Broadview
WWW.BROADVIEW.COM.CN

深入剖析 ASP.NET 组件设计

ASP.NET组件设计
最佳实践

黄忠成 编著

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

·李·维·工·作·室·作·品·

深入剖析 ASP.NET 组件设计

黄忠成　编著

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

内 容 提 要

这是一本专门讨论 ASP.NET 组件设计的书籍，与已有的同类书相比，本书提供了更为完整的范例，对 ASP.NET 核心运作模式做了更为深入的探讨，并且纳入了许多书籍跳过不谈的章节——IDE 环境。因此，它为目前惟一一本深入探讨 ASP.NET 核心、IDE 交互、提供完整且具商业价值范例的 ASP.NET 组件设计专著。本书重点介绍 ASP.NET 内部核心运作模式、组件设计基础与 IDE 环境协同作业，乃至如何撰写市面上常见的专业型组件等内容。作者倾力让本书涵盖几乎所有与设计 ASP.NET 组件相关的技术，带您进入 ASP.NET 组件设计的殿堂。

图书在版编目（CIP）数据

深入剖析 ASP.NET 组件设计 / 黄忠成编著. —北京：电子工业出版社，2004.5
ISBN 7-5053-9852-0

Ⅰ.深… Ⅱ.黄… Ⅲ.主页制作－程序设计 Ⅳ.TP393.092

中国版本图书馆 CIP 数据核字（2004）第 033918 号

责任编辑：周 筠 陈元玉
责任校对：张兴田
印 刷：北京智力达印刷有限公司
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
经 销：各地新华书店
开 本：787×980 1/16 印张：33.75 字数：700 千字
印 次：2004 年 5 月第 1 次印刷
印 数：5 000 册 定价：54.00 元(含光盘 1 张)

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。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。

序

相信许多使用 Delphi 和 C++Builder 的朋友都知道 Code6421，因为 Code6421 经常在“Delphi 深度论坛”上回答问题并且和网友互动，此外，Code6421 也写了许多的技术文章发表在他的网站上，这些文章帮助了许多人，也让我了解到许多宝贵的知识，当然这些文章也获得了高度的评价。

那么 Code6421 到底是谁呢？他就是黄忠成先生。

和忠成第一次见面是在 2003 年，由李匡正先生介绍的。由于当时台湾 Borland 分公司想找一些合约的讲师，因此便和忠成接触。在和忠成进行了更多的互动之后就更确定忠成是一位学有专精的 Borland 产品专家，对于 Delphi、C++Builder 和.NET 都有着非常深入的了解，更重要的是忠成的工作也集中在这些领域，这让忠成也能够在日常工作中印证理论和实务。

也许是上天已经安排好了吧，当博文视点的周筠女士（yeka）和我提起想成立一个工作室，找一些优秀的作者为中文信息技术书籍市场撰写高质量的书籍时，我的脑中立刻浮现出几个人的名字，而忠成就是其中之一。我心中知道，如果不找忠成出来写书绝对是一大损失，因为忠成不管是在技术资历或是个人素质上都是撰写技术书籍的好人才。当然，如果忠成愿意为 Borland 的产品或是技术撰写书籍的话，更是对我有极大的帮助，因为我的工作之一就是培养使用 Borland 产品和技术的潜在使用者，而技术书籍是最好的媒介之一。在我和忠成联络之后，忠成欣然允诺，当忠成告诉我他的想法和书籍内容时，和我的想法竟不谋而合，以致连我都迫不及待想捧着忠成的大作好好地品味一番。

忠成撰写书籍的这段时间刚好也是我工作最忙碌的时候，因为我同时负责了台湾、香港和大陆区域，经常需要出差。然而我还是注意到忠成撰写书籍的速度飞快，质与量都和规划的设想非常的符合。这也说明忠成不但对写作有股热忱，技术功底也十分深厚，这才能够支撑一个作者快速而且有深度地呈现想要表达的内容。

如果您是.NET 开发人员或是正要进入.NET 领域的人士，而且您对 ASP.NET 非常有兴趣，那么我建议您绝对不要错过忠成这本深入 ASP.NET 的技术书籍，因为我知道这本书绝对是在相关主题的书籍中名列前茅的好书。最后我也恭喜忠成在辛苦了这么久的日子之后，第一本心血结晶终于面世，相信一切付出都是值得的！

李 维

2004 年 5 月于台北新店

前　言

程序员总是处于一个多变的年代中，不管你现在拥有多少技术，明天你都有可能又得从头学习，此言或有悚听之意，但事实离此不远。回想一下，当年你所学会的程序语言或开发工具，今天有多少还是你手中的赚钱工具？这种现象在商用软件中更是严重，C、C++、Clipper、FoxPro、VB、Delphi、Power Builder，各式各样的开发工具与解决方案，加上百家争鸣的数据库系统，MS SQL、Sybase、Oracle，程序员的名字是超人，数不尽的技术与惊人的体力为你带来了财富，却也将你带离了家庭、亲子互动与享受人生的乐趣。许多程序员都有个通病，别人写的程序就不是好程序，总想自己动手做，“假如是我写的，就不会有这些问题！”事实上，以今日的软件规模，已经不再是一人或是单一公司所能包揽，举例来说，你不可能自己开发一个数据库系统，也不可能放着 MFC、VCL、.NET Framework 不用，土法炼出一套应用程序来，这不但不符合经济效益，也会让你丧失与亲人手牵手漫步于花前月下的机会。严格来说，现今的商用软件工业大致分成两块，一块是 3rd Party 厂商，他们开发各式各样的组件与套件，供另一块的商用软件公司使用，这种分工可以减轻程序员的工作量，至少不必事必躬亲，每个功能都自己写。在今日的软件市场中，3rd Party 市场 90%由外国人所包办，华人市场中此类厂商不多，但我们明白，这是一个快速成长的市场，也是一个具备高度需求的市场，这里面绝对存在着区域性套件的需求，只要有应用软件，3rd Party 组件或套件就能够生存。经历了十多年的职业程序员生涯后，我拿起笔杆写下了这本书，叙述 ASP.NET 组件技术与组件思维逻辑。如果你是程序员你或许不须以撰写组件维生，但我确信你必定得依赖组件维生！适当地了解它们，你定会豁然开朗，发现原来世界是那么的宽广。

黄忠成
2004 年 4 月于台北

致　谢

一本书绝不可能是一个人的努力就能完成的，没有博文视点资讯有限公司的周筠编辑、方舟编辑及其他同仁的帮忙，这本书不可能出现在各位面前，感谢他们的努力与耐心。Borland 的李维老师是催生这本书的最大功臣，感谢李老师的邀约及提供宝贵意见，以及无数次的帮忙。也感谢 Lucy 与匡正，在我离开正职转为兼职时期给予的帮助与宝贵意见，尤其是每个礼拜都得接一次我聊天电话的匡正，谢谢你的耐心。最后要感谢的是好友光政，谢谢你在我最困难的时候伸出援手，让我得以持续做喜欢做的事。

导　读

本书的重点

当你翻开这一页时，相信你明白手中的书所谈的是哪方面的技术。是的!这是一本专门讨论ASP.NET 组件设计的书籍，主轴环绕着 ASP.NET 内部核心运作模式、组件设计基础与 IDE 环境协同作业，乃至如何撰写市面上常见的专业型组件等课题。作者尽可能地让本书涵盖所有设计 ASP.NET 组件相关的技术，也尽可能地让它成为进入 ASP.NET 组件设计殿堂的开门钥匙，甚至是魔法之钥。不可否认，这样的一本书对于我与出版社来说都不是件简单任务，面对 ASP.NET Team 与 MSPress 所出版的类似原文书籍，我们感受到无比沉重的压力，倘若本书不能掘得比他们深，比他们广，那么吸引你的就只剩下此书是中文的而已，我们不希望看到这样的结果，所以竭尽全力地提供更完整的范例，更深入地探讨 ASP.NET 核心运作模式，并于本书中纳入许多书籍跳过不谈的章节:IDE 环境。在你看到这本书时，我们可以自信地说，这是市面上惟一一本深入探讨 ASP.NET 核心、IDE 互动、提供完整且具商业价值范例的 ASP.NET 组件设计专著。

阅读本书所需的基础知识

看得懂 C#，写过简单的 ASP.NET 应用程序，做过一些简单的网页，这是阅读本书所需要的基础知识。

适用的开发环境及系统配置

本书与 IDE 开发环境紧密相连，使用 Visual Studio.NET 2003 及.NET Framework 1.1，读者们必须要拥有这些产品，书中不会告诉读者如何用 Command Line 来编译程序，因为组件的用户也不会希望你卖给他的组件要用 Notepad 一个字一个字地敲入 ASP.NET Script，未拥有 IDE 支持能力的组件就像是折翼的飞鸟一般，难以生存。除了 Visual Studio.NET 之外，你也有其他的选择：可用 ASP.NET Web Matrix、Borland C# Builder、Borland DELPHI.NET 等，但基于各种开发环境支持程度不同，某些组件可能无法在这些 IDE 上运作，例如 Borland 系列的.NET IDE

缺乏 Template 编辑器，ASP.NET Web Matrix 不允许安装 Component 系组件等等。不允许安装 Component 系组件的问题很容易就能克服，只要将父类由 Component 改为 Control 就能适用于 ASP.NET Web Matrix，但 Template 编辑器就难了，因此本书要求你备有 Visual Studio.NET 或是 ASP.NET Web Matrix 其中任意一种软件。

名词引用的规则

Component、Control、Object，组件、控件、对象，在最后调整此书中所用的名词时，作者决定以组件来代表 Component，用控件来取代 Control 这个词，当你看到组件这个词时，意思不包含与用户有互动接口的 Component，例如 DataSet、SqlDataAdapter 等，当看到控件这个词时，意思指的是拥有与用户互动接口的 Control，如 TextBox、Button 等。当所提及的技术两者通用时，一律使用组件这个词。

书籍的结构

本书分为几个单元，第一章介绍组件设计的基本概念，第二章讨论.NET Framework 与组件设计相关的技巧，这两章是本书的第一单元即导引部分。第三章深入地探讨 ASP.NET 的运作模式，即从 IIS 到 Page 的生成与释放，第四、五章教读者撰写基本的 ASP.NET 组件，第三、四、五章是本书的第二单元即基础部分。第六章引导读者进入组件设计技术最神秘，也最重要的 IDE 环境，其中讨论了 IDE 的运行模式，组件如何与它互动等课题，第七章深入讨论 ASP.NET 的 Data Binding 技术，教读者如何撰写具有 Data-Bound 功能的控件，于此章中读者将遇到本书中最吸引人的控件:WebComboBox。第八章将焦点放在 ASP.NET 中最令人赞叹的 Template 技术，此章中持续地改进前面章节的控件，使其支持 Template 技术，第六、七、八章是本书的第三单元，即核心部分。第九章重点放在 ASP.NET 的 Validator 技术上，深入地探讨 Validator 的运作模式及其可能造成的困扰，最后撰写一个 PowerValidator 控件来缓解这些困扰。第十章属于锦上添花的章节，其内容包含了 WebMenu、WebDatePicker 两个控件，它们是本书中最复杂的两个控件。第九、十章是本书的第四单元，即提高部分，第十一章是本书的结尾，讨论了.NET Framework 1.1 的 Mobile Controls 技术，以简单的范例介绍撰写 Mobile Controls 的技巧，此章中同时讨论了下一代的 ASP.NET 2.0，为读者日后的发展提高预作准备。

选择适合的阅读方式

按作者的愿望，当然希望读者们可以逐章地阅读本书，毕竟那是我所喜欢的编排，不过这只是个人的主观认知，对于不同层次的读者，这里提供几个阅读本书的建议。假设你是 ASP.NET

与 C#的初学者，循序阅读对你来说是最好的方式，在第一次阅读时，别急着了解第三章所谈的核心运作模式，也别急着全盘了解第六章所讨论的技巧，待看完全书后再回头细续，会有事半功倍的效果。假设你是 ASP.NET 的老手，但从未写过 ASP.NET 组件，建议由第三章看起，其内所谈论的核心运作知识一定会给你一种“原来如此”的感受，接着阅读第四、五两章后上机实现，待上手后再回头来阅读剩下的章节。假设你已读过其他的 ASP.NET 组件设计的书籍，也写过 ASP.NET 组件，那么第三、六两章可以为你补足所需的 IDE 与核心运作知识，其余章节中的范例也可以帮你提高撰写不同组件所需的技巧。

范例在哪里

本书随附的光盘中包含了书中所有的范例及组件。

技术支持网站

作者拥有个人网站，在本书出版后，网站中会维护一份勘误表及范例更新纪要，网站中也会不定期地发表相关的文章及范例。

网址:http://www.dreams.idv.tw/~code6421

目　录

第一章

组件概论

1-1 历史的沿革

1-1-1 很久很久以前

计算机时代初期，撰写程序是件相当直觉的工作，当初软件的复杂度普遍不高，只要了解程序语言的语法，清楚客户的部分需求，几乎就能马上进入实现的阶段，接着就是边做边改。当时，软件程序通常不会有什么曲折离奇的设计，多半只是进行一些数据的存取与运算的工作，作业平台也仅限于单一计算机而已，因此程序代码通常不多，也就没有分割的必要性。随着计算机的普及，人们对于软件功能的需求也日渐增加，从简单的数据库文件到产生复杂的分析报表，原来所使用的程序语言在能力上已渐渐不敷使用。以称霸语言界多年的 C 语言为例，撰写这些复杂功能是绝对可能的，但也绝对是个大工程，在商用软件起飞的初期，这种以 C 语言作为开发工具的复杂软件的确出现过，但是其庞大的程序代码数量与高复杂度却造成了难以维护的窘境，最后大多步上了抛弃旧有软件，重头来过的结局。开发工具厂商意识到 C 语言的困境，着手开始推行新的程序语言，与以往的方向不同，此次采取了分门别类的方式，撰写一般应用程序就继续使用 C 语言，数据库应用程序则采用新发明的专属程序语言，如 DBase、FoxPro、Clipper 等。数据库专用的语言具备了简化存取数据的特色，可让设计者专注于数据与逻辑处理，不必像使用 C 语言般要自行撰写存取数据的程序代码，但是其缺点就是缺乏处理低阶动作的能力，例如 IO 控制、直接内存存取等等。有鉴于 C 语言逐渐地屈居弱势，但数据库语言又有其限制存在，基于对语言的执着与不放弃，C 语言的支持者们开始寻找延展 C 语言的解决方案，最初的目标就是提供一组 C 函数库，让 C 语言也能像数据库语言般轻易地存取数据。果不其然，很快多数的 C 语言程序员纷纷回到了本位，但更快的是他们发现这些函数库仍然无法达到与数据库程序语言一样的水平，存取数据仍需撰写大量的程序代码，生产力也依然比不上使用数据库语言来得简单与快速。在 C 语言扩充的同时，数据库语言厂商也看到了自己的缺点，开始于

数据库语言上连接 C 函数库，让设计人员能以 C 语言撰写低阶的功能，并与数据库语言连接使用，数据库语言由此完完全全地占据了数据库软件的王座，霎时间商用软件产品如雨后春笋般推出，轻型软件工业从此也进入了高峰期。

1-1-2 OOP 的年代

随着软件功能的复杂化，单一程序中重复再重复的程序代码也越来越多，维护的工作也日趋繁重，虽然程序员已竭尽所能地将程序中重复的部分独立成为小函数，最终还是无法逃出维护困难的牢笼。语言设计者看到了这一点，着手开始进行新世代语言的进化工作，OO(面向对象)概念于此时振翅飞翔，其中最老也最具代表性的语言是 Smalltalk，不过 Smalltalk 从未引起语言的全面革命，真正将 OOP 发扬光大的是由 C 进化而来的 C++语言。相较于以往文件和函数分割的设计模式，OOP 采取了以对象为主的设计模式，有效地将单一程序中程序代码重复的几率降低，挟着 C 语言霸主的余威及拥有最多用户的优势，C++俨然成为 OOP 语言的强者，OOP 的时代正式拉开了序幕。

1-1-3 Component-Oritend Programming

OOP 倡导以对象为单位的设计方式，将特定功能独立成一个对象的设计方式也随之普及化，渐渐地设计者也学会了将某个功能独立成一个对象，此种设计模式不但减少了程序代码重复的频率，也减低了后续的维护成本，许多设计者看到了此种开发模式背后所隐含的商机，纷纷推出了各式各样的对象库，主要目的在于提供程序员一个工具箱，降低其开发成本，3rd-Party 对象库由此进入了首次高峰期，同时新一代的“组件”设计模式也开始萌芽。

1-1-4 RAD 的战局

在 Microsoft 公司推出 Windows 系统之后，人们开始习惯这个以图形接口为主的操作系统，随之而来的窗口软件需求量也大幅增长，许多 DOS 软件厂商都将既有软件移植至 Windows 系统视为重点工作，但 Windows 仅提供了一组以 C 语言为主的 API 函数库，而以 C 语言撰写一个最简单的窗口程序就需要近百行的程序代码，这不但吓退了许多想移植软件的厂商，也间接阻挠了 Windows 系统的普及，毕竟当时多数用于商业的应用程序仍然是以 DOS 系统为主，如果移植成本过高，软件厂商自然就会将成本反映到售价上，而用户当然也会对升级系统做更审慎的评估，毕竟旧有的软件尚未到无法使用的地步。针对此问题，许多开发工具厂商纷纷推出了缩减程序代码的 OOP 对象库，其主要的目的在于简化设计者的工作，以预设的框架程序代码减少设计者撰写无意义却不得不写的程序代码

数量，在此时代中以 Microsoft 推出的 MFC 与 Borland 推出的 OWL 为市场两大主流，比起原来的上百行程序代码，同样的功能 MFC 及 OWL 仅需 20 行不到的程序代码即可达成，Windows 软件于此时也进入了快速成长期。但是 MFC 及 OWL 仍然无法解决数据存取的问题，与此同时 FoxPro 与 Clipper 也推出了窗口版本的开发工具，两者竞争的结果是 Clipper 不敌败阵，从主流开发工具市场中消失，Visual FoxPro 成为 Windows 上主流的数据库语言。虽然开发工具厂商所推出的对象库可以大量减少设计人员的工作，但配置 UI(用户接口)画面仍然是最耗费时间的工作，设计人员必须反复修改源代码、执行、再修改来取得满意的画面，开发工具厂商自然也看到了这个问题，开始推出所见即所得的开发环境，Microsoft 推出了 Visual Basic，不仅提供了所见即所得的窗口编辑器，还提供了一个开放式平台，允许设计者撰写 OCX 组件安装至开发工具中，很快 Visual Basic 成为 Windows 上最强的开发工具，以提供 Visual Basic OCX 组件为生的 3rd-Party 厂商也正式成为软件产业中的要角。即使如此，Visual Basic 无法操作低阶功能及解译的缺点仍让许多程序员头痛不已，此时俗称 VB Killer 的 DELPHI 推出了，这个由 Borland 公司推出的开发工具不仅拥有 Visual Basic 所有的功能，还提供了更多更丰富的组件，其使用的 Object Pascal 语言与编译型的特色掳获不少视 Visual Basic 为扶不起的阿斗的程序员，这时 RAD 开发工具之战进入了白热化状态，VB 与 DELPHI 的强大数据库操作能力渐渐迫使 Visual FoxPro 淡出主流市场。

1-1-5 RAD Web Development 的时代

Internet 的诞生，让信息技术进入了一个自杀式的疯狂跟风时代，Yahoo 的成功更让许多软件公司视 Internet 为赚钱的最快道路，网页设计的需求也随之大幅增长。在初期，网页多半是用来展示产品、或是简易互动的平台，但在商人的推波助澜之下，网页开始变成了事务平台，其复杂性也随之增高，原来的平面网页编辑软件慢慢地不敷使用，这时开始出现以网页为中心的程序语言，如 PHP、ASP、JSP、CFM，等等。不过这类以 Script 语言为主的设计语言却将时空拉回到最原始的软件时代，混乱的程序代码，重复又重复的设计，再加上难以处理的版面配置问题，一切的一切都说明此类语言的不适用，可是碍于技术瓶颈，再加上商人们急于将网络推到世界的各个角落，这个时代居然延续了几年之久。终于，如许多悲观的观察者所预期的，Internet 的泡沫化时代来临了，短短数个月之内，许多以网页为生的公司纷纷倒下，泡沫化的结果也将人们由梦想拉回现实，网络仍然是信息工业中的热门产业，只是少了当初那一种疯狂性而已。终于，程序员期待已久的 RAD 网页开发工具萌芽了，有趣的是这个动作居然是由原软件开发工具厂商所发动，而非网页开发工具厂商，DELPHI 在此发挥了其强大的扩充能力，由最初的 InternetExpress、WebSnap 一直走到了 IntraWeb 与 EWF 时代，不过这些努力仍不敌 Microsoft 公司所推出的 ASP.NET，终于 RAD Web Development 时代正式来临。

1-2 组件的定义

1-2-1 混乱的定义，Component、Control、Bean

计算机世界发明新名词的速度远超过现实世界，在推动 OOP 时，语言厂商们倡导着软件 IC 的梦想，绘制着与硬件组装一样简单的软件 IC 蓝图，Component 这个名词也应运而生，在辞典中 Component 这个名词被解释为“执行特定数据处理功能的电路或装置，又称为组件(Element)”。在近代的计算机专业辞典中，Component 被进一步解释为“可独立运作的软件单元”，不管是哪一种解释，Component 都代表着一个可独立运作的物体，独立运作！这代表着称为 Component 的东西必须拥有低耦合性，高重用性等特色。借软件 IC 的梦想所赐，Component 这个名词的使用远超过原先所设定的范围，不管是硬件、软件都有这个名词出现，名词的混沌时代就此展开。在多年的混乱之后，各家软件厂商开始针对 Component 做更细的分类，Microsoft 将 Component 划分为两部分，一部分延用 Component 的名词，意指具备特定功能、可独立运作、不具备 UI 接口的单元；另一部分则使用 Control 这个名词，意指具备特定功能、可独立运作的 UI 接口单元。另一方面，Java 为了与 Microsoft 区隔，也将 Component 分成两个部分，Java Bean 意指 Client 端 UI 或非 UI 单元，Enterprise Java Bean 意指 Server 端非 UI 单元。

1-2-2 破题，关于组件

面对这些混乱的名词，程序员常有不知该如何称呼它们的困扰，事实上！目前的定义已渐趋落实，Component 代表着具备特定功能、可独立运作的单元，不管其是否具备 UI，都可称其为 Component。接下来的分割只是随开发厂商起舞，当与 Microsoft 系设计人员沟通时，Control/Component 的分别较能让人了解，反之与 Java 设计人员沟通时，Java Bean/EJB 则通顺许多，但重点是，它们都是 Component，也就是组件。

1-2-3 软件 IC 的梦想

经过多年的努力之后，软件 IC 的梦想究竟实现了没呢？在我看来，此梦想仍未实现，因为硬件 IC 的通用性仍是组件技术所不及的，组件技术尚停留在程序语言的框架之中，或许有些人会反驳此论点，COM、ActiveX 不是早就跳出此框框了吗？是的！就理论上来说是如此，但实务上 COM 的不易撰写与使用步骤繁琐，再加上效能上的考虑，倘若这就是软件 IC 的最终型，那未来我们该怎么办？那么.NET 呢？是的！.NET 是目前看起来最具软件 IC 最终型的技术，但

是其普及仍须你我的努力，而这也正是本书的目的之一。

1-3 Library、Suite、Framework

1-3-1 Library（程序库）的定义

早期，Library 意指一群函数的集合，其全盛时期是 C 语言的时代，那时不管是数据库处理、数学运算、绘图等函数库都称为 Library，截至目前为止，此名词仍然有许多软件产品使用，不过比起以前少了许多，主要是 Library 给人的旧印象是较难上手，也有种旧时代产品的感觉。

1-3-2 新名词：Suite（套件）

这个名词是组件时代中期所常用的，通常指某一组组件涵盖该领域所有功能，如 Editor suite、Data Access Suite、Graphics Suite、Image Suite，等等。许多非开发工具厂商也使用此名词，意指 Plug-In 的子系统。

1-3-3 具学术气质的 Framework（框架）

近几年来，Framework 这个名词被大量使用，意指该组件组或函数库提供了高度的延展性，可供其他厂商撰写加值性的产品。以往，Framework 通常意味着该组件组或函数库相当庞大，拥有相当多的功能，但现在看来就不是如此了，许多轻量级的函数库也开始用起了 Framework 这个名词，因为量已经不是衡量某一组件组或函数库是否为 Framework 的指标，质才是重点。近代较为显眼的 Framework 代表大概就是.NET Framework 了，一个量与质兼备的组件库。另外在软件工程领域中也慢慢出现 Framework 的名词，代表着一种开发模式，意指运用大量的设计技巧，使成品具备高度的延展性，与组件库或函数库不同，这种 Framework 通常锁定于特定商用程序领域。

1-3-4 三思而后行

为组件组命名，大概是组件设计员最喜欢做的事，只是在享受命名喜悦的同时，也不能太过背离产品原意，例如将一组网络操作的函数库称为 Framework 就显得有些抬举，将一组抽象

化数据存取的组件组命名为 Suite、Library 又显得有些委曲，不管如何！恰如其分的名称才是重点。

1-4 Design Patterns 与组件设计

1-4-1 为何需要 Design Patterns(DP)

一本由 Gof 于 1994 年所出版的书籍，为软件工程投下了一颗足以震动软件工业的[illegible]弹，而其涟漪居然是在近两年才波及台湾地区与大陆，许多初次阅读该书的设计人员都有种相见恨晚的感觉，事实上 Design Patterns 一书中所提及的技术并不算创新，因为这些技术早已在业界盛行许久，不管设计人员有没有看过该书，多多少少都用过其内的技巧。只是这些技巧通常都锁在老程序员的宝箱中，幸运的可于源代码中窥知一二，不幸运的可能得花同样的时间来学得相同技巧，Gof 此书一出，等同将老程序员的压箱宝物摊在阳光下。除了教育新进程序员之外，Design Patterns 还具备了标准及规范的意义，其中已定的 23 个 Patterns 可作为设计人员互相讨论及沟通的标准之用，一句采用 Adapter 样式的语句，比起一大串的说明与解释来得清楚多了，在协同作业时也较容易掌控软件的质量及大方向。

1-4-2 鸡与蛋，组件 vs.Design Patterns

不管是在 Gof 推出该书之前或之后，组件设计绝对离不开 Design Patterns，原因很简单，Design Patterns 就是由以前的设计经验所整理出来的，以前的组件中必然有其影子存在。而在之后，组件设计仍然持续使用 Design Patterns，因为许多问题总是不停地重复，与其花费时间重新思考解决方案，倒不如直接取用来得快速与直观。Design Patterns 生于组件，而组件使用 Design Patterns，日后随着组件的多样化，新的 Design Patterns 也会被发明出来，周而复始，循环不息。

1-4-3 别被绑住了思路

组件使用 Design Patterns，这并不是铁规，必要时组件也必须改变原 Pattern 的设计来符合需求，程序员千万别把 Design Patterns 当成是万能钥匙，见到锁就想拿出来试试，记住！那只是一个建议与过往经验罢了，聪明的你一定知道，它们不一定适用目前的问题，别惧怕去改变它们，你的改变将使 Design Patterns 无限成长。