由浅入深地诠释面向对象的设计理念是如何被应用于实践的问题。这些设计过程基本可以做到与语言无关，但本书中则是基于.NET C#实现的具体样例。

这些样例并不局限于某个项目的具体应用，而是侧重于通用、共性的特点，当把这些通用、共性的设计成分组织成一个相对完整的体系之后，就可以形成一个具有“复用”价值的开发工作环境。以这些通用设计为基础进行应用项目的开发，可以大大提高开发效率、降低对人员素质的要求，也可以在减少测试工作量的同时保证开发质量。

只要能够坚持面向对象的设计理念，做到自定义“构件”一点也不难，在你设计了一批“构件”并能形成体系的时候，自然就会发现“构件”化的设计体系也不过如此。同时也会感慨：面向对象的殿堂原来是如此壮观与辉煌。

本书涉及软件工程的各个方面：从基本规范到数据构造，从对象设计到系统框构，从项目组织到实施交付等各个层面。如果你正置身于软件行业之中，不管你在担当着哪种角色，都会或多或少地得到一些有价值的参考。

编程人员：本书系统性地介绍了一大批非常经典的“自定义类”及“公用方法”，编程人员可以从中选择一些自己感兴趣的设计样例作为突破点，通过模拟实现或直接引用来解决实战中的真实命题。因为这些样例就是针对那些实战中带有普遍性的问题而设计的，只有能够回归实战才会有实际意义。通过这个过程可以尽快达到领悟设计理念、提高技术水平、解决实战问题的多重目的。

架构设计：架构设计是项目的难点，也是质量优劣的关键。因为它是将抽象思维付诸实践的过程，经常是代码不多，但其作用会遍及整个系统。如果单纯地介绍这种存在于有形无形之间的构思会很困难，也不容易被应用于实践。本书则特别注重了架构在体系中的形成过程与作用方式，并且通过各个层面上的设计样例来描述这些基于“控件思维”的各种具体实现，其中既包括简单的控件对象，也包括各种元素的作用关系，还包括整个系统的状态控制与功能组配。在设计样例中不乏那些贯穿于系统、要求适应性、富于变化而又基于代码难以穷尽的各种命题，只有通过抽象构思、架构方案才有可能彻底解决。比如：容器类的封装、授权机制、审计机制、菜单管理、热键配置等类似的命题都是在架构设计过程中所要面对的实际命题。

开发管理：通过全面理解面向对象的开发特点，对把握规范的执行、理解设计难度、合理指派人力资源、合理安排任务的实现顺序等问题会有更明确的认识。尤其是如何避免主观理解的差异、沟通中的偏差等常见问题，这些都需要具体的措施与策略。通过对象的封装与模板的使用，可以达到简化设计任务描述、精确控制设计目标与设计过程的目的。

项目实施：设计方案的“可配置”性是一种系统化的构思，这样的设计得益于面向对象的封装，也得益于系统架构的体系完整。基于这样的构思可以在极大程度上改善实施过程对编程人员的依赖性。设计过程侧重于业务逻辑的控制，实施过程则侧重于外在的数据表现及简单的参数调整。很多需求变化完全可以通过现场的参数配置来适应用户需求的各种变化，甚至能在现场实现应用功能的“重载”。

经营决策：如何减少对自然人的依赖程度，把历史上的技术过程积淀成企业的技术积累是企业所关心的问题。只要组织得当，通过面向对象的设计过程，不但能够有效形成企业的技术积累，还能形成“应用模型”的有效积累。一旦积累形成，就能形成有效的“复用”，也会减少对自然人的过度依赖。在技术层面上对人力资源的细分，有助于降低成本与提高效率。这些经验都已纳入本书的阐述目标，并且形成了可被复用的具体模式。

工程监理：本书涉及软件工程的大部分通用性问题（而不是针对某个具体项目），从中所能得到的规律性内涵对把握软件项目的关键节点能够起到重要的作用。这些工程理念对准确理解甲方意图、准确分辨设计方案的优劣、清晰控制进度节点、敏感发现项目中的症结所在等都会有所帮助。

面向对象之领悟

“对象”以真实的事物（不是抽象概念或通俗比喻）为原型，通过封装与继承来解决事物规律（类）与真实应用（实例）中可能出现的差别，这就是把错综复杂的命题变成整体打包的“封装”过程。由于这个过程必须要强调对事物过程的归纳与抽象，所以必然会提高设计“对象”的门槛，也正是因为“类对象”存在着这样的抽象过程，所以才有可能降低应用实例的实现难度，并能具备较大的适用范围。

很多程序员对“对象”概念的理解是“雾里看花”的状态，似乎能理解，但又不知道该怎样应用。只会使用系统提供的基本“对象”，却不会（或根本就不知道）设计或封装

属于自己的“自定义对象”。仅仅会使用别人提供的工具干活，只是一个劳动者，并不能成为一个名副其实的“设计”者。

客观地说，“面向对象”的设计理念听起来容易做起来难。经常见到一些自认为“精通”面向对象理念的程序员却从来没有实现过“对象”的设计与封装，此间不乏带有一些啼笑皆非的意味。如果把面向对象的理念比作一个殿堂，很多徘徊在门口的人都自以为早已身居殿堂之中，其实这是一个误解。

编程的意义并不在于通过“代码（或简单的应用系统对象）”来描述需求，更主要的是要通过符合“设计理念”的方式，将原始需求转变成能够体现“设计思想”的“对象（最贴近于事物自身特性的设计单元）”并通过这种代表着需求内涵的“对象”来完成“设计”。

如果要判别是否真正理解了“面向对象”的开发理念，最简单、最直接的方式就是看他是否实现过“自定义对象”的封装，因为这是对面向对象理念的初衷与归宿。

开发环境为我们提供了设计、封装、引用各种“对象”的技术手段，如何在这个舞台上抽象、构造、设计、使用属于我们自己的各种应用“对象”，才是真正展示我们驾驭“面向对象”能力的具体体现。

不要以为“面向对象”就是那么深奥、那么难以理解，其实它就像一层“窗户纸”，只要你轻轻地“捅”一下，一切问题就会迎刃而解。关键是是否意识到了这层“窗户纸”的存在，并要伸出手指即可。相反，如果没有意识到它的存在，或不愿意为此有所付出，那也只能是在殿堂的门口“雾里看花”了。

面向对象之设计

面向对象的魅力在于真实与抽象之间的演变。

管理信息系统是个强调个性化的主题，但应当明确的是：这种个性化所对应的是系统功能，而不一定是软件的设计元素与设计过程。当我们把设计粒度降低到各个“对象”元素的时候，就会惊奇地发现，绝大部分对象具备相同的形态与操作逻辑。

把问题的普遍性抽象成“类”，它将表示一个独立的事物单元，在“实例化”的过程中将会通过属性的改变实现个性化功能的引用与设置。这就是我们所说的“自定义控件”的封装与引用。这些就是我们设计自定义控件的意义所在。

项目的设计过程应当理解为事物单元“组装”过程。事物单元的抽象程度越高、涵盖能力越强，项目设计就会越轻松，可以被引用的事物单元（类）越多，实例级的代码量就会越少。在设计过程中，面向用户功能的代码量可以作为对“面向对象”程度的基本评价。

如果没有自定义控件的存在，所有的业务功能都要通过实例级的代码实现，功能与模式的重复往往会意味着代码的复制。这种设计方式只是面向过程理念在面向对象环境下的沿用而不是真正的面向对象设计。对于这一点一定要有清醒的认识。

出于事物自身的特点，完全屏蔽掉实例级的代码是不可能的，但如果抽象程度合适，实例级的代码相当于装配过程中的连接件，而不再是设计的主体所在。我们的设计目标是：尽量通过封装与派生的手段来解决问题，在实例化的过程中代码量越少越好。

代码量的减少只是一种现象，但它所表现出来的应当是面向对象的构思能力与设计水

平。只有高度的抽象与完美的封装，才有可能在最大程度上减少实例级的代码量，这才是我们的目的所在。

就项目过程而言，控件的封装只是最基本的元素单元，在此基础上，还有很多可以被抽象、归纳的功能与模式，这些可以理解为更高层次的封装，它们在解决项目代码的复用程度上具有更加简约、有效的表现能力。

规范是建立体系的基本前提，承载能力取决于数据体系的构建，适应能力则取决于参数方案的设计。项目的总体构成基本包括代码（包括控件）、数据（包括参数）、服务（参数维护与定式功能），这些就是我们实现每个设计的实际过程。

开发只是项目过程中的一个环节，实施过程中的各种变化与适应性调整也是成本消耗的重要组成部分。完成用户的功能需求只是最基本的设计要求，高水平的设计应当充分考虑在实施或应用过程中各种可能会出现的变化。这是适应性的起源。基本功能的实现来自用户的业务要求，而运行状态下的“柔性”定制则是来源于对规律的抽象与升华。关于适应性的提升是降低成本、提高开发价值的重要课题。

更高层次的设计应当是“模式”的概念，代码的复用只能解决功能点上的问题，模式的复用则会侧重于一类功能或一类实现的共性。控件基于对代码层面的封装，模式则侧重于程序结构的复用。自定义控件与自定义模式共同形成了基于项目定制实现快速开发的“工作平台”。

面向对象之管理

软件开发商要保证营利，软件从业人员要体现自身价值，其中具有共性的内涵就是“如何才能快速执行客户订单”的核心问题。这也是本书所要解决的核心问题：系统性地介绍软件开发过程中所遇到的种种难题并翔实地探讨实战应对的具体策略。这里的目标是如何高效率、高质量地实现各种企业管理应用项目的开发。

朦胧的需求、朦胧的签约，合同的约束能力形同虚设，这种项目的风险性可想而知。投资人只知道朦胧的管理目标，剩下的全部是开发商的问题了。双方都有良好的初衷，开发商也会竭尽全力，直到成本消耗殆尽时才发现离项目总体目标的实现仍然很遥远。这是很多项目过程的真实写照，也是双方都要共同面对的尴尬。面对实现难度极大、工作量极为浩繁的代码设计过程，如何才能规避这些风险，难道不值得我们采取慎重、积极的应对措施吗？

要想改变投资方的状态几乎是不可能的，争取更大的投资也不现实，剩下的只有一个可能，即“软件开发商必须要提高自身项目交付的能力”！只有妥善解决了这个问题，才有可能在市场环境下持续生存。所以，每个开发商都希望形成自己的技术积累，都希望形成自己的核心竞争力，但要想实现这个目标并不是一件容易的事，这里有人才问题、技术问题、成本问题、时间问题等。

软件工程既是一个强调个人能力的过程，也是一个强调协同效率的过程。既要突出精英的核心作用，又要把精英的设计精华被更多的人来复用。这是平抑成本与工期的主要策

略。这里的核心问题是代码的“复用”与实施过程的“适应与变通”。形成可被复用的技术积累只是一个基本的层面，如果能把技术积累组织成完整、完备的体系，则会进一步提升积累的市场价值。

很多人把这个过程看成是一个难以达到的境界，其实不然。面向对象的设计理念为技术积累的有效形成提供了最基本的技术保证，只要我们能够正确理解并能持之以恒地加以运用，形成有效的技术积累只不过是个时间问题，这一点不管对团队还是对个人，都是相同的。只要认认真真地从一点一滴做起，仔仔细细地研究并能相对完美地解决好每个命题（无论问题的大小，只要有复用价值），就能通过“自定义对象”的方式来形成有效的技术积累并能达到有效复用的目的。积累的对象多了、范围广了、解决问题的思路深化了，一个属于自己的技术体系也就会在自然过程中逐步形成。这就是所谓的“道法自然”。

目前流行的编程环境无一例外地都在支持“面向对象”的开发技术，因为它是迄今为止最能被人公认的一种开发理念，也是最能体现软件成熟度的一种实现手段。在经历了几年的普及过程之后，几乎所有应用系统都采用了“面向对象”技术，似乎这种技术已是“深入人心”。如果仔细观察一下就会发现，实际情况并非如此。很多人把使用“面向对象”的开发环境与运用“面向对象”的设计理念混为一谈。就如同会“使用汽车”与会“设计汽车”之间的差别类似，从“会用开发环境”到“掌握设计理念”之间确实还存在一个“门槛”，并不是每个程序员都能轻而易举跨过这个“门槛”的。

“面向对象”的理念既是一种思维模式，又是一种实现手段。如果忽略了思维模式的作用，就会以“面向对象”的开发环境实现着“面向过程”设计的不伦不类的开发模式，目前这种开发模式的存在带有相当范围的普遍性。

领悟并运用面向对象的开发理念，达到提高设计水平的效果，增加企业或个人的自身价值是贯穿全书的基本宗旨。

关于本书阅读

本书以.NET C#为实现的开发环境，其中的需求分析、命题构思等与语言环境基本无关，在其他语言环境下仍然具有完整的参考价值。本人曾用 Delphi、Visual Basic、Visual Foxpro 分别实现过这些设计方案，全都取得了令人满意的效果。

限于篇幅，本书认为读者基本具备了使用.NET C#的水平。为了更加通俗易懂，对于有些备受专业人员推崇的接口、虚类、虚方法等设计技巧采取了保守的态度。尽管这些技术可能会带来一些优化设计的效果，但也会增加理解与实现上的难度。在有可能采用简单技术的时候，尽量地避免了这些抽象设计的采用，这也是回避技术风险的有效措施。

为了有效利用版面，所以在代码格式上进行了压缩，请读者阅读时谅解。

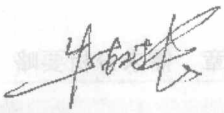
1. 去掉了代码之间的间隔行，必要时把多个命令合并到一行，极个别的码行缩进部分略有压缩，这对阅读效果略有影响，但却节省了版面空间。

2. 凡是自动生成且没有被变更的构造函数“private void InitializeComponent()”则从代码中删除，因为这部分代码可以从再现的过程中自动生成。

3．在前部分的样例中可能会引用后面一些设计作为样例，只要理解控件在样例中的作用就可以了，这些样例在后续部分将会给出更详细的介绍。

本书以通俗的方式对上述各种命题给予了真实的实现。叙述、图式等各方面都是基于简单的工作文档为素材（并非专业级的设计方案）。其中的观点纯属个人体会，不免存在各种各样的偏颇之处，恳切希望能够得到读者的赐教并请海涵。

与本书相关的设计由俞丹琛、黄智洪主持。系统构思由牛树长提供，代码实现主要由杨海波完成，其中的窗体生成器由肖庆实现，帮助文档生成器由熊峰实现。在此对上述人员及所有参加设计、编写的人员表示感谢。



2009-1-18

目 录

本章要解决表、主键、内键、外键、数据字段的命名规范，以及控件、属性、方法的命名规范。数据体系构建策略的优劣对项目影响重大，也是提高团队开发效率的关键。

本章以系统的“Button”类为例讲解自定义控件封装的实现步骤与方法，针对每个命题详细讲解归纳、抽象、构思、设计、应用的全部过程，最终形成具有通用价值的按钮类。

封装 Grid 控件需要更多的经验、归纳、抽象与参数，封装向着功能化发展。经过完善、强化的 Grid 控件集中解决了数据编程的难点问题，同时提升了数据的表现能力与控制能力。