Practical...

ASP.NET

函数实用手册

Functional Handbook

张曜　张青　编著

冶金工业出版社

ASP.NET 函数实用手册

张曜　张青　编著

内容简介

本书介绍了开发 ASP.NET 的 Web 程序中常用的类和函数。主要内容包括系统架构，数据集合，Web 应用程序，Web 用户界面——Web Form，数据访问，多线程和文本处理。本书从各个功能方面给出了常用的类，以及类中方法、属性、异常的详细描述，并配合完整的示例程序，让读者更快地掌握其用法。

本书内容全面，深入浅出。提供功能模块、类、函数三级详尽参考，适合 ASP.NET 开发者和广大编程爱好者作为参考手册使用。

图书在版编目（CIP）数据

ASP.NET 函数实用手册 / 张曜等编著. —北京：冶金工业出版社，2002.11
ISBN 7-5024-3118-7

I. A… II. 张… III. 主页制作—程序设计—技术手册 IV. TP393.092-62

中国版本图书馆 CIP 数据核字（2002）第 075166 号

出版人 曹胜利（北京沙滩嵩祝院北巷 39 号，邮编 100009）
责任编辑 程志宏
中山市新华印刷厂有限公司印刷；冶金工业出版社发行；各地新华书店经销
2002 年 12 月第 1 版，2002 年 12 月第 1 次印刷
787mm×1092mm 1/16； 20 印张； 949 千字； 308 页； 1-2500 册
35.00 元
冶金工业出版社发行部 电话：（010）64044283 传真：（010）64027893
冶金书店 地址：北京东四西大街 46 号（100711） 电话：（010）65289081
（本社图书如有印装质量问题，本社发行部负责退换）

前　言

1. 关于 ASP.NET

自 2000 年 6 月微软公布.NET 计划以来，“.NET”就一直是 IT 业界的热门话题。微软在.NET 计划中不仅给我们带来了全新的思维，也陆续地围绕这种思维开发了许多崭新的技术。而 ASP.NET，就是.NET 计划中推出的激动人心的技术之一。ASP.NET 的推出，不仅很好地解决了 ASP 的遗留问题，而且提出了新一代 Web 应用程序的概念和框架。ASP.NET 最具新意的技术特点主要有 C#编程语言、Web Form 技术、Code-Behind 技术等等。全新的 C#语言解决了 ASP.NET 的运行效率的问题，而全面的基于.NET Framework Library 使得 ASP.NET 本身就相当于内置众多功能强大的组件，Web Form 技术体现了新一代 Web 应用程序的发展方向，它同 Code-Behind 技术相结合，使程序框架清晰可辨，不再跟页面代码混淆不清。同时，通过把代码预编译为动态连接库（*.dll）文件，很好地保护了源代码。由于 ASP.NET 众多的优点，因此，用它来取代 ASP 是理所当然的事。ASP.NET 被称为新一代的 ASP。

读者最好使用 Visual Studio.NET 集成工具来开发 ASP.NET 程序，该软件为开发者提供了最大的便利。当然，也可以使用任何文本编辑器来编辑 ASP.NET 源程序，但是.NET Framework 是运行 ASP.NET 程序必不可少的。

2. 本书的内容结构

本书涉及 ASP.NET 最实用的功能，包括系统架构、Web 应用程序、Web Form、数据集合、文本处理、数据访问、多线程等。本书共分 7 章，各章主要内容如下：

第 1 章：系统架构。本章介绍了整个系统的一些基础类的知识。包括 System.Object 根类、异常处理、类型转换、时间和日期、数学运算以及随机数发生器的相关类。

第 2 章：数据集合。本章介绍了.NET Framework 中的数据集合类。包括动态数组、散列表、栈、队列和有序表的相关类。

第 3 章：Web 应用程序。本章介绍了 Web 应用程序。包括 Application、Server、C/S 交互以及 Cookie 和 Session。

第 4 章：Web 用户界面——Web Form。本章介绍了 ASP.NET 最激动人心的技术之一 Web Form。包括 ASP.NET 页面及控件、数据绑定、HTML 服务器端控件及 Web 服务器端控件。

第 5 章：数据访问。本章介绍了连接和使用数据库的相关类。包括 System.Data.SqlClient.SqlCommand 类、System.Data.SqlClient.SqlConnection 类、System.Data.SqlClient.SqlDataAdapter 类、System.Data.SqlClient.SqlDataReader 类、System.Data.OleDb.OleDbCommand 类、System.Data.OleDb.OleDbConnection 类、System.Data.OleDb.OleDbDataAdapter 类、System.Data.OleDb.OleDbDataReader 类。

第 6 章：多线程。本章介绍了多线程机制。包括多线程的基本操作、线程的同步和互斥。

第 7 章：文本处理。本章介绍了字符串处理的实现。包括 System.String 和

System.Text.StringBuilder 类。

另外还在附录中介绍了 C#编译选项及类索引。

3. 本书特点

本书内容全面，条理清晰，按功能模块、类、函数分为三级目录，为读者查阅相关资料提供了方便。本书中同一个功能模块中的类是按照字母顺序排列的。在一个类的说明中，本书首先对类进行了描述，接着列出了参考属性或函数列表。本书给出了每一个函数的重载形式和参数说明，对函数的使用方法和使用时应注意的事项进行了说明，随后附带了示例代码。本书涵盖了开发 ASP.NET 的 Web 程序中常用的类和函数，是 ASP.NET 开发者难得的一本好参考书。

4. 本书适合对象

本书内容全面，深入浅出。不仅适合于 ASP.NET 的开发者，也可供广大编程爱好者参考。

本书由张曜、张青编著。此外，汪晓明参与编写了本书第 1、2、3、4、5 章，朱海波参与编写了第 6、7 章，全书由郭立山审校。本书中的源代码可以从网上下载，**网址：http://www.TianDing.net 或 http://www.cnbook.net**。如果读者在查阅本书过程中遇到疑难问题或觉得不妥之处，也可到该网站的相关论坛进行探讨。

虽然本书的作者具有多年的 ASP 开发经验，而且本书经过作者反复推敲、验证，但是 ASP.NET 推出不久，限于时间仓促，加之水平有限，错漏在所难免，恳请读者批评指正。

编 者

2002 年 10 月

目　录

第 1 章　系统架构

在.NET Framework Library 中，与系统架构相关的类都定义在命名空间（namespace）System 中。因此，使用其中的类时，都必须导入命名空间 System。在 ASP.NET 中，导入一个命名空间的方法是在页面顶部写入如下语句：

<% @Import NameSpace="System" %>

@ Import 指令所具有的 namespace 属性不能多于一个。要导入多个命名空间，必须使用多条 @Import 指令。默认情况下，系统将为所有的 ASP.NET 应用程序页面导入如下命名空间：

1）System。
2）System.Collections。
3）System.Collections.Specialized。
4）System.Configuration。
5）System.IO。
6）System.Text。
7）System.Text.RegularExpressions。
8）system.web。
9）System.Web.Caching。
10）System.Web.Security。
11）System.Web.SessionState。
12）System.Web.UI。
13）System.Web.UI.HtmlControls。
14）System.Web.UI.WebControls。

除此以外的所有命名空间，用户都必须手动导入。

本章介绍了系统架构类库中与 ASP.NET 密切相关的内容，具体包括以下内容：

1）System.Object 根类。
2）异常处理。
3）类型转换。
4）时间和日期处理。
5）数学运算。
6）随机数发生器。

1.1　System.Object 根类

System.Object 根类支持.NET 框架类层次结构中的所有类，并为派生类提供低级别服务。是.NET 框架中所有类的最终超类；是类型层次结构的根。

类名：System.Object。

声明：

public class Object

继承与被继承：

System.Object

派生类

描述：

System.Object 类是整个.NET Framework 的根类，其他所有的类都必须继承它。但是无需显示地派生，因为从 Object 类继承是隐式的，系统自动完成。它提供了一些最基本的操作，其中，下列操作可以由用户重写：

Equals——支持对象间的比较。

Finalize——在自动回收对象之前执行清理操作。

GetHashCode——生成一个与对象的值相对应的数字以支持哈希表的使用。

ToString——生成描述类的可读文本字符串。

属性：

该类是整个系统的基类，没有定义公共的或者保护的属性。

方法：

表 1-1 所示为 System.Object 类的方法。

表 1-1　System.Object 类的方法

属性	描述
Object	构造函数
Equals	已重载。确定两个 Object 实例是否相等
GetHashCode	用作特定类型的哈希函数，适合在哈希算法和数据结构（如哈希表）中使用
GetType	获取当前实例的 Type
ReferenceEquals	确定指定的 Object 实例是否是相同的实例
ToString	返回表示当前 Object 的 String
Finalize	已重写。允许 Object 在“垃圾回收”回收 Object 之前尝试释放资源并执行其他清理操作。在 C# 和 C++ 中，使用析构函数语法来表示终结程序
MemberwiseClone	创建当前 Object 的浅表副本

注意事项：

该类是所有类的基类，所有的类都继承了该类的方法。因此在本书中，除特殊情况之外，都不提及其他类自此类继承的方法。

方法参考：

1. Object 构造函数

重载列表：

public Object();

功能：

构造函数，可以由派生类调用，也可直接创建 Object 对象。

2. Equals 方法

重载列表：

1）public virtual bool Equals(object obj);

参数：

obj：跟当前对象比较的对象。

功能：

将指定对象 obj 跟当前对象比较，如果等于当前的 Object，则为 True；否则为 False。

2）public static bool Equals(object objA,object objB);

参数：

objA：要比较的第一个对象。

objB：要比较的第二个对象。

功能：

该方法是静态方法，无需对象引用。将两个对象 objA 和 objB 进行比较，如果相等，返回 True；否则返回 False。

3. GetHashCode 方法

重载列表：

public virtual int GetHashCode();

功能：

返回当前 Object 的 Hash 代码。

提示：

① 此方法可由派生类重写。值类必须重写此方法，以提供

适合该类并且确保哈希表中有更好的分布的 Hash 函数。Hash 函数应该为所有输入生成随机分布。

② 在哈希表中可以用作键的类必须重写此方法，在哈希表中用作键的对象通过此方法生成类自己的哈希代码是必需的。

③ 重写 GetHashCode 的派生类还必须重写 Equals，以保证被视为相等的两个对象具有相同的哈希代码；否则，Hashtable 可能不会正常工作。

④ 关于 Hashtable 和 Hash 函数的更多信息请参考本书相关内容。

代码示例：

```
/*****************************************************
程序编号：1-1。
演示函数原型：public virtual int GetHashCode();
简介：演示重写 GetHashCode()函数，其中使用了异或操作作为 Hash 函数。
文件名：11131.cs。
*****************************************************/
using System;
// 定义一个结构体
// 该结构体包含一个重写的 GetHashCode 方法
// 以及两个公有的整形变量
public struct Point
{
    public int x;
    public int y;
    // 重写的 GetHashCode 方法
    public override int GetHashCode()
    {
        // 重写后的方法返回自定义的 Hash 代码
        return x ^ y;
    }
}
```

4. GetType 方法

重载列表：

public Type GetType();

功能：

返回 Type 实例，表示当前实例的确切运行时类型。Type 对象公开与当前 Object 的类关联的元数据。

代码示例：

```
/*****************************************************
程序编号：1-2。
演示函数原型：public Type GetType();
简介：使用 GetType()方法返回对象的运行时类型。
文件名：11141.cs。
*****************************************************/
using System;
public class MyBaseClass: Object
{
}
public class MyDerivedClass: MyBaseClass
{
}
public class Test
{
    public static void Main()
    {
        MyBaseClass myBase = new MyBaseClass();
        MyDerivedClass myDerived = new MyDerivedClass();
        object o = myDerived;
        MyBaseClass b = myDerived;
        // 运行该程序，从下面的输出中可以看到对象的类型的文本表达
        // 输出基类型：MyBaseClass
        Console.WriteLine("mybase: Type is {0}", myBase.GetType());
        // 输出继承类型：MyDerivedClass
        Console.WriteLine("myDerived: Type is {0}", myDerived.GetType());
        // 将对象 myDerived 封装于 ojbect 类时的类型信息
        Console.WriteLine("object o = myDerived: Type is {0}", o.GetType());
        Console.WriteLine("MyBaseClass b = myDerived: Type is {0}", b.GetType());
    }
}
```

运行结果：

运行结果如图 1-1 所示。

图 1-1 运行结果

5. ReferenceEquals 方法

重载列表：

public static bool ReferenceEquals(object objA, object objB);

参数：

objectA：用来比较的第一个 Object 对象。

objectB：用来比较的第二个 Object 对象。

功能：

比较两个对象，如果 objA 是与 objB 相同的实例，或者如果二者都为空引用，则为 True；否则为 False。该方法为静态成员，不能使用对象来引用。

6. ToString 方法

重载列表：

public virtual string ToString();

功能：

返回代表当前 Object 的字符串。

在派生类中可以重写此方法，以返回对该类型有意义的值。

代码示例：

```
/*****************************************************
程序编号：1-3。
演示函数原型：public virtual string ToString();
```

简介：输出一个 Object 对象和一个 Int32 对象的字符串表示，其中系统类 Int32 重写了该方法，使返回值为一个表示整数的字符串而不是类名。

文件名：11161.cs。

```
*************************************************/
using System;
public class Sample
{
    void Method()
    {
        Object o = new Object();
        int i=10;
        // 对象 o 将被返回类名
        // 数 i 将被返回字符串 "10"
        Console.WriteLine (o.ToString());
        Console.WriteLine (i.ToString());
    }
}
```

运行结果：

运行结果如图 1-2 所示。

图 1-2　运行结果

7. Finalize 方法

重载列表：

Object();

功能：

由用户手动终止一个对象的运行并且释放其所有资源。

提示：

这是一个 protect 级别的函数，只有其派生类可以访问。事实上，派生类型中的每个 Finalize 实现都必须调用其基类型的 Finalize 实现。这也是惟一允许应用程序代码调用 Finalize 的情况，一般的情况下，垃圾回收机制会自动处理一切，没有必要使用这个方法。

如果 Finalize 或 Finalize 的重写引发异常，运行时将忽略异常，终止此 Finalize 方法，并继续终结进程。

8. MemberWiseClone 方法

重载列表：

protected object MemberwiseClone();

功能：

创建当前 Object 的浅表副本并返回之。

提示：

① 用户不能重写此方法，如果该函数不能满足要求，则派生类应实现 ICloneable 接口。

② 浅表副本创建与原始对象具有相同类型的新实例，然后复制原始对象的非静态字段。如果字段是值类型的，则对该字段执行逐位复制。如果字段是引用类型，则复制该引用但不复制被引用的对象；这样，原始对象中的引用和复本中的引用指向同一个对象。相反，对象的深层副本复制对象中字段直接或间接引用的全部内容。

1.2　异常处理

异常是程序执行时遇到的任何错误情况或意外行为。当用户的代码或调用的代码（如共享库）中有错误，操作系统资源不可用，公共语言运行库遇到意外情况（如无法验证代码）等情况出现时，就会引发异常。对于这些情况，ASP.NET 开发者都可以利用.NET 框架提供的异常处理机制正确应对，从而最大程度地提高 Web 应用程序的健壮性。

在 ASP.NET 中有两种类型的异常：由执行程序生成的异常和由公共语言运行库生成的异常。另外，还有由应用程序或运行库引发的异常的层次结构。Exception 是异常的基类。若干异常类直接从 Exception 继承，其中包括 ApplicationException 和 SystemException，这两个类构成几乎所有运行库异常的基础。

这一部分介绍 ASP.NET 异常处理的三个最基础的类，通过对这三个基类的继承和使用，用户可以轻松地处理几乎所有的异常。在使用方法及示例代码中，可以看到如何处理和引发异常，以及处理自定义异常。

1.2.1　System.ApplicationException 类

类名： System.ApplicationException。

声明：

public class ApplicationException : Exception

继承与被继承：

System.Object

System.Exception

System.ApplicationException

System.Reflection.InvalidFilterCriteriaException

System.Reflection.TargetException

System.Reflection.TargetInvocationException

System.Reflection.TargetParameterCountException

描述：

① ApplicationException 由用户程序引发，而不是由公共语言运行库引发。它继承自 Exception 类，但没有添加新功能。

② 该异常出现的目的是作为所有用户自定义异常的基类，从而可以区分应用程序定义的异常与系统定义的异常。

属性：

该类是 System.Exception 类的子类，但并没有扩展，其属性请参考 System.Exception 类的说明。

方法：

表 1-2 所示为 System.ApplicationException 类的方法。

表 1-2　System.ApplicationException 类的方法

方法	描述
ApplicationException	构造函数，有参数重载及访问权限重载

该类是 System.Exception 类的子类，但并没有扩展，其他方法请参考 System.Exception 类的说明。

注意事项：

该类自 Exception 继承过来，但是并未提供扩展。

所有用户自定义异常都应该作为该类的子类，该类的存在目的之一就是如此。

使用该类的原则之一是尽量不要抛出该类的对象，而应该抛出基于该类的自定义异常类的对象。

代码示例：

```
/*************************************************
程序编号：1-4。
演示函数原型：无。
简介：自定义一个异常类。
文件名：1211.cs。
*************************************************/
public class MailFailedException:ApplicationException
{
```

```
        // 异常描述信息字段
        public string Description;
        // 不带参数的构造函数
        public MailFailedException()
        {
            Description="";
        }
        // 提供带参数的构造函数
        public MailFailedException(string des)
        {
            Description=des;
        }
    }
```

方法参考：

ApplicationException 构造函数

重载列表：

1）public ApplicationException();

功能：

此构造函数将新实例的 Message 属性初始化为系统提供的消息，该消息对错误进行描述，例如："已发生应用程序错误"。此消息将考虑当前系统区域性。

2）public ApplicationException(string message);

参数：

message：用户自定义的错误信息，应该保证其易于理解和便于错误查找。

功能：

使用用户自定义的错误信息初始化一个 ApplicationException 实例。

3）protected ApplicationException(SerializationInfo info,StreamingContext context);

参数：

info：保存序列化对象的数据。

context：有关源或目标的上下文信息。

功能：

在反序列化过程中调用该构造函数来重建通过流传输的异常对象。

4）public ApplicationException(string message,Exception innerException);

参数：

message：解释异常原因的错误信息。

innerException：作为当前异常的原因的异常。如果 innerException 参数不为空引用，则在处理内部异常的 catch 块中引发当前异常。

功能：

使用用户自定义错误信息和引发该异常的内部异常初始化实例。作为上一异常直接结果而引发的异常应在 InnerException 属性中包括对上一异常的引用。InnerException 属性返回的值与传递到构造函数中的值相同；或者，如果 InnerException 属性不向构造函数提供内部异常值，则为空引用。

1.2.2 System.Exception 类

类名： System.Exception。

声明：

public class Exception : ISerializable

继承与被继承：

System.Object

System.Exception

派生类

描述：

① 类 System.Exception 是所有异常类的基类，整个.NET Framework 的异常处理机制就是基于这个类构建的。

② CLR 提供一种异常处理模型，该模型基于对象形式的异常表示形式，并且将程序代码和异常处理代码分到 try 块和 catch 块中。可以有一个或多个 catch 块，每个块都设计为处理一种特定类型的异常，或者将一个块设计为捕捉比其他块更具体的异常。

③ 用户在使用 try-catch-finally 结构进行异常处理时，通常是把执行代码放在 try 块中，处理由 try 块引发的异常的应用程序代码放在 catch 语句中，称为 catch 块。一个 try 块后面可以有零个或多个 catch 块，每个 catch 处理一种预料中的错误类型，这个错误类型在 catch 块的参数中说明。系统在找到第一个处理该异常的 catch 块后即停止搜索，因此，为了有针对性地处理错误，在应用程序代码中处理某类型的 catch 块必须在处理其基类型的 catch 块之前指定。处理 System.Exception 的 Catch 块最后指定。

④ 如果当前 try 块所关联的所有 catch 块均不处理该异常，且当前 try 块嵌套在当前调用的其他 try 块中，则搜索与下一个封闭 try 块相关联的 catch 块。如果没有找到用于该异常的 catch 块，则系统搜索当前调用中前面的嵌套级别。

⑤ 如果系统堆栈中没有更多的信息，则系统调用默认的异常处理程序，表现在 ASP.NET 中就是大家看到的出错页面，同时应用程序被中止。

⑥ 使用 System.Exception 类型对象处理异常，可以查看到异常发生时运行库产生的通知用户错误性质的文本消息及解决该问题的操作建议，还可以使用 StackTrace 属性跟踪堆栈，此外，若要向用户提供有关异常发生原因的大量信息，可以使用 HelpLink 属性保存帮助文件的 URL（或 URN）。

⑦ 从该类派生了两个异常处理的基类：ApplicationException 和 System.Exception。

属性：

表 1-3 所示为 System.Exception 的属性。

表 1-3 System.Exception 的属性

属性	描述
HelpLink	获取或设置指向此异常所关联帮助文件的链接
InnerException	获取导致当前异常的 内部的 Exception 实例
Message	获取描述当前异常的消息
Source	获取或设置导致错误的应用程序或对象的名称
StackTrace	获取当前异常发生时调用堆栈上的帧的字符串表示形式
TargetSite	获取引发当前异常的方法
HResult	protect 级别的属性。获取或设置 HRESULT，它是分配给特定异常的编码数值

方法：

表 1-4 所示为 System.Exception 类的方法。

表 1-4 System.Exception 类的方法

方法	描述
GetBaseException	当在派生类中重写时，返回 Exception，它是一个或多个并发的异常的源
GetObjectData	当在派生类中重写时，用关于异常的信息设置 SerializationInfo
ToString	已重写。创建并返回当前异常的字符串表示形式