

## .Net

选择 12

填空 10

判断 12

程序填空 30

2-3 题

简答题 36 (6\*6)

## 第一章

### 1. net 体系的构成

.NET 体系由公共语言运行库 (CLR) (Common Language Runtime) 和 .NET Framework 类库所构成。CLR 是 .NET Framework 的核心执行环境, 也称为 .NET 运行库。 .NET Framework 类库是一个内容丰富的类集合, 它可以完成以前要通过 Windows API 来完成的绝大多数任务。

### 2. 编译器和集成开发环境

Visual studio

### 3. 跨语言的操作性

ASP.NET 目前能支持多种语言, 主要包括:

C# (读作“C Sharp”意思是 C++++)、Visual Basic、J# (Java 语言的克隆体)、具有托管扩展的 C++ 以及 Jscript .NET (JavaScript 的服务器端版本) 等。

### 4. 类库

.NET Framework 类库是一个由 Microsoft .NET Framework SDK (软件开发工具包 Software development toolkit) 中包含的类、接口和值类型组成的库。该库提供对系统功能的访问, 是建立 .NET Framework 应用程序、组件和控件的基础。

### 5. 托管代码

托管代码是指用 .net framework 支持的语言写的代码。

### 6. Net 跨平台性 (。Net 中间语言是不是为了跨平台)

其实 ASP.NET 并不存在跨平台的问题, 只要你将 ASP.NET 程序部署到相应的服务器上。其他只要有浏览器的客户端都可以访问得到。

其实 WEB 的最大好处就是跨平台, 不管什么机器、什么系统只要有浏览器你都可以访问你的部署好的 ASP.NET 程序。

### 7. 垃圾回收

Garbage Collector (垃圾收集器, 在不至于混淆的情况下也成为 GC) 以应用程序的 root 为基础, 遍历应用程序在 Heap(堆)上动态分配的所有对象, 通过识别它们是否被引用来确定哪些对象是已经死亡的、哪些仍需要被使用。已经不再被应用程序的 root 或者别的对象所引用的对象就是已经死亡的对象, 即所谓的垃圾, 需要被回收。

## 第二章

### 1. 动态网页编程技术 (主要概念)

静态网页

在网站设计中, 纯粹 HTML 格式的网页通常被称为“静态网页”, 早期的网站一般都是由静态网页制作的。

静态网页一般会以 .htm 或者 .html 扩展名存储在服务器的文件系统中。

在 HTML 格式的网页上, 也可以出现各种动态的效果, 如 .GIF 格式的动画、FLASH、滚动字幕等, 这些“动态效果”只是视觉上的, 不是区分静态网页和动态网页的要素。

静态网页的特点主要有:

- (1) 静态网页的每个网页都有一个固定的 URL，且网页 URL 以 .htm、.html、.shtml 等常见形式为后缀；
  - (2) 网页内容一经发布到网站服务器上，都是保存在网站服务器上的；
  - (3) 网页内容不会发生变化
  - (4) 静态网页一般没有数据库的支持
  - (5) 静态网页不能和浏览器用户交互，只能单向发布信息，不适应动态和即时信息发布
  - (6) 维护工作量较大。
- 动态网页

动态网页的内容不是预先定制的静态 HTML 文档，而是在请求或使用过程中根据实际的数据内容和条件实时生成的页面。

动态网页的内容可随用户的输入和互动而有所不同，或者随着用户、时间、数据修正等而改变。

使用动态页面，用户可以提交信息给服务器，服务器可以将数据库中的数据返回给用户，例如股市行情、天气预报等动态检索结果。

## 2. IIS 虚拟目录

### 第三章

#### 1、基类

#### 2、继承

Class A:baseA    baseA 为父类，A 为派生类

#### 3、命名空间概念作用

- 命名空间用于组织 .NET Framework 提供的几千个类。
- 作为组织类的逻辑单元，命名空间成了应用程序的组织形式。
- 使用命名空间可以解决类名冲突问题。
- 利用 using 语句导入命名空间，如“using system;”表示导入 system 命名空间。

#### 4、引用类型/值类型包括

引用类型包括：类类型、数组类型、委托类型、接口类型

值类型包括：简单类型、结构类型、枚举类型

#### 5、二维数组的定义

C#中，我们在创建二维数组的时候，一般使用 arr[][] 的形式，例如

```
int[,] aInt = new int[2][];
```

但声明二维数组还有一种方法，是使用 arr[,] 的形式。两者有什么区别呢？

实际上，形如 arr[,] 只能声明等长的二维数组，例如

```
int[,] ab1 = new int [2,3]; //默认值为 0; int[,] ab2 = new int[2,3]{{1,2,3},{4,5,6}};
```

形如 `arr[][]` 的形式则可以声明等长二维数组，也可以声明不等长二维数组。例如  
`int [][] abc = new int[2][]; abc[0] = new int[]{1,2}; abc[1] = new int[]{3,4,5,6};`

```
namespace 二维数组
{
    class Program
    {
        static void Main(string[] args)
        {

            //二维数组：
            //规则二维数组的定义和初始化
            int[,] Arr = new int[2, 5] { { 1, 2, 3, 5, 6 }, { 1, 2, 3, 4, 5 } };
            Console.WriteLine("规则二维数组的输出： ");
            for (int i = 0; i < 2; ++i)
            {
                for (int j = 0; j < 5; ++j)
                {
                    Console.Write(Convert.ToString(Arr[i,j]) + " ");
                }
                Console.WriteLine();
            }
            Console.WriteLine("-----");
            //不规则二维数组
            int [][] arr = new int [3][]; //表示含有三个一维数组的数组
            arr[0] = new int[5]{1,2,3,4,5};
            arr[1] = new int [2]{0,1};
            arr[2] = new int[0] { };
            Console.WriteLine("输出方法一： ");
            for (int i = 0; i < 2; ++i)
            {
                for (int j = 0; j < 5; ++j)
                {
                    Console.Write(Convert.ToString(Arr[i, j]) + " ");
                }
                Console.WriteLine();
            }
            Console.WriteLine();
            Console.WriteLine("输出方法二： ");
            for (int ii = 0; ii < arr.Length; ++ii)// arr.Length 是 3 可以看出 arr 是包含三个
一维数组的数组
            {
                foreach (int j in arr[ii])
                {
```

```

        Console.WriteLine(j+ " ");
    }
    Console.WriteLine();
}
Console.ReadKey();

}
}

```

## 6、Try-catch-final(什么时候执行 final、什么时候执行 catch)

try { //执行的代码，其中可能有异常。一旦发现异常，则立即跳到 catch 执行。否则不会执行 catch 里面的内容 }

catch(这和括号里面的参数是表示的这个 catch 语句里面要对应处理的错误类型及其参数) { //除非 try 里面执行代码发生了异常，否则这里的代码不会执行 }

finally { //不管什么情况都会执行，包括 try catch 里面用了 return ,可以理解为只要执行了 try 或者 catch，就一定会执行 finally

## 7、装箱

- C#中，所有类型都是从 Object 对象继承来的，所以任何类型都可以给 Object 对象的实例赋值。
- 给 Object 对象赋值的过程称为装箱(Boxing)；反之，称为拆箱(Unboxing)。

```

int i = 123;
//装箱
object o = i;
// 拆箱，必须是显示转换
int j = (int) o;
Console.WriteLine("j: {0}", j);

```

## 8、类型强制转化

显式类型转换，即强制类型转换。显式转换需要强制转换运算符，而且强制转换会造成数据丢失。

```

static void Main(string[] args)
{
    double d = 5673.74;
    int i;

    // 强制转换 double 为 int
    i = (int)d;
    Console.WriteLine(i);
}

```

```
Console.ReadKey();
```

```
}
```

## 9、对象引用属性（string）

在使用类（class）的非静态字段、属性和方法前，必须先实例化对象，然后通过实例操作非静态字段、属性和方法。

因为被实例化对象是一个引用类型的变量，因此“非静态的字段 方法或属性要求对象引用”。

```
// 定义一个类 TestClass
class TestClass
{
    // 非静态字段
    public int X;
    // 非静态属性
    public string Name{get;set;}
    // 非静态方法
    public int Add(int a, int b)
    {
        return a + b;
    }
}

// 要使用 TestClass 非静态成员，必须先实例化 TestClass
// 注意：变量 tc 引用类型的
// 如果不实例化 TestClass，调用非静态成员时就会出错！
TestClass tc = new TestClass();
tc.X = 100;
tc.Name = "Li Shi";
int r = tc.Add(10, 10);
```

## 第四章

### 1、获取 UID 的方法

### 2、页面首次加载 pageload 里面的方法

- 当控件的事件被触发时，Page\_Load 事件会在控件的事件之前被触发。
- 如果想在执行控件的事件代码时不执行 Page\_Load 事件中的代码，可以通过判断属性 Page.IsPostBack 实现。
- 属性 IsPostBack 在用户第一次浏览网页时，会返回值 false，否则返回值 true。

### 3、自动回发 ispostback、autopostback（必考）

Autopostback: 获取或设置一个值，该值指示用户在 TextBox 控件中按 Tab 键时，是否会发生自动回发到服务

器的操作，默认为 false。

注:不论值为 true 或 false，按 Enter 键始终回发。

IsPostBack:

Page.IsPostBack 是一个标志：当前请求是否第一次打开。调用方法为：Page.IsPostBack 或者 IsPostBack 或者 this.IsPostBack 或者 this.Page.IsPostBack，它们都等价。

1)当通过 IE 的地址栏等方式打开一个 URL 时是第一次打开，当通过页面的提交按钮或能引起提交的按钮以 POST 的方式提交的服务器时，页面就不再是第一次打开了。（每点击一次按钮，都是一次加载）

2)IsPostBack 只有在第一次打开的时候是 false，其它时候都是 true

#### 4、属性绑定字段

Dropdownlist/checkbox(listcontrol 控件)

设置属性怎么考

DropDownList 控件是将选项显示为下拉列表，并从中进行单项选择。

CheckBox 和 CheckBoxList 控件都是为用户提供了一种在二选一（如真/假、是/否或开/关）选项之间切换的方法。

## CheckBox控件属性和事件

| 属性/事件名             | 说明                    |
|--------------------|-----------------------|
| Text               | 字符串，一般显示在单选框后边        |
| TextAlign          | Right/Left，文本对齐方式     |
| AutoPostBack       | true/false，默认为false   |
| Checked            | true/false，默认时控件是否被选中 |
| OnCheckedChanged() | 单击时触发的事件              |

#### 5、用户自定义控件（注册语句写法）

用户控件扩展名为.ascx

■ 使用 Register 指示符指定自定义控件的命名空间和名称

➢ TagPrefix: 标记前缀，定义控件的命名空间

➢ TagName: 标记名，指向所使用控件的名字

➢ Src: 执行控件的资源文件，使用相对路径（control.ascx 或~/path/control.ascx），不能使用

物理路径 (c:\path\control.ascx)

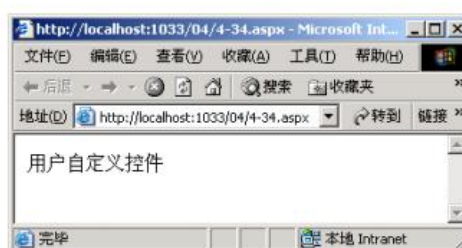
4-33.ascx

```
<% @Control Language="C#" %>
<p>用户自定义控件</p>
```

案例名称：用户控件  
程序名称：4-34.aspx

```
<%@ Page Language="C#" %>
<%@ Register TagPrefix="myCo" TagName="myTxt" Src="4-33.ascx" %>

<form id="form1" runat="server">
    <myCo:myTxt ID="MyTxt1" runat="server" />
</form>
```



## 6、验证控件：（1）服务器端验证控件与客户端验证控件区别（2）验证机制在什么时候用（3）可以放几个验证控件

- 为保证响应速度，一般设置验证控件的属性 `EnableClientScript` 值为 `true`。这样，当在页面上改变属性 `ControlToValidate` 指定控件的值并将焦点移出时，就会产生客户端验证。此时验证用的 `JavaScript` 代码不是由开发人员开发，而是由系统产生。
- 若将 `EnableClientScript` 值设为 `false`，则只有当页面有往返时，才会实现验证工作，此时完全使用服务器端验证。

## 第五章

### 1、Response.Write

- `Response` 对象主要功能是向浏览器输出信息。
- 常用的方法包括：
  - `Response.Write()` 向客户端输出数据
  - `Response.End()` 将缓冲区中信息输出，并且终止页面执行
  - `Response.WriteFile()` 向浏览器输出文本文件内容
  - `Response.Redirect()` 转向其他 URL 地址或网页
  - `Response.Clear()` 清除缓冲区中所有信息，前提是 `Response.Buffer` 设为 `True`
  - `Response.Flush()` 将缓冲区中信息输出，前提是

Response.Buffer 设为 True

## 2、Request 值获取方法

- Request 对象的功能是从客户端得到数据。

- 常用的三种取得数据的方法是：

- Request.Form

- Request.QueryString

- Request

其中，第三种是前两种的一个缩写，可以取代前两种情况。而前两种主要对应的 Form 提交时的两种不同的提交方法：分别是 Post 方法和 Get 方法。

- Request.Form[数据名称]获取 Post 方式提交的数据。

- Request.QueryString[数据名称]获取 Get 方式提交的数据

Request 的编码方式

当 Request 从客户端读取数据的时候，利用默认的编码方式对数据进行编码，可以用“ContentEncoding”属性得到当前 Request 的编码方式。

案例名称：从浏览器获取数据

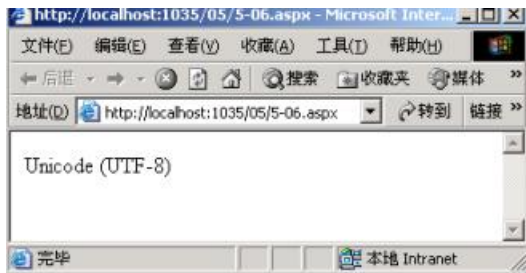
程序名称：5-06.aspx

```
<%@ Page Language="C#" %>
```

```
<%
```

```
Response.Write(Request.ContentEncoding.EncodingName);
```

```
%>
```



## 3、Application

- Application 对象是多用户共享的对象。所有的用户都可以对某个特定的 Application 对象进行修改。可以在 Web 应用程序运行期间持久地保持数据。生存期直到网站关闭。

- 利用 Application 特性，可以创建聊天室和网页计数器等常用网页应用程序。



## 向Application对象中写入数据并读取

案例名称：向Application对象中写入数据并读取  
程序名称：5-10.aspx

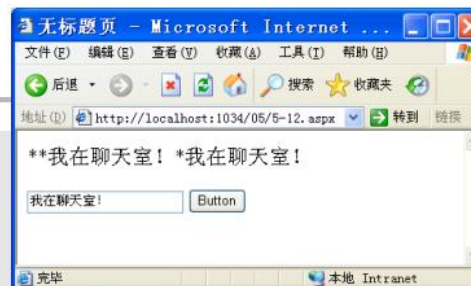
```
<% @ Page Language="C#" %>
<%
Application["Greeting"]="欢迎您的光临! ";
//创建一个名为Greeting的Application变量
%>
<%=Application["Greeting"]%>
```



## 聊天室应用

案例名称：单一文件聊天室  
程序名称：5-12.aspx

```
<% @ Page Language="C#" %>
<%
string mywords=Request["mywords"];
Application.Lock(); //保证同一时间只有一个用户操作Application对象
Application["chat_content"] = Application["chat_content"]
+ "*" + mywords;
Response.Write (Application["chat_content"]);
Application.UnLock();
%>
<form action="5-12.aspx" method="post">
<input type="text" size="30" name="mywords" value="我在聊天室! "/>
<input type="submit" value="提交"/>
</form>
```



### 4、Cookie、Session 生命周期

■ 使用 Session 对象存储特定的用户会话所需的信息。当用户在应用程序的页之间跳转时，存储在 Session 对象中的变量不会清除。

■ 使用 Session 的方法“Abandon()”显式地结束一个会话，利用属性“Timeout”设置 Session 的失效时间，默认时间是 20 分钟。

Session.Timeout = 1;

Session["Greeting"]="欢迎!";

Response.Write(Session["Greeting"]);

Session.Abandon();

Cookie 对象需要利用.NET 提供的 HttpCookie 类重新定义。使用 Response.Cookies.Add()将信息发送并保存到客户端的浏览器

```
<% @ Page Language="C#" %>
```

```
<%
```

```
HttpCookie MyCookie = new HttpCookie("user"); // user 为 cookie 名
```

```
MyCookie.Value = "为人民服务! "; // cookie 值
```

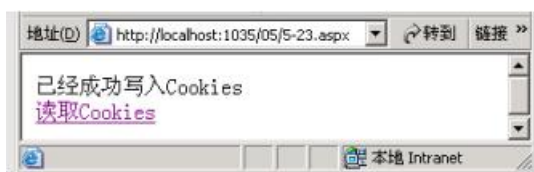
```
MyCookie.Expires = DateTime.Now.AddMinutes(20);//此时保存到硬盘
```

```
Response.Cookies.Add(MyCookie);
```

```
%>
```

已经成功写入 Cookies<br />

```
<a href="5-24.aspx">读取 Cookies</a>
```



**案例名称：从浏览器读取Cookie**

**程序名称：5-24.asp**

```
<% @ Page Language="C#" %>
```

```
<%
```

```
string mycook=Request.Cookies["user"].Value;  
Response.Write(mycook);
```

```
%>
```



## 5、Server。Mappath 意思、server url encode decode 互相转换输出

Server 对象

- 通过 Server 对象可以访问服务器的方法

和属性，比如得到服务器上某文件的物理路径和设置文件的执行期限，等等。

- Server 对象的常用方法：

- HTMLEncode()将字符串转换成 HTML 编码格式(例如字符“<”和“>” 被编码为<和>)

- MapPath()转换为服务器上的物理路径

- Execute()执行新的页面

- Transfer()转移到新的网页

- UriEncode()将字符串中某些特殊字符转换为 URL 编码，如将

“/”转换为“%2f”、“+”转换为“%2b”、空格转换为“+”等。常用于处理链接地址，如地址中包含空格等

■ `UrlDecode()`与 `UrlEncode` 作用相反

## 向浏览器输出HTML编码

案例名称：向浏览器输出HTML编码

程序名称：5-15.aspx

```
<% @ Page Language="C#" %>
<%
Response.Write(Server.HtmlEncode( "hello world;<p>你好! </p>"));
Response.Write("<br>");
Response.Write("hello world;<p>你好! </p>");
%>
```

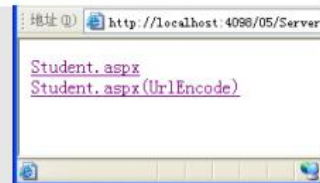


## Server.UrlEncode()应用

案例名称：Server.UrlEncode()应用

程序名称：Server.aspx

```
<% @ Page Language="C#" %>
protected void Page_Load(object sender, EventArgs e)
{
    //单击链接时将丢失“张”后面的信息
    Response.Write("<a href=Student.aspx?name=张 三> Student.aspx </a><br />");
    //编码后再单击链接时不会丢失“张”后面的信息
    Response.Write("<a href=Student.aspx?name=" +
        Server.UrlEncode("张 三") + "> Student.aspx(UrlEncode) </a>");
}
```



程序名称：Student.aspx

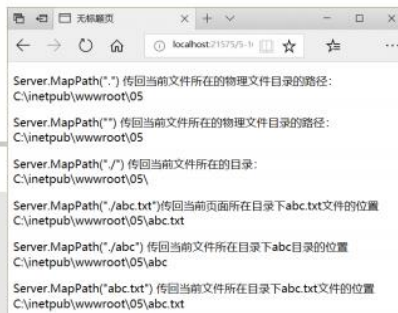
```
<form id="form1" runat="server">
<asp:Label ID="lblMsg" runat="server" Style="font-size: large" Text="Label"></asp:Label>
</form>
```

```
protected void Page_Load(object sender, EventArgs e)
{
    lblMsg.Text = Request.QueryString["name"] + "学生，欢迎您！";
    //lblMsg.Text = Server.UrlDecode(Request.QueryString["name"]) + "学生，欢迎您！";
}
```

## 取物理路径

案例名称：取物理路径  
程序名称：5-16.aspx

```
<form id="form1" runat="server">
  <p>Server.MapPath(".") 传回当前文件所在的物理文件目录的路径：<br />
    <%=Server.MapPath(".")%></p>
  <p>Server.MapPath("") 传回当前文件所在的物理文件目录的路径：<br />
    <%=Server.MapPath("")%></p>
  <p>Server.MapPath("./") 传回当前文件所在的目录：<br />
    <%=Server.MapPath("./")%></p>
  <p>Server.MapPath("./abc.txt") 传回当前页面所在目录下abc.txt文件的位置<br />
    <%=Server.MapPath("./abc.txt")%></p>
  <p>Server.MapPath("./abc") 传回当前文件所在目录下abc目录的位置<br />
    <%=Server.MapPath("./abc")%></p>
  <p>Server.MapPath("abc.txt") 传回当前文件所在目录下abc.txt文件的位置<br />
    <%=Server.MapPath("abc.txt")%></p>
</form>
```



### 6、配置文件

Global.asax 什么时候用，不用，干啥

- Global.asax 文件，也叫做 ASP.NET 网站配置文件。在网站的根目录中，该文件是可选的，可以不定义该文件。
- 主要用来提供应用程序和会话的开始和清除代码以及设置应用程序整体的参数。
- 当网站开启时自动调用 Global.asax 文件。

Web.config (1) 作用、(2) 与 machineconfig 的关系、(3) 配置内容读出来 (4) 带不带“”，方法名字

- ASP.NET 提供两种类型的运行配置文件：
  - 1、机器配置文件 machine.config——用以机器范围内的设置，提供了对整个 Web 服务器的配置。
  - 2、应用程序配置文件 web.config——用以应用程序特定的设置。



## 应用程序配置文件web.config

- ASP.NET 配置层次结构的根是一个称为根 web.config 的文件，它与 machine.config 文件位于同一个目录中。根 web.config 文件继承 machine.config 文件中的所有设置。
- 新建一个Web应用程序后，默认情况下会在根目录自动创建一个默认的web.config文件，该文件从根 web.config 文件继承默认配置设置。
- 多个文件名为 web.config 的配置文件可以出现在 ASP.NET 应用程序的多个目录中。每个 web.config 将配置设置应用于自身目录及其下所有子目录。子目录中的配置文件除了可以提供从父目录中继承而来的配置信息，还可以覆盖或修改父目录中定义的设置。

## 应用程序设置

应用程序设置允许在配置文件中存储应用程序的详细资料，比如对数据库连接串的设置：

案例名称：web配置文件

程序名称：web.config

```
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <appSettings>
    <add key="DSN" value="server=SZG-NB;uid=sa;pwd=;database=pubs" />
  </appSettings>
  <connectionStrings>
    <add name="sqlconn" connectionString="Data Source=localhost;
      Integrated Security=true;Initial Catalog=Northwind;" />
  </connectionStrings>
</configuration>
```

# 读取配置文件

案例名称: 读取配置文件  
程序名称: 5-26.aspx

```
<% @ Page Language="C#" %>

//读AppSettings
<%
string strData=System.Configuration.ConfigurationSettings.AppSettings["DSN"]; //1.1中
Response.Write(strData);
%><br />
<%=System.Configuration.ConfigurationManager.AppSettings["DSN"]%><br />
<%=System.Web.Configuration.WebConfigurationManager.AppSettings["DSN"]%><br />

//读ConnectionStrings<br />
<%=System.Configuration.ConfigurationManager.ConnectionStrings["sqlconn"].ConnectionS
tring%><br />
<%=System.Web.Configuration.WebConfigurationManager.ConnectionStrings["sqlconn"].Co
nnectionString%><br />
```

## 7、视图方法 ViewState

### ViewState 对象

- ViewState 是 ASP.NET 中对同一个窗体页的多次回发（PostBack）之间维持页及控件状态的一种机制。
- 通过 ViewState 在页面回传的过程中保存状态值，使原本没有“记忆”的 Http 协议变得具有了“记忆”。
- 当请求某个页面时，ASP.NET 会把所有控件的状态序列化成字符串，然后作为窗体的隐藏字段送到客户端。
- 由于隐藏窗体字段是发送到客户端的页面的一部分，所以 ViewState 值被临时存储在客户端的浏览器中。如果客户端选择将该页面回传给服务器，则 ViewState 字符串也将被回传

## 第六章

### 1、重要对象 connection 事务处理 connectstring 怎么写，不同数据库用什么样的 connectionstring

SqlConnection Conn;

```
//Conn=new SqlConnection("Addr=172.23.199.75;database=;uid=sa;pwd='123456'");
```

```
//Conn = new SqlConnection("server=localhost;database=;uid='sa';pwd='123456'");
```

```
//SQL Server 身份验证
```

```
Conn = new SqlConnection("Data Source=localhost;Integrated Security=true;Initial Catalog=;");
```

```
//Windows 身份验证
```

```
Conn.Open();
```

## Connection对象

用于建立数据库连接，常用方法有Open()和Close()。DataSource属性用来获取数据源的服务器名或文件名；Provider属性用来指定数据库驱动程序；ConnectionString属性用来指定连接字符串。

案例名称：使用Connection对象 程序名称：6-01.aspx

```
<% @ Page Language="C#" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.OleDb" %>
<script language="C#" runat="server">
void Page_Load(Object sender, EventArgs e) {
    OleDbConnection Conn=new OleDbConnection();
    Conn.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;" +
        "Data Source="+Server.MapPath("person.mdb");
    Conn.Open();
    Message.Text=Conn.State.ToString();
    Conn.Close(); }
</script>
<asp:Label id="Message" runat="server" />
```



### ADO.NET 事务处理

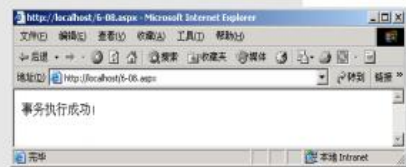
- 事务是一些事件的集合，执行一条 SQL 语句可理解成一个事件。
- ADO.NET 使用 Connection 对象的 BeginTransaction() 方法来声明事务开始，利用 Transaction 对象的 Commit() 方法来提交事务，利用 Transaction 对象的 Rollback() 方法来回滚事务

案例名称：使用事务的基本格式 程序名称：6-08.aspx

```

<% @ Page Language="C#" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.OleDb" %>
<script runat="server">
void Page_Load(Object sender, EventArgs e) {
    OleDbConnection Conn=new OleDbConnection();
    Conn.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;"+
        "Data Source="+Server.MapPath("person.mdb");
    Conn.Open();
    OleDbCommand Comm = new OleDbCommand();
    OleDbTransaction Trans;
    Trans = Conn.BeginTransaction();
    Comm.Connection = Conn;
    Comm.Transaction = Trans;
    try {
        Comm.CommandText = "UPDATE grade SET 数学=100 WHERE 姓名 LIKE '%周%'";
        Comm.ExecuteNonQuery();
        Comm.CommandText = "UPDATE grade SET 数学=60 WHERE 姓名 LIKE '%张%'";
        Comm.ExecuteNonQuery();
        Trans.Commit();
        Response.Write("事务执行成功!");
    }
    catch(Exception ex) {
        Trans.Rollback();
        Response.Write("出现错误，事务已经回滚!");
    }
    finally { Conn.Close(); }
}
</script>

```



## 2、数据源与数据绑定控件同步更新要怎么操作

写在哪，什么方法

## 3、GridView

## 4、数据绑定不同写法对象

## 5、类似 contro 类数据绑定时属性处理（值，索引，显示）

## 6、配置 SqlDataSource 数据源控件，设计视图，高级按钮（1）干啥（2）能操作什么东西（3）什么东西操作不了

## 二、程序设计题

### 1、用什么对象从数据源获取数据，对数据操作时示例（写关键部分代码）

### 2、C#

### 3、验证控件（重要）

## 三、简答题

### 1、Net Framework, C#, Asp.Net, Visual Studio.Net 之间的关系

.net framework 是一个多语言组件开发和执行环境，他提供一个跨语言的统一编程环境。

ASP.net 是全新的动态网页制作技术，是建立在.Net Framework 的公告语言运行时之上，可用在服务器上生成功能强大的 WEB 应用程序。ASP.Net 是.Net Framework 的一部分。C#是 ASP。Net 支持的语言之一。Visual Studio 是 ASP.Net 网站开发的集成环境。

### 2、页面与服务器交互过程简述



## 扩展：网页执行方式

当用户进行网页操作的时候，客户端浏览器会向服务器进行请求，然后服务器会生成一个新的网页返回给客户端。

■ 那么这个过程循环如下：

- |     |   |    |  |
|-----|---|----|--|
| 第一次 | { | 1. | 用户请求页面（HTTP GET方法）这个网页第一次运行，执行第一次运行所需要的步骤。                         |
|     |   | 2. | 服务器生成页面（标记语言）发送给浏览器，浏览器解析标记语言。                                     |
|     |   | 3. | 用户输入信息或者选择可选项，然后点击提交按钮。（如果用户点击了链接，那么会跳转到其他页面）                      |
| 回传  | { | 4. | 这个页面就发送（posted）给Web服务器。（HTTP POST 方法，在ASP.NET 中是postback）可以回发页面自己。 |
|     |   | 5. | 在Web服务器上，这个页面就重新运行，用户输入的信息就可以在这个页面上被使用。                            |
|     |   | 6. | 这个页面会执行你编程的逻辑任务。   |
|     |   | 7. | 这个页面再把自己发回浏览器上。  |



### 3、ADO.NET 怎么取数据，用到那些类

### 4、命名空间与类之间的关系（类库）

- 命名空间用于组织.NET Framework 提供的几千个类。
- 作为组织类的逻辑单元，命名空间成了应用程序的组织形式。
- 使用命名空间可以解决类名冲突问题。
- 利用 using 语句导入命名空间，如“using system;”表示导入 system 命名空间。

### 5、页面重定向

```
Response.Redirect("5-02.aspx");  
<asp:Button ID="Button1" runat="server" PostBackUrl="5-08.aspx" Text="提交" />  
Response.Redirect();
```

Server.Execute()和 Server.Transfer()  
<a href="5-19.aspx">在另外页面查看</a>

## 6、页面之间传递数据

### 一、Request.QueryString

核心代码：

```
1 protected void getQueryString_Click(object sender, EventArgs e)
2 {
3     string QueStr = Request.QueryString["name"];
4     Response.Write(QueStr);
5 }
```

总结：

- 1、Request.QueryString:获取http查询字符串变量集合。有两重载，即Request.QueryString[string name]和Request.QueryString[int index];
- 2、Request.QueryString主要是获取url中的“?”之后的参数，如url:a.aspx?name="queryString", 则Request.QueryString["name"]的值为“queryString”。

-- -- --

### 二、Request.Form

核心代码：

```
1 protected void getQueryString_Click(object sender, EventArgs e)
2 {
3     string strQueForm = Request.Form["TextBox1"];
4     Response.Write(strQueForm);
5 }
```

总结：

- 1、Request.Form 获取窗体变量集合。有两重载，即Request.Form[string name]和Request.Form[int index]。
- 2、获取表单指定名字的参数值。

## Session对象

- 使用 Session 对象存储特定的用户会话所需的信息。当用户在应用程序的页之间跳转时，存储在Session对象中的变量不会清除。
- 使用 Session的方法“Abandon()”显式地结束一个会话，利用属性“Timeout”设置Session的失效时间，默认时间是20分钟。

## Cookie对象

- Cookie对象也可以保存客户信息，与Session对象相似，分别保存不同用户的信息。
- 和Session的区别是：Session对象的信息保存在服务器上，Cookie对象的信息保存在客户端的浏览器上。

7、页面往返处理(同 2)

8、表单 Html 表单与 Web 表单的区别

## HTML表单

- HTML表单包含了表单内部控件和相应的布局信息。
- HTML表单
  - HTML表单是在Web页中的<form>和</form>标记之间定义的控件组，用于让用户输入数据并提交。例：

```
<form method="post" action ="page.html">
  输入您的用户名:
  <input type="text" name="username">
  <input type="submit" name="ok" value="提交" >
</form>
```

## Web表单

- ASP.NET引入了Web表单的概念。Web表单中则包含了表单内部控件、相应的布局信息及数据提交后的数据处理代码。
- Web表单
  - Web表单和HTML表单从本质上来讲是完全不同的表单，但在表现形式上没有太大差别。Web表单通常采用下面的方式来表示：

```
<form runat="server">
```

```
.....
```

```
</form>
```

### 9、静态页面与动态页面的区别

#### 静态页面

- 在网站设计中，纯 HTML 格式的网页通常被称为“静态网页”，网页 URL 通常以.htm、.html、.shtml、.xml 等常见形式为后缀。
- 所谓“静态”是指网站的网页内容“固定不变”，客户端浏览器访问.htm 文件时，IIS 不经过任何处理就直接送往浏览器，由浏览器解释执行。
- 在 HTML 格式的网页上，也可以出现各种动态的效果，如.GIF 格式的动画、FLASH、滚动字母等，这些“动态效果”只是视觉上的。

#### 动态页面

- 动态网页是与静态网页相对应的，也就是说，网页 URL 的后缀不是.htm、.html、.shtml、.xml 等静态网页的常见形式，而是以.asp、.aspx、.jsp、.php、.perl、.cgi 等形式为后缀。
- 动态网页使用的语言：HTML+ASP(ASP.NET) 或 HTML+PHP 或 HTML+JSP 等。
- 在服务器端运行的程序、网页、组件，属于动态网页，它们会随不同客户、不同时间，返回不同的网页，例如 ASP、PHP、JSP、ASP.NET 页。
- 动态网页以数据库技术为基础，可以大大降低网站维护的工作量。



## 10、页面保持视图状态（viewstate）

### ViewState对象

- ViewState是ASP.NET中对同一个窗体页的多次回发（PostBack）之间维持页及控件状态的一种机制。
- 通过ViewState在页面回传的过程中保存状态值，使原本没有“记忆”的Http协议变得具有了“记忆”。
- 当请求某个页面时，ASP.NET会把所有控件的状态序列化成一個字符串，然后作为窗体的隐藏字段送到客户端。
- 由于隐藏窗体字段是发送到客户端的页面的一部分，所以ViewState值被临时存储在客户端的浏览器中。如果客户端选择将该页面回传给服务器，则ViewState字符串也将被回传。

## 11、DataReader、DataSet 什么时候用，区别

在决定应用程序应使用 DataReader 还是使用 DataSet 时，应考虑应用程序所需的功能类型。

DataSet 用于执行以下功能：

- 在应用程序中将数据缓存在本地，以便可以对数据进行处理。如果只需要读取查询结果，DataReader 是更好的选择。
- 从 XML Web 服务对数据进行远程处理。
- 与数据进行动态交互，对数据进行排序、筛选或组合并关联来自多个源的数据。
- 对数据执行大量的处理，而不需要与数据源保持打开的连接，从而将该连接释放给其他客户端使用。