

<<ASP.NET企业级架构开发技>>

图书基本信息

书名：<<ASP.NET企业级架构开发技术与案例教程>>

13位ISBN编号：9787111393481

10位ISBN编号：7111393481

出版时间：2012-8

出版时间：机械工业出版社

作者：杨树林，胡洁萍 编著

页数：339

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>

## <<ASP.NET企业级架构开发技>>

### 内容概要

《asp.net企业级架构开发技术与案例教程》将带领读者进入asp.net技术世界，重点讲解企业级架构开发技术，由浅入深地学习各项知识。  
全书共分9章，主要讲解asp.net基础、web窗体技术与用户界面设计、数据访问层与业务逻辑层实现技术、数据控件与视图层实现技术、应用其他常用技术完善系统、asp.net mvc框架、持久化技术nhibernate、集成框架spring.net、博客系统的设计与实现。  
每章内容都与案例相结合，有助于学生理解知识、应用知识，达到学以致用目的。

《asp.net企业级架构开发技术与案例教程》内容丰富，实例典型，知识讲解系统，适合作为大中专院校计算机及相关专业的教材或参考书，也适合作为软件开发人员及其他有关人员的技术参考书。

书籍目录

出版者的话

编委会

丛书序言

前言

教学建议

第1章 asp.net基础

1.1 asp.net概述

1.1.1 .net框架简介

1.1.2 asp.net及其特点

1.1.3 asp.net 4.0新特征

1.2 集成开发环境的安装与使用

1.2.1 安装vs 2

1.2.2 集成开发环境的使用

1.3 分层架构与系统分析

1.3.1 分层架构模式

1.3.2 asp.net中常用的分层结构

1.3.3 系统分析与建模

1.3.4 案例1-1：企业信息展示系统的分析与设计

1.4 asp.net web程序结构

1.4.1 asp.net web应用程序的构成

1.4.2 asp.net web应用程序配置

1.4.3 案例1-2：建立企业信息展示系统的程序结构

1.5 c#语言基础

1.5.1 c#语言的基本知识

1.5.2 c#面向对象编程

1.5.3 案例1-3：企业信息展示系统的实体模型类及接口设计

1.5.4 c#的新特征

本章小结

习题与实验

第2章 web窗体技术与用户界面设计

2.1 web窗体技术基础

2.1.1 web窗体及其组成

2.1.2 web服务器控件

2.1.3 验证控件

2.1.4 案例2-1：实现企业信息展示系统的用户注册界面

2.2 应用主题与母版

2.2.1 应用主题

2.2.2 案例2-2：企业信息展示系统的主题设计

2.2.3 应用母版

2.2.4 案例2-3：企业信息展示系统的母版设计

2.3 用户控件设计

2.3.1 用户控件简介

2.3.2 用户控件的创建

2.3.3 用户控件的使用

2.3.4 案例2-4：企业信息展示系统的用户控件设计

## <<ASP.NET企业级架构开发技>>

### 2.4 网站地图与页面导航

#### 2.4.1 网站地图

#### 2.4.2 使用导航地图实现网站导航

#### 2.4.3 案例2-5：企业信息展示系统的 站点导航设计

#### 本章小结

#### 习题与实验

### 第3章 数据访问层与业务逻辑层实现技术

#### 3.1 数据库访问基础

##### 3.1.1 ado.net简介

##### 3.1.2 数据库的连接

##### 3.1.3 案例3-1：连接企业信息展示系统数据库

##### 3.1.4 数据更新与数据查询

##### 3.1.5 案例3-2：企业信息展示系统的数据库操作辅助类设计

#### 3.2 数据绑定与两层应用结构实现

##### 3.2.1 数据绑定简介

##### 3.2.2 数据源控件

##### 3.2.3 两层应用结构的实现方案

##### 3.2.4 案例3-3：企业信息展示系统的数据访问层实现

#### 3.3 三层应用结构

##### 3.3.1 业务逻辑层的作用

##### 3.3.2 三层应用结构的实现方案

##### 3.3.3 工厂模式与工厂类设计

##### 3.3.4 案例3-4：企业信息展示系统的业务逻辑层实现

#### 本章小结

#### 习题与实验

### 第4章 数据控件与视图层实现技术

#### 4.1 数据控件与gridview控件

##### 4.1.1 数据控件概述

##### 4.1.2 gridview控件

##### 4.1.3 案例4-1：实现企业信息展示系统中的产品管理

#### 4.2 detailsview与formview控件

##### 4.2.1 detailsview控件

##### 4.2.2 formview控件

##### 4.2.3 案例4-2：实现企业信息展示系统中的产品的显示、添加和修改

#### 4.3 datalist与repeater控件

##### 4.3.1 datalist控件

##### 4.3.2 repeater控件

##### 4.3.3 案例4-3：实现企业信息展示系统中的产品分类显示

#### 4.4 listview与datapager控件

##### 4.4.1 listview控件

##### 4.4.2 datapager控件

##### 4.4.3 案例4-4：实现企业信息展示系统的主界面

#### 4.5 asp.net ajax

##### 4.5.1 asp.net ajax概述

##### 4.5.2 创建ajax应用程序

##### 4.5.3 案例4-5：在企业信息展示系统中使用asp.net ajax

## <<ASP.NET企业级架构开发技>>

本章小结

习题与实验

第5章 应用其他常用技术完善系统

5.1 asp.net状态管理

5.1.1 视图状态和控件状态

5.1.2 案例5-1：改进企业信息展示系统中的产品分类控件

5.1.3 隐藏域与查询字符串

5.1.4 cookie及其应用

5.1.5 案例5-2：完善企业信息展示系统的用户登录控件

5.1.6 应用程序状态和会话状态

5.1.7 案例5-3：实现企业信息展示系统中的用户统计

5.2 成员管理

5.2.1 验证方式及其配置

5.2.2 成员管理api 及其配置

5.2.3 使用网站管理工具管理角色和用户

5.2.4 asp.net登录控件

5.2.5 案例5-4：基于成员管理实现用户管理

5.3 数据缓存

5.3.1 缓存概述

5.3.2 页输出缓存

5.3.3 使用应用程序缓存

5.3.4 缓存依赖

5.3.5 案例5-5：在企业信息展示系统中应用缓存技术

本章小结

习题与实验

第6章 asp.net mvc框架

6.1 asp.net mvc概述

6.1.1 mvc模式

6.1.2 asp.net mvc

6.1.3 asp.net mvc程序结构

6.1.4 案例6-1：按mvc模式设计企业信息展示系统

6.2 url路由

6.2.1 url模式

6.2.2 默认路由

6.2.3 添加路由

6.2.4 创建路由约束

6.3 控制器

6.3.1 控制器类

6.3.2 操作方法

6.3.3 案例6-2：企业信息展示系统的路由及控制器设计

6.4 过滤器

6.4.1 mvc过滤器概述

6.4.2 几个内置的过滤器

6.4.3 自定义过滤器

6.4.4 案例6-3：实现企业信息展示系统的异常处理和权限验证

6.5 asp.net mvc应用程序中的传递数据

6.5.1 控制器向视图传递数据

## <<ASP.NET企业级架构开发技>>

- 6.5.2 在操作方法之间传递状态
- 6.5.3 视图向控制器传递数据
- 6.6 asp.mvc视图与htmlhelper
  - 6.6.1 概述
  - 6.6.2 aspx ( c# ) 引擎视图
  - 6.6.3 用于呈现视图的帮助器 ( htmlhelper )
  - 6.6.4 扩展帮助器方法
  - 6.6.5 razor ( cshtml ) 引擎视图
  - 6.6.6 案例6-4 : 企业信息展示系统的布局页设计
- 6.7 webgrid
  - 6.7.1 webgrid概述
  - 6.7.2 webgrid的使用
  - 6.7.3 案例6-5 : 实现企业信息展示系统的视图
- 本章小结
- 习题与实验
- 第7章 持久化技术nhibernate
  - 7.1 nhibernate原理
    - 7.1.1 nhibernate简介
    - 7.1.2 在项目中引用nhibernate
    - 7.1.3 配置nhibernate
    - 7.1.4 创建实体类及其映射
    - 7.1.5 利用nhibernate api访问数据库
  - 7.2 nhibernate的实体映射
    - 7.2.1 实体映射基础
    - 7.2.2 实体关系映射
    - 7.2.3 案例7-1 : 企业信息展示系统的实体类及其映射设计
  - 7.3 实体操作与数据查询
    - 7.3.1 实体操作
    - 7.3.2 数据查询
    - 7.3.3 辅助类设计
    - 7.3.4 案例7-2 : 基于nhibernate的企业信息展示系统的dal设计
- 本章小结
- 习题与实验
- 第8章 集成框架spring.net
  - 8.1 spring.net基础
    - 8.1.1 spring.net框架
    - 8.1.2 spring.net控制反转
    - 8.1.3 对象的配置与对象factory
    - 8.1.4 案例8-1 : 在企业信息展示系统中使用spring.net
  - 8.2 spring.net的aop
    - 8.2.1 aop的概念
    - 8.2.2 使用spring.net中的aop
  - 8.3 spring.net集成其他框架
    - 8.3.1 spring.net与nhibernate集成
    - 8.3.2 spring.net与asp.net mvc集成
    - 8.3.3 案例8-2 : 基于msn架构的企业信息展示系统配置
  - 8.4 spring事务管理与任务调度

## <<ASP.NET企业级架构开发技>>

8.4.1 事务的基本配置

8.4.2 事务的传播属性

8.4.3 任务调度

本章小结

习题与实验

第9章 博客系统的设计与实现

9.1 系统分析与设计

9.1.1 系统分析

9.1.2 总体设计

9.1.3 实体类（数据模型）及映射设计

9.1.4 接口设计

9.1.5 分页辅助类设计

9.1.6 页面的整体布局

9.2 系统配置

9.2.1 spring.net 配置

9.2.2 spring.net与nhibernate.net整合配置

9.2.3 安全管理配置

9.3 数据访问层实现

9.3.1 数据访问层基类（ybbasedal）

9.3.2 文章分类数据访问类（ybcategorydal）

9.3.3 文章数据访问类（ybarticledal）

9.3.4 文章评论数据访问类（ybreviewdal）

9.3.5 相册数据访问类（ybpicturedal）

9.3.6 在对象配置文件中描述数据访问对象

9.4 业务逻辑层实现

9.4.1 文章分类业务逻辑类（ybcategorybll）

9.4.2 文章业务逻辑类（ybarticlebll）

9.4.3 文章评论业务逻辑类（ybreviewbll）

9.4.4 相册业务逻辑类（ybpicturebll）

9.4.5 用户业务逻辑类（ybuserbll）

9.4.6 在对象配置文件中描述业务逻辑对象

9.5 控制层实现

9.5.1 控制器基类（basecontroller）

9.5.2 主页控制类（homecontroller）

9.5.3 文章分类控制类（categorycontroller）

9.5.4 文章控制类（articlecontroller）

9.5.5 相册控制类（picturecontroller）

9.5.6 用户控制类（usercontroller）

9.5.7 asp.net mvc与spring.net集成

9.6 视图层设计

9.6.1 用户控件（分部页）设计

9.6.2 主页和关于页视图设计

9.6.3 文章分类视图设计

9.6.4 文章视图设计

9.6.5 用户视图设计

9.6.6 相册视图设计

本章小结

习题与实验  
参考文献



## 章节摘录

版权页：插图：5.3 数据缓存 ASP.NET提供了一些用于提升程序性能的技术特性，其中，缓存技术是非常重要的一个特性，它提供了一种非常好的本地数据缓存机制，从而有效地提高了数据访问的性能。

5.3.1 缓存概述 1.什么是缓存 缓存是将数据暂存于内存缓存区中（有时也暂存于硬盘缓存区中）的一种技术。

当数据本身改变得不很频繁，而被访问的频率又比较高时，采用这种技术可大大改进应用程序的性能。

缓存是由Cache类实现的；缓存实例是每个应用程序专用的。

缓存生存期依赖于应用程序的生存期；重新启动应用程序后，将重新创建Cache对象。

ASP.NET使用两种基本的缓存机制来提供缓存功能。

第一种机制是页输出缓存，它保存页处理输出，并在用户再次请求该页时，重用所保存的输出，而不是再次处理该页。

第二种机制是应用程序缓存，它允许缓存所生成的数据，如DataSet或业务对象。

（1）页输出缓存 页输出缓存在内存中存储处理后的ASP.NET页的内容。

这一机制允许ASP.NET向客户端发送页响应，而不必再次经过页处理生命周期。

页输出缓存对于那些不经常更改，但需要大量处理才能创建的页特别有用。

例如，如果创建大通信量的网页来显示不需要频繁更新的数据，页输出缓存则可以极大地提高该页的性能。

可以分别为每个页配置页缓存，也可以在Web.config文件中定义缓存配置，只定义一次缓存设置就可以在多个页中使用这些设置。

页输出缓存提供了两种页缓存模型：整页缓存和部分页缓存。

整页缓存允许将页的全部内容保存在内存中，并用于完成客户端请求。

部分页缓存允许缓存页的部分内容，其他部分则为动态内容。

部分页缓存可采用两种工作方式：控件缓存和缓存后替换。

控件缓存有时也称为分段缓存，这种方式允许将信息包含在一个用户控件内，然后将该用户控件标记为可缓存的，以此来缓存页输出的部分内容。

这一方式可缓存页中的特定内容，并不缓存整个页，因此每次都需重新创建整个页。

例如，如果要创建一个显示大量动态内容（如股票信息）的页，其中有些部分为静态内容（如每周总结），这时可以将静态部分放在用户控件中，并允许缓存这些内容。

缓存后替换与控件缓存正好相反。

它对页进行缓存，但是页中的某些片段是动态的，因此不会缓存这些片段。

例如，如果创建的页在设定的时间段内完全是静态的（如新闻报道页），可以设置为缓存整个页。

如果为缓存的页添加旋转广告横幅，则在页请求之间，广告横幅不会变化。

然而，使用缓存后替换，可以对页进行缓存，但可以将特定部分标记为不可缓存。

编辑推荐

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>