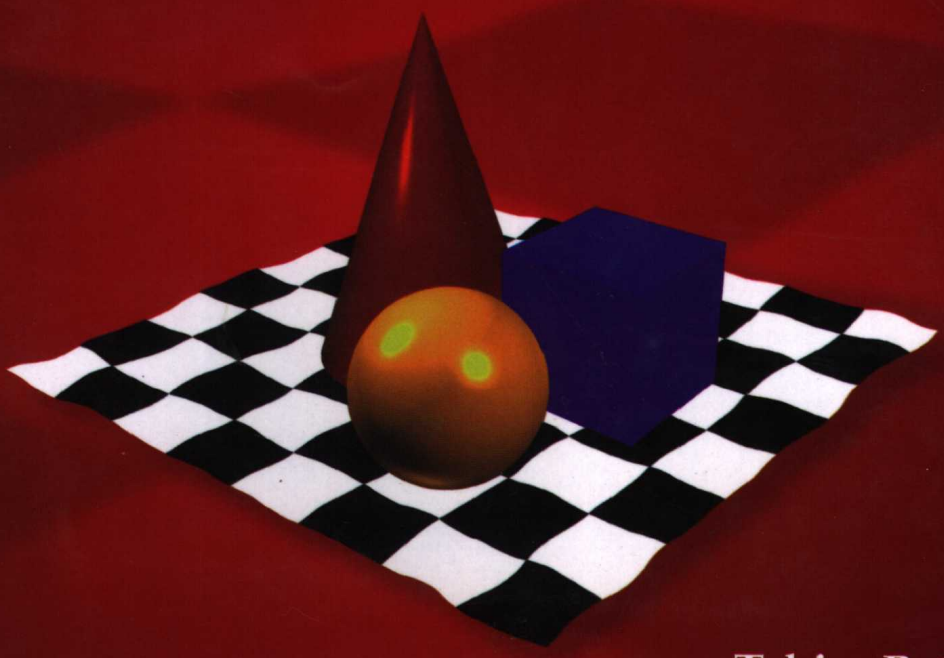
HZ BOOKS

Web Application Development with PHP 4.0

Linux与自由软件资源丛书

PHP 4.0

Web开发技术指南



附赠 CD-ROM

(美) Tobias Ratschiller
Till Gerken 著

陈军 龙浩 李向荣 等译

机械工业出版社
China Machine Press

New Riders

Linux与自由软件资源丛书

PHP 4.0 Web 开发技术指南

(美) Tobias Ratschiller
Till Gerken 著

陈 军 龙 浩 李向荣 等译

机械工业出版社
China Machine Press

本书介绍用PHP编写Web应用程序的方法。内容包括高级PHP语法、Web应用程序设计思想、基本网络应用程序策略、PHP数据库访问、尖端应用程序、PHP 4.0扩充内容等。本书内容丰富，论述严谨，不仅讲解了如何编写高效的应用程序，还讲解了为何这么做，为编写安全、稳定的应用程序提供了实际参考。本书附带光盘包括PHP、MySQL、Apache相关信息、相关应用开发工具、书中大量实例的代码。本书适合了解PHP并用PHP编写网络应用程序的技术人员参考。

本书版权登记号：图字：01-2000-2490

JS482/18

图书在版编目(CIP)数据

PHP 4.0 Web开发技术指南/（美）拉斯奇乐（Ratschiller, T.）、（美）杰肯（Gerken, T.）著；陈军等译. -北京：机械工业出版社，2001.1

(Linux与自由软件资源丛书)

书名原文：Web Application Development with PHP 4.0

ISBN 7-111-08628-7

Ⅰ.P… Ⅱ.①拉… ②杰… ③陈… Ⅲ.PHP语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字(2001)第00684号

机械工业出版社(北京市西城区百万庄大街22号 邮政编码 100037)

责任编辑：高智勇

北京昌平第二印刷厂印刷 · 新华书店北京发行所发行

2001年1月第1版第1次印刷

787mm × 1092mm 1/16 · 17.5印张

印数：0 001-5 000册

定价：40.00元（附光盘）

译 者 序

PHP是Web开发中应用非常广泛的脚本语言。它被视为脚本技术的先驱，并因其开发功能的强大性，越来越受到人们的欢迎。同样，人们对它的要求也越来越高。所以版本被一再更新。从1997年至今，PHP经历了PHP/FI1.0和2.0版升级至PHP 3.0，再由PHP 3.0升级至PHP 4.0。本书就是为了满足人们对PHP应用的进一步要求而编写的。

本书作者对PHP有很深的研究，在他编写本书的时候，付出了大量的时间和心血。所以本书涉及到的深奥问题都是经过严格认证的，并且包含大量的程序清单实例。这使得读者通过对本书的学习，能很好地掌握高级PHP的语法，并能大大提高开发者解决开发Web应用程序时遇到的问题的能力。

参加本书翻译的工作人员有陈军、龙浩、李向荣、肖虎勤、段小华等，他们在经过数月的辛勤工作之后将本书奉献给了广大读者。由于我们的水平有限，所以尽管我们为此付出很多努力，书中难免存在一些翻译不当之处，欢迎广大读者批评指正。

2000年11月

序　言

当我三年前第一次接触到PHP的时候，我怎么也想不到有一天我会为一本关于PHP的书写序言。事实上，在那个时候，编写任何有关PHP的书都会显得有一点牵强，让我们回顾一下究竟是什么使PHP成为Web开发中应用最为广泛的脚本语言。我开始从事PHP项目是很偶然的。作为一个终端用户，我在PHP/FI2.0中碰到了一个绊脚的bug，它是如此的奇怪，以致使得伙伴Audi Gut mans和我不得不深入研究，以看个究竟。当我们看到了产生PHP/FI2.0 标记的代码时，我们对它很不满意。另一方面，我们确实欣赏这样一种想法，即：开发一种内嵌HTML和服务器的服务器端脚本语言。因此，就像软件工程师一样，我们认定从零做起一定很酷，这正是一次机会。

我们编写了这种语言，并在函数模型和样本代码中投入了大量精力，这使得PHP被推进到远超过我们想象和期待的程度。今天，PHP正在超过一百万个Internet域内使用。它是经常被选择在UNIX环境下进行服务器端脚本编写的工具。PHP 4.0使人们确信，在未来的一段时间内，PHP仍将是Web脚本技术的先驱。Zend引擎（www.zend.com）使得以PHP为基础的Web站点性能和可扩展性发生了革命性的变化。其集成的会话支持，内在的XML、Jave和COM支持，以及许多其他优点使得Web开发者能够开发出更加强大的动态网站，并比以前做得更加容易。

随着前沿技术的继续发展和集成，PHP一直在更新发展。新的Java和DCOM支持，先进的XML特性，以及改善了的OOP特征进一步增加了PHP在商业环境中的可用性，使得PHP成为企业计算的可行工具。来自Zend Technologies的商业附件（例如：debugger、IDE和Compiler）将会导致进一步的飞跃。同时，PHP 4.0的内核，已经发生了革命性的结构变化，例如，Web服务器接口已被完全提取出来，这将允许除Apache 以外的Web服务器支持。一些书籍（例如你现在要读的这一本）会给你提供成功利用这些新技术的必要的背景知识。

在我看来，总的开放式资源，特别是PHP的未来是很光明的。在1997年，要说服你的经理“Linux至少和Windows NT同样稳定”，需要费很大力气。并且，在大公司中没有人想过使用开放式资源。现在，这一切都已经改变了。采用以Linux为基础的方法解决问题的公司（如RedHat、SuSE和VA Linux）不仅已成为商业巨人，而且整体上将Linux和开放式资源定位成一个今天每一个公司都可以接受的方案。这些公司很幸运，保持了开放资源的精神及与这一群体密切联系。开放资源式发展模型和坚定的商业支持，使Linux具有无限潜力。我确信，支持PHP的商业公司（例如Zend，Technologies）将会帮助PHP成为更加广泛使用的解决方案，特别是在高级终端网站中。

我想借此机会感谢IBM Haifa的Michael Rodeh教授，他鼓励Andi和我与Rasmus Lerdorf（PHP/FI1.0和2.0的作者）合作。Rasmus lerdorf也很愿意同我们合作，使PHP 3.0成为PHP/FI 2.0的正式新版本。在此我们也向他表示感谢。同样需要致谢的是，PHP发展群体和所有PHP开发者，

没有他们，PHP将不可能成为今天这样出色工具。最后，我们还要感谢PHP使用者，他们是创意和支持的无尽源泉。

我敢肯定你会发现这本书在你学习高级PHP和Web应用时非常有帮助。这本书不仅介绍一种语言的句法，而且还将介绍语言背后的思想，并能帮助你提高在Web编程时解决问题的能力。

祝你好运！

Zeev Suraski

前　言

当开放资源式软件如Linux或Apache的成功已经被主要媒体广泛记载、传播的时候，PHP的发展却在很大程度上被人们忽视。根据E-Soft的调查（WWW.e-Softinc.com/survey/），Web脚本语言PHP仍是ApacheWeb服务器最流行的模块。Netcraft研究发现：全球有超过6%的Web区域使用PHP（参见www.netcraft.com/survey）。对一个非专业化的产品来说，这是一个不可思议的市场渗透。并且，它的使用指数还在急剧增长。这一点在传统媒体中越来越多地被反应出来。自从2000年5月起，20种以上关于PHP的著作已用不同语言出版，同时还有更多的书在撰写中。

商业运作者正开始加入开发PHP潮流中：PHP被包含在Web服务器中，如：C2's Sronghold和Linux distributions。一个新的公司——Lend Technologies——已经形成，它提供PHP的商业附件，并支持许多大型Web站点和数以万计的中小型的Web站点开始使用PHP。

本书于1999年开始撰写，当时我们受New Riders Publishing之约写一本关于高级PHP的书。其实，写一本关于PHP的书的想法在我们脑中已存在了很久，所以我们很高兴地接受了提议。

在完成许多艰辛的工作以后，这本书终于与大家见面了，我们认为我们成功地写了一本不同于纯粹参考资料的书。我们尽力解释Web应用程序开发的思想，而不是仅仅给读者一个PHP特征的枯燥概述。

从一个没受过多少正规教育的编程新手发展到一个软件开发专家，要经过不同的阶段。程序员开始其职业时是一个新手，在这一阶段，他一般不关心代码风格、规划或调试。这时经常出现代码的可读性差、安全性丧失等问题。当程序员了解到了一门语言所有的诀窍和特征后，在团体开发、维护更大的开发项目时还会遭遇到困难，在这时，他们开始问如下问题：

- 我如何避免反复地执行同一个函数？
- 我必须做出什么规定才能使我的应用程序安全、稳定？
- 如何才能使我的应用程序更容易维护。
- 如何使很多人高效地在一个团体中合作。

这就是本书入手的地方，我们希望给软件开发者提供一些关于更好的PHP和Web应用软件开发的指导。许多技术今天已经可以获得，但只有当你理解隐藏在开发过程背后的基本原则，并且练就解决问题的技巧时才能充分利用它们。一般的参考手册并不提供这方面的帮助。

本书所面向的读者

如果你是编程新手，这本书并不适合你，本书适合以下人员：

- 你已进行过PHP应用开发工作，并想把你的能力提到一个新的水平。
- 你有其它编程语言的经验，并想用PHP进行Web应用开发。
- 你是一个PHP专家，并想在你个人基础上扩充PHP的特征。

你不必是一个PHP能手才读这本书，但你应该熟悉PHP的句法并有较丰富的程序规则的

知识。

设备需求

这本书认为你有一个正在工作的PHP设置，最好是PHP4.0或更新版本，鉴于其流行性，我们在必要时使用MySQL作为系统。由于平台独立性是PHP最大的特征之一，我们的例子将在UNIX和Windows下运行。

本书的结构

这本书分三部分，第一部分“高级PHP”，涵盖了PHP的高级语法体系；例如：目标定向、动态函数和变量、自修正代码等。这一部分同时给你一个项目策划原则、编码风格和应用程序设计的概述，这一部分指出了Web应用程序的快速、高效开发的必要基础。

第二部分“Web应用”，主要集中阐述软件构造：解释了为什么“会话”是重要的，什么样的安全性指导方针你必须牢记心中，为什么可用性至关重要，以及如何使用PHPLib来进行“会话”管理和数据库访问。你会在本书中学到三个成功的PHP项目案例研究，这会帮助你获得IT管理者的信服。

本书第三部分“深入研究PHP”，是为那些要超越目前PHP可得内容的人而设置的，它解释了如何用C扩充PHP。这是有关PHP扩充、发展的文件证据，已获Zend Technologies批准。

各章的具体内容如下：

第1章——开发思想

要处理高级项目，就必须使用代码规范、适当的策划和高级句法。这一章包含了一般的代码规范，这是对所有工业质量项目的要求，如：命名和评价规范以及如何把源代码拆成逻辑模块。

第2章——高级语法

这一章讲述了PHP的高级语法，例如，多维数组、变化变量、自修正代码等等。

第3章——应用程序设计：一个实际的例子

在这一章中，我们介绍一个完整的Web应用程序设计的全过程。这个应用程序是phpChat，一种Web聊天的IRC接口。这一章将展示设计的基本原则，给出关于项目组织的指导方针，并展示如何实现模块化的、插件应用的应用程序设计。

第4章——Web应用程序思想

会话管理、安全性考虑和授权、实用性，是每一个Web应用程序的基础，没有适当的“会话”管理，Web应用是不可能的。如果你想把变量联系起来（就像购物车将某个特定用户联系起来一样），那么你必须找到一种方式，在许多页的请求间识别用户变量。如果你不想让一个用户看到

另外一个用户的信用卡信息的话，这种识别最好是安全的。事实上，你必须进行特殊的考虑，以提高应用的安全性，虽然PHP比其他CGI环境更不易受黑客的攻击，但是当你并没有在头脑中想着本章介绍的某些重要规则时，你也很容易写出完全暴露式的应用程序。

这一章也介绍了一些基本的实用性概念，只要我们开始谈论应用程序而非独立脚本的时候，用户的地位变得更加重要，毕竟是用户来最终决定一个项目的成败。这一章也将向你介绍一些指导性方针，以使用户更满意。

第5章——基本网络应用程序策略

这一章讨论Web应用程序更多的基础内容，所有的Web应用程序都处理如表格输入或布局分隔等问题。从这些问题出发，这一章也将通过CVS版本控制的概述向你介绍高效的团体开发。最后，本章从PHP的观点讨论多层应用程序，如COM和Java。

第6章——PHP数据库访问

没有数据库做后盾，Web应用是不可能的，第6章提供PHPLib作为供方独立数据库访问的工具，并且概述了它的其他特征，例如："会话"管理、用户授权和许可管理。

第7章——尖端应用程序

通过使用PHPLib开发一个完整的信息库。这一章将使你熟悉PHPLib的模板类、SQL中的自参考和其他高级主题。然后，这一章陈述了XML的概况以及如何才能使应用程序从这一令人兴奋的技术中获益。这一章也描述了PHP用于XML分析及其WDDX函数的接口。

第8章——案例研究

当把一种新技术引入一个公司环境中时，成功的例子会有巨大的帮助。在第8章中，我提供一个案例研究来描述成功使用PHP的数百家公司中的三个较显著的例子：Six Open Systems、BizChek 和Marketplayer.

第9章——扩充PHP 4.0：探究PHP内核

1200个函数仍不够你用吗？没问题。因为这一章是关于扩充PHP的官方记录，如果你知道一些C的知识，第9章会给你一些压缩的关于PHP 4.0内部构件的知识，并告诉你如何编写自己的模块来扩充PHP的功能。

目　录

第一部分 高级 PHP

第1章 开发思想

命名是所有事的开始。

要真正掌握一门编程语言，不仅要理解它的语法和语义，更重要的是掌握语言所体现的哲学思想、语言产生和发展的背景以及设计特点。

1.1 PHP与我

大家是否想过，为什么会有这么多的编程语言？除了所谓“主流语言”，例如C、C++、Pascal等之外，还有其他的如Logol、Cobol、Fortran、Simula和许多更加特殊的语言。当列出一个项目的梗概时，大多数软件开发者不会真正地考虑到可以使用多种编程语言；他们都有自己偏爱的语言（也许是公司指定的一种语言），了解它的优点和它的缺点，并根据语言的具体特点修正项目。但当克服所选语言的缺陷时，就可能会增加不必要的额外工作。

了解如何使用一门语言却缺乏其特定的概念知识，就好像一个开卡车的人想参加二轮马车比赛一样，当然，一般来讲他应该懂得如何驾驶二轮马车，他甚至可能在终点线上跻身前列，但他绝不可能成为一个出色的车手，除非他熟悉新车的独特之处。

类似地，当面向对象程序设计（oop）程序员编写一个应用程序的时候，他会尽力使程序满足项目要求，处理同一个任务，不同的程序员会运用不同的方式。哪种方式更好？每一个程序员会说他（她）的方法最好，但只有那些熟悉两种概念——oop和过程化编程——的人能够作出判断。

前面提到的每一种语言代表一种解决问题的特定方法，这些问题多属于具有特殊要求的某一特殊种类。因为这些语言集中在一个有限的应用领域内，他们的成功性也限制在这些领域。像C和Pascal这样的语言变得如此流行，就是因为它们被广泛应用，并且它们不针对特殊问题，却提供了能很好地解决普遍问题的工具。

那么PHP是如何适应这一体系的呢？尽管它被称之为一种语言，但PHP并不是一种真正独立的语言，而是许多语言的混和体。它主要用C的句法，但与C有很大不同。它是被解释的，PHP能识别不同的变量类型，但没有严格的类型检查，PHP识别类，但没有结构体类型，类似的例子很多，但你可能已领会到了关键一点：PHP融合了许多种不同的解决问题的思想，形成了一种全新的、独一无二的方法。

为了能够用PHP成功地开发Web应用程序，我们鼓励你首先回答下述问题：PHP是我的项目所需的理想语言吗？问得好。如果我们说不，那我们就会显得很愚笨（谁会去写一本关于他们

认为不好的东西的书呢？）。让我们重新阐述这个问题，对项目来说有比PHP更好的语言吗？这次我们可以很有把握地回答，如果你正在从事网络应用程序的开发，PHP就是为你准备的最好的语言。

1.2 计划的重要性

你为什么应该阅读这一部分

即使你是一个很熟悉PHP的职业程序员，我们也建议你阅读下面的部分，因为这里包含了成功开发的基本知识，如果你对所讨论的题目已很熟悉，也应该花时间浏览一下，你可能会发现新的信息——新的题观点、新的解决方法、新的答案，你对解决未来项目的不同方面的问题了解得越多，你就能越好地抓住关键点，并且用更好的方式处理。我们希望你信任我们是职业开发者，并相信我们的经验，这将使你在以后受益。

在深入探讨PHP特定问题之前，先让我们从一个更广泛的观点开始。不论你使用什么语言，也不论你在什么平台上开发。有一些问题在应用开发中是总会涉及到的。

当从事一个专业项目的时候，考虑一下你正在做什么是至关重要的，“了解你的敌人，永远不要低估它”。尽管你的项目并不是一个真正的敌人，这句话的寓意仍然适用，在转向其他题目时，要知道项目的所有技术条件、目标平台、用户，并且决不要低估那些没有考虑周全的小问题的重要性。

据我们的经验，计划占用了50%的开发时间。项目越大，它的纲要就应该越详尽。这一原则既适用于同你的顾客相联系并与他们密切合作以确定一个总的项目概要，又适用于与你的开发者探讨确定一个编码概要。在一致性和可维护性上花的气力越少，就越容易在重新打开旧文件并设法清除错误或添加新的特征时遇到问题。

计划所用时间与项目大小并不一定成比例，例如，想一下要设计的一个搜索算法。这一应用程序只需要在一堆信息中进行基本的，搜索并能根据规则抽取数据，由于数据已经存在，所以创建和输出将不会需要太多的努力。这一应用程序将把它的大部分运行时间花在搜索循环上。这个循环也许用不了100行代码，但是为一个优化的循环选择设计一个优化的算法很容易耗费一整天的时间，这个小小的循环也许是设计阶段最庞大的部分，但另一方面，你可以在不到一天的时间内策划好数千行的代码。

同样，我们假定需要一个小脚本来列出某个目录中的所有文件，你能够很快地完成它，使其能从事某一特定任务，在一个特定的目录列出所有文件，不必再担心它了——问题已解决，可以转向其他任务，把你的程序抛在脑后。但另外一种策略是考虑一下以后的某个时间，甚至可能是在一个完全不同的项目中——你可能会再一次需要一种类似的工具，仅仅一遍又一遍地重做目录列举器，每一个对应一个特定的任务，这简直是在浪费时间。因此，当首次遇到这种情况时，应该考虑到这一点，应从一个目录列举器中创建一个分离的模块，允许它列举不同的目录，有选择性地递推子目录，甚至允许使用通配符，你可以创建一个“防弹”函数，它即能处理大多数特例，又能完美地应付一个目录列举器的普通要求。采用这种策略经过几个项目之后，你将拥有一个工具参数的库，可以安全地重新使用和依赖这个库，从而可以极大地减省开发时间。

当然，有了一个日益增大的免费工具函数库，依然不能满足全部需要，也不能优化这个库以适应特殊需求，有些库太庞大以致不能随处安装，因为每一次选中都必须分析几百K字节的代码，这将严重降低站点的性能。在这种情况下，需要用100%自己创造的优化解决方案，以取代非最优解决方案。

更大的项目如果缺乏计划将导致更多的错误，在开发后期，可能会遇到没有或无法预见的困难，这是由于缺乏计划的时间和工作，这些困难可能会严重到让你彻底地重组整个项目。例如，对一个依赖额外数据库提取层的数据库支持的应用程序，其数据库提取层仅能接收文本数据，但后来你发现也需要用它接收数值性的数据，通过工作区转换，可以使它能够接收数值性数据。但后来你又感觉到这个工作区仍旧不能满足需要，这时唯一能做的就是改变数据库接口，这需要重构提取层并对所有主代码调用进行检查，当然也需要清除先前创建的工作区。

这样，数小时甚至整天的工作将不得不耗费在本来从一开始就可以避免的问题上，这些问题往往决定了程序开发的成败，因为“时间是你永远都不可能充分拥有的珍贵资源”。下面的内容将针对大部分基本的却是非常重要的开发中的实际问题进行讨论：改善代码质量以及基本设计和文件管理的问题。陈述完这些后，我们创建一个应用程序接口（API），采取简单的、实用的方式使你熟悉这一新的思想，然后我们从头创建一个API，在纸上从理论上开发它，并明确一些实用规则来帮助你实施下一个API，例如风格问题以及商业技巧等。

1.3 编码规范

好的编码和差的编码之间究竟有何区别呢？实际上，这个问题很简单。好的代码（确实好的代码）能够像一本书一样被阅读。你能从任何地方读起，并且能够时刻意识到你所读的这些行是干什么用的，它们在什么条件下执行，它们所要求的设置。即使你缺乏背景知识，遇到了一个错综复杂的算法，你也能很快看出它所从事的任务，以及它的风格。

举个例子，然后说“照着做”总是很容易的，但我想这一章应该使你打下写专业化代码的坚固基础，这一基础将区分真正精心编制的代码和一个草草完成的程序段。抱歉的是，由于篇幅所限，我们不能按我们所希望的那样详尽地讨论良好的代码书写风格的每一方面，但本章将给你一个很好的开始。我们期望你能迅速获得专用的材料，以熟悉软件设计和工程的每一要点。编码是一个很广的领域，几乎是一门独立的科学。有许多论文论述它，虽然这些论文大多很乏味，很理论化，但在应用中是不可放弃的。下面我们就最重要的问题进行最基本的讨论。

1.3.1 选择名字

选择变量名可能是程序员最常做、但却想得最少的。如果你已建立了这些在大项目中出现的变量名字、类型、定义位置的清单，那么你就创建了一个类似于小电话簿的东西，你想让你的清单成为什么样子呢？不同的命名方案已发展起来了，它们有不同的思想及各自的优点和缺点，这些方案一般分为两类：简短的变量和函数名及谈话式的变量和函数名（描述变量类型和目的的更长的名字）。

某个电话目录可能是这个样子的，如表1-1所示。

表1-1　电话目录

姓　名	地　址	电　话
J.D.	382W.S	-3951
M.S.	204E.R.	-8382

这份列表非常有意思：该列表有两个条目，但并没有更多的信息。人名只有首字母，没有全称；只有房间号，但没有街道名；只有电话号码的一部分，却没有完整的号码。

让我们看另外一个例子，如表1-2所示。

表1-2　电话目录

姓　名	地　址	电　话
ht5ft9in_age32_John Doe_male_married	386 West Street, Los Angeles, California, USA, Earth	+1-555-304-3951
ht5ft6in_age27_Mary Smith_female_single	204 East Road, Los Angeles, California, USA, Earth	+1-555-306-8382

在这个例子中，每个人的名字包括身高、年龄、性别及婚姻状况。地址中不但包括街道和城市，而且也包括州、国家、甚至星球。电话号码附加了国家和地区号。

第二种解决方案比第一种好吗？两个都不是最好的。在程序课上讲授的这两种解决方案，都不令人满意，定义一种类型tpIntMyIntegerCounter，然后声明一个变量instMYInteger CyunterInstance。如果仅仅需要遍历一个数组并将所有元素都设为0，这无疑显得太冗长了（见清单1-1）。

清单1-1　一个过于冗长的实例

```
for ( $instMyIntegerCounterInstance = 0;
      $instMyIntegerCounterInstance < MAXTPINTEGERCOUNTERRANGE;
      $instMyIntegerCounterInstance++)
         $instMyArrayInstance[$instMyCounterInstance] = 0;
```

另一方面，使用I、j、k（而不是像$instMyIntegerCounterInstance这样长的名字）也是不可接受的，尤其当我们从事的是像压缩这样复杂的缓冲操作的时候更是如此。

这只是普遍思想被误用的一个简单例子，该怎么办？解决的办法是选择好的整体思想，然后在适当的地方加以例外处理，当写一个应用程序时，应该知道你的代码从事的是什么工作，能够快速地从一点转到另一点——但其他人可能认为这并不容易。如果你从开发组的某个人手中获得一个源文件并需要添加一些特征，首先必须对其进行整体把握，并区分代码的各个部分。理想情况下，这一过程将和阅读源文件平行进行，但由于在没有提示和公共样本帮你理清代码来阅读的情况下，这是不可能做到的，所以在源代码中包含尽可能多的额外信息，并且使得明显的事实不易于混淆就显得很重要了。

那么如何能查知这些信息，并将其合并入自己的代码呢？

- 使代码更易读。