

## <<.NET软件架构之美>>

### 图书基本信息

书名：<<.NET软件架构之美>>

13位ISBN编号：9787115200181

10位ISBN编号：7115200181

出版时间：2009年9

出版时间：人民邮电出版社

作者：Dino Esposito, Andrea Saltarello

页数：432

字数：547000

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## <<.NET软件架构之美>>

### 内容概要

本书出自两位具有多年软件开发经验的 ASP .NET 专家、作者和培训师之手，内容涉及多层架构、设计模式以及设计原则。

第一部分简要介绍 UML、设计原则及模式；第二部分从技术架构角度讨论分层设计。

本书行文流畅，语言通俗易懂，阐述了各种架构设计技术方案的优与劣，并讲述了如何在优与劣中做出权衡。

中设计了真实的场景，展示了如何将这些设计原则更加具体地应用到 .NET 应用程序中。

本书适合各层次 .NET 开发人员阅读。

## <<.NET软件架构之美>>

### 作者简介

作者：(意大利)埃斯波西托(Dino Esposito) (意大利)索尔塔雷罗(Andrea Saltarello)DinoEsposito，.NET和软件架构技术方面的世界级权威，微软ASP.NET MVP。

目前就职于著名的.NET技术咨询公司IDesign。

他是广受欢迎的技术作家。

担任MSDN Magazine特邀专栏作家多年。

并撰有Programming ASP.NET-3.5 Core References等名著。

Andrea Saltarello，微软ASP.NET MVP。

意大利.NET用户组负责人。

现任Managed Designs公司首席软件架构师。

<<.NET软件架构之美>>

书籍目录

Part I Principles	1 Architects and Architecture Today	What's a Software Architecture, Anyway?
Applying Architectural Principles to Software	Architecture Is About Decisions	What's Architecture and What's Not
Requirements and Quality of Software	Who's the Architect, Anyway?	An Architect's Responsibilities
How Many Types of Architects Do You Know?	Common Misconceptions About Architects	Overview of the Software Development Process
The Software Life Cycle	Models for Software Development	Summary
Murphy's Laws of the Chapter	2 UML Essentials	UML at a Glance
Motivation for and History of Modeling Languages	UML Modes and Usage	UML Diagrams
Use-Case Diagrams	Class Diagrams	Sequence Diagrams
Summary	Design Principles and Patterns	Murphy's Laws of the Chapter
3 For What the Alarm Bell Should Ring	Structured Design	Separation of Concerns
Object-Oriented Design	Principles	Advanced Principles
Basic OOD	From Principles to Patterns	What's a Pattern, Anyway?
Patterns vs. Idioms	Dependency Injection	Applying Requirements by Design
Testability	Security	From Objects to Aspects
Aspect-Oriented Programming	AOP in Action	Summary
Murphy's Laws of the Chapter	The Business Layer	What's the Business Logic Layer, Anyway?
Part II Design of the System	Where Would You Fit the BLL?	Business and Other Layers
4 Dissecting the Business Layer	Business Layer	The Transaction Script Pattern
The	Pattern in Action	The Table Module Pattern
The TM	Pattern in Action	The Active Record Pattern
The AR	Pattern in Action	The Domain Model Pattern
The DM	Pattern in Action	Summary
5 The Service Layer	What's the Service Layer, Anyway?	Responsibilities of the Service Layer
What's a Service, Anyway?	Services in the Service Layer	The Service Layer Pattern in Action
Generalities	of the Service Layer Pattern	The Service Layer Pattern in Action
The	Remote Façade Pattern	The Data Transfer Object Pattern
DTO	vs. Assembly	Service-Oriented Architecture
What SOA Is Not	SOA and the Service Layer	The Very Special Case of Rich Web Front Ends
Refactoring the Service Layer	Designing an AJAX Service Layer	Securing the AJAX Service Layer
6 The Data Access Layer	Summary	Murphy's Laws of the Chapter
What's the Data Access Layer, Anyway?	Functional Requirements of the Data Access Layer	Responsibilities of the Data Access Layer
Designing Your Own Data Access Layer	Access Layer	The Data Access Layer and Other Layers
The Inversion of Control Pattern	The Contract of the DAL	The Plugin Pattern
Crafting Your Own Data Access Layer	Laying the Groundwork for a Data Context	Implementing the Persistence Layer
Implementing	Transactional Semantics	Implementing Query Services
Implementing	Concurrency	Implementing Uniquing and Identity Maps
Power to the DAL with an O/RM Tool	Object/Relational Mappers	Using an O/RM Tool to Build a DAL
To SP or Not to SP	About Myths and Stored Procedures	What About Dynamic SQL?
Summary	Laws of the Chapter	7 The Presentation Layer
Murphy's	Responsibilities of the Presentation Layer	Responsibilities of the User Interface
Common	Pitfalls of a Presentation Layer	Evolution of the Presentation Patterns
The	Model-View-Controller Pattern	The Model-View-Presenter Pattern
The Presentation Model	Pattern	Choosing a Pattern for the User Interface
What Data Is Displayed in the View?	Processing User Actions	Design of the Presentation
Idiomatic Presentation Design		MVP

<<.NET软件架构之美>>

[in Web Presentations](#)

[MVP in Windows Presentations](#)

[Summary](#)

[Murphy's Laws of the](#)

[Chapter](#)

[Final Thoughts](#)

[Appendix: The Northwind Starter Kit](#)

[Index](#)

## 章节摘录

插图：To design a system——any system in any scientific field——you first need to create an abstraction of it. An abstraction is essentially a model that provides a conceptual representation of the system in terms of views, structure, behavior, participating entities, and processes. A model exists to be shared among the stakeholders of the system, including developers, architects, owners, and customers. Stakeholders should be able to understand the model in order to provide feedback, spot wrong assumptions, and suggest improvements. To share a model, though, you need to express it in a formal way using a common, and possibly broadly accepted, notation. For this, you need a modeling language. Typically, a modeling language is a graphical or textual language based on a set of rules, symbols, diagrams, and keywords. All together, the language elements are used to express the structure and behavior of the model in a way that transmits clear knowledge and information to anybody familiar with the language. There are several well-known examples of modeling languages——for example the Integrated Definition ( IDEF ) family of modeling languages used for functional modeling, information modeling, simulation, and more. There's also the Virtual Reality Modeling Language ( VRML ) , which is used to represent 3D graphics, and EXPRESS ( textual ) and EXPRESS-G ( graphical ) for data modeling. However, when it comes to modeling languages, the most popular one is Unified Modeling Language ( UML ) . UML is a general-purpose graphical modeling language that, over the years, has become the industry standard for modeling software systems. Based on a family of graphical notations, UML is particularly suited to creating models in object-oriented scenarios. It might not be as effective when another paradigm is used, such as a functional or perhaps a relational paradigm, but it's a good fit for object-oriented systems.

## <<.NET软件架构之美>>

### 媒体关注与评论

“所有架构师的必读之作……无可替代。

”——.NET Developer's Journal “还等什么？

如果你有机会看到本书，请尽快把它‘消灭’。

就像我在地铁上如饥似渴地畅读一样……”——王涛（AnyTao）。

微软MVP“本书酣畅淋漓地阐发了.NET平台下企业软件架构的精髓。

为开发人员献上了不可多得的饕餮大餐。

”——陈黎夫（Dflying），微软MVP

## <<.NET软件架构之美>>

### 编辑推荐

《.NET软件架构之美(英文版)》填补了这一缺憾。

两位作者人选可谓众望所归，他们将GoF设计模式、MartinFowler企业架构模式、EricEvans领域驱动设计等业界精华与自己多年软件开发实战经验结合起来，深刻阐述了软件架构设计思想精髓。

作者还从技术架构角度逐章讲述了业务层、服务层、数据访问层和表现层的分层设计，同时介绍了各种软件架构设计方案的优劣，如何在各种方案中做出抉择，以及如何将这些设计原则更具体地用到应用程序中。

软件架构设计是现代软件开发的核心，它不仅是一门技术，更是一门艺术。

然而，长期以来，还没有一本专门讲述.NET架构设计的书。

Amazon全五星图书，紧贴实战，透过实例探讨架构设计最佳实践，深刻阐述软件开发思想。



## <<.NET软件架构之美>>

### 版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>