Web服务器控件中, 控件用于创建文本或图片超链接。

关于DropDownList控件，下列说法正确的是。 A

- A 在客户端被解释成HTML标记
- B 可以有多个选项处于选中状态 （下拉，只能一个，多选要ListBox）
- C DataTextField属性用于设置列表项的值内容 （Text部分，名称内容）
- D SelectedValue属性用于获取选中的项的Text属性 （Value部分）

在ADO.NET中，对于Command对象的 ExecuteNonQeury（）方法和ExecuteReader（）方法，下面叙述错误的是。C

- A insert、update、delete等操作主要用ExecuteNonQuery（）方法来执行
- BExecuteNonQuery（）方法返回执行Sql语句所影响的行数。
- CSelect操作的Sql语句只能由ExecuteReader（）方法来执行。(Select * 用ExecuteReader，返回单值使用ExecuteScale())
- DExecuteReader（）方法返回一个SqlDataReader对象

Request对象的MapPath方法可以用来获取应用程序的虚拟路径。（）X 通过虚拟路径获取绝对路径

在网页生命周期事件中，先执行页面加载Page_load事件，再响应Button_Click按钮单击事件。√

验证控件的ErrorMessage属性值和Text禹性值同时设置时，验证控件中显示的文本是ErrorMessage属性值。（）X 显示的text值，ErrorMessage会在验证汇总控件中显示

使用SqlDataAdapter与数据库交互时，通过Query方法实现查询功能，通过Update方法实现增加、修改、删除功能。（）X

使用 IsPostback 属性可判别页面是首次加载还多次回访问。
通过 CompareValidator 控件验证用户的输入是否符合指定的类型或者符合另一个输入控件的值，通过设置比较验证控件的 ControlToValidate 属性指定被验证控件的ID。

简答

5种异常处理机制

验证控件的名称及功能

如何建立与特定数据库的连接
（四个配置项？）

程序填空

课本

3.1.2生命周期事件

说明	事件
启动后第一个用户请求触发	Application_Start()
应用重启后触发	Application_End()
相应未被处理的错误	Application_Error()
一次会话第一次请求web	Session_Start()
会话超时或以编程方式结束	Session_End()

3.2.3 Page类的属性

- IsPostBack属性

```
protected void Page_Load(Object sender,EventArgs e)
{
    if(!IsPostBack)
    {
        Response.Write("First");
    }
    else
    {
        Response.Write("Not the first")
    }

}
```

3.2.4 网页的生命周期与Page类的事件

常规页生命周期阶段

阶 段	说 明
页请求	页请求发生在页生命周期开始之前。用户请求页时，ASP.NET将确定是否需要分析和编译页（从而开始页的生命周期），或者是否可以在不运行页的情况下发送页的缓存版本以进行响应。
开始	在开始阶段，将设置页属性，如Request和Response。在此阶段，页还将确定请求是回发请求还是新请求，并设置IsPostBack属性。此外，在开始阶段，还将设置页的UICulture属性。
页初始化	页初始化期间，可以使用页中的控件，并将设置每个控件的UniqueID属性。此外，任何主题都将应用于页。如果当前请求是回发请求，则回发数据尚未加载，并且控件属性值尚未还原为视图状态中的值。

29

加载	加载期间，如果当前请求是回发请求，则将使用从视图状态和控件状态恢复的信息加载控件属性。
验证	在验证期间，将调用所有验证程序控件的Validate方法，此方法将设置各个验证程序控件和页的IsValid属性。
回发事件处理	如果请求是回发请求，则将调用所有事件处理程序。
呈现	在呈现之前，会针对该页和所有控件保存视图状态。在呈现阶段中，页会针对每个控件调用Render方法，它会提供一个文本编写器，用于将控件的输出写入页的Response属性的OutputStream中。
卸载	完全呈现页并已将页发送至客户端，准备丢弃该页时，将调用卸载。此时，将卸载页属性（如Response和Request）并执行清理。

30

生命周期

页事件	典型使用
Page_PreInit	检查IsPostBack属性来确定是否是第1次处理该页；创建或重新创建动态控件；动态设置母版页；动态设置Theme属性；读取或设置配置文件属性值。注意：如果请求是回发请求，则控件的值尚未从视图状态还原。如果在此阶段设置控件属性，则其值可能会在下一事件中被覆盖。
Page_Init	在所有控件都已初始化且已应用所有外观设置后引发。使用该事件来读取或初始化控件属性。
Page_Load	读取和设置控件属性；建立数据库连接。

31

控件事件	使用这些事件来处理特定控件事件，如Button控件的Click事件或TextBox控件的TextChanged事件。注意：在回发请求中，如果页包含验证程序控件，在执行控件事件前，一般要检查Page和各个验证控件的IsValid属性，看页面验证是否通过。
Page_PreRender	使用该事件对页或其控件的内容进行最后更改。
Page_UnLoad	该事件首先针对每个控件发生，继而针对该页发生。在控件中，使用该事件对特定控件执行最后清理，如关闭控件特定数据库连接。对于页自身，使用该事件来执行最后清理工作，如：关闭打开的文件和数据库连接，或完成日志记录或其它请求特定任务。注意：在卸载阶段，页及其控件已被呈现，因此无法对响应流做进一步更改。如果尝试调用方法（如Response.Write方法），则该页将引发异常。

3.3 Page类的内置对象

Response对象

方法	说明

方法	说明
Redirect	重定向到新url
Write	将信息写入HTTP相应
WriteFile	将一个文件直接输出到客户端

Request对象

属性	说明
PhysicalApplicationPath	获取目前执行的服务器端程序在服务器的真实路径

Server对象

方法	说明
Executue	执行对另一页的请求
Transfer	终止当前页的执行，并开始执行新页

3.5 Web异常处理

五种异常处理机制，按优先级排序：

1. 通过try-catch异常处理块处理异常

```
try
{

}
catch (Exception ex)
{
    Response.Write("发生异常，原因"+ex.Message);
}
```

2. 通过页面级Page_Error事件处理异常

```
protected void Page_Load(Object sender,EventArgs e)
{
    throw new Exception("发生异常");

}
```

```
protected void Page_Error(object sender,EventArgs e)
{
    Exception objErr = Server.GetLastError();//获取未处理异常
    Response.Write("Error:"+objErr.Message);//输出异常信息
    Server.ClearError();//清除异常
}
```

3. 通过页面级ErrorPage属性处理异常
   web.config中的system.web配置块中添加`<customErrors>`，并mode属性必须设置为On
   添加ErrorPage.aspx和Error.htm

```
#ErrorPage.aspx
protexted void Page_Load(object sender, EventArgs e)
{
    thie.ErrorPage = "~/Error.htm";//发生异常跳转Error.htm
    throw new Exception("ErrorPage");

}
```

4. 通过应用程序级的Application_Error事件处理异常
   Global.asax文件下添加

```
void Application_Error(object sender, EventArgs e)
{
    //获取异常
    Exception ex = Server.GetLastError();
    //异常写入日志
    System.Diagnostics.EventLog log = new System.Diagnostics.EventLog();
    log.Source="应用程序级异常处理";
    log.WriteEntry(ex.Message, System.Diagnostics.EventLogEntryType.Error);
    Server.ClearError();
}
```

5. 通过配置应用程序的`<customErrors>`配置节处理异常
   web.config中的system.web配置块中添加`<customErrors>`修改为

```
<customErrors mode = "RemoteOnly" defaultRedirect = "ErrorPage.htm">
    <error statusCode = "403" redirect = "NoAccess.htm"/ >
    <error statusCode = "404" redirect = "FileNotFound.htm"/ >
</customErrors>
```

RemoteOnly意味着远程显示通用错误，本地显示详细错误
off：都详细错误
on：都显示通用错误

4 服务器控件

4.2 HTML服务器控件

添加`runat="server"`，可以将html控件转换为HTML服务器控件
如`<input id = "xx" type="submit" runat="server">`

添加服务器端onserverlick事件，添加`onserverclick="Submit1_ServerClick"`

```
#aspx.cs添加
protected void Submit1_ServerClick(Object sender,EventArgs e)
{
    Response.Write("服务端ServerClick事件");
}
```

web控件公共属性

属性	说明
AccessKey	定义控件快捷键
TabIndex	设置控件的Tab顺序
BackColor	控件的背景颜色
Enable	是否能被用户访问
Font	控件上的文本字体
ForeColor	控件上文本颜色
Height	
Width	
ToolTip	设置当鼠标悬停在web控件上时显示的文本
Visible	是否可见

web服务空间常用服务器端事件

事件	说明

事件	说明
Click	控件被单机时会触发该事件，button具有此事件
TextChanged	web文本发生变化时会触发该事件
CheckChanged	web控件上的选项发生变化时触发该事件

控件

Button控件

特有属性：

- CommandArgument
- CommandName
- OnClientClick
- PostBackUrl
- Text
  还支持俩事件：
- Click：单机按钮引发
- Command：单击按钮时引发。CommandName和CommandArgument属性的值传给这个事件
  网页添加

```
<asp:Button ID="控件名" runat="server" onlick="bunSubmit_Click" Text="提交"/ >
```

后台代码添加

```
protected void btnSubmit_Click(Object sender,EventArgs e)
{
    lblTime.Text = System.DateTime.Now.ToShortDateString();//显示系统时间
}
```

DropDownList控件

属性：

- AutoPostBack属性：改变DropDownList选项时，是否自动回传数据
- Items属性
- SelectedIndex属性
- SelectedItem属性
  网页

```
<asp:DropDownList ID = "DropDownList1" runat="Server"> </asp:DropDownList>
```

CheckBoxList

属性

- RepeatDirection
- RepeatColumns
- TextAlign

```
<asp:CheckBoxList ID="控件名" runat="server">
    <asp:ListItem>唱歌</asp:ListItem>
    <asp:ListItem>唱歌</asp:ListItem>
</asp:CheckBoxList>
```

RadioButtonList

```
<asp:RadioButtonList ID="控件名" runat="server">
    <asp:ListItem>唱歌</asp:ListItem>
    ...
</asp:RadioButtonList>
```

后台

```
protected void Button_Click(object sender,EventArgs e)
{
    if(RadioButtonList1.SlectedItem! = null)
    {
        Response.Write(RadioButtonList1.SelectedItem.Text);
    }
}
```

4.4 验证控件

验证类型	控件	说明
必填	RequireFieldValidator	验证一个必填字段
与某值对比	CompareValidator	将用户输入与另一个值进行比较或类型检查
范围验证	RangeValidator	是否上下限内，可检查数值、字符串、日期
模式匹配	RegularExpressionValidator	检查输入是否匹配正则

验证类型	控件	说明
自定义验证	CustomValidator	自定逻辑验证输入

共有属性

- ControlToValidate 指定要验证的输入控件名字

4.5 用户控件

连接数据库

8.2 连接模式

web.config，`<configuration>`添加

```
<connectionStrings>
    <add name="StudentCnnString" connectionString = "Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\Student.mdf;Integrated
Security=True;User Instance=True" providerName="System.Data.SqlClient"/ >
</connectionStrings>
```

使用SqlConnection对象

常用方法：

- Open() 打开链接
- Close()
- BeginTransaction() 开始事务，可以指定事务名字和隔离等级
- ChangeDatabase() 打开链接情况下更改数据库
- CreateCommand() 创建并返回SqlCommand对象
- Dispose() 调用close关闭并释放所占用的系统资源
  aspx.cs中引入

```
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
```

```
protected void Page_Load(object sender,EventArgs e)
{
    //从web.config读取连接字符串
```

```
    string strCnn =
ConfrigurationManager.ConnectionStrings["StudentCnnString"].ConnectionStrings;
    //定义连接对象
    SqlConnection cnn = new SqlConnection(strcnn);

    //打开链接
    cnn.open();
    cnn.close();
}
```

SqlCommand

属性：

- CommandText
- CommandType
- Connection
- Parameters
- Transaction

方法：

- ExecuteNonQuery 返回数据表被影响的行数，用于Insert，Update和Delete
- ExecuteReader 返回DataReader对象，用于执行多条记录的select
- ExecuteScalar 以object类返回结果第一行第一列的值，用于查询单值select命令

```
    string strCnn =
ConfrigurationManager.ConnectionStrings["StudentCnnString"].ConnectionStrings;
    //定义连接对象
    SqlConnection cnn = new SqlConnection(strcnn);

    SqlCommand cmd = new SqlCommand();//创建命令对象
    cmd.Connection = cnn;//设置数据连接属性
    cmd.CommandText = "xxxxx";

    cnn.open();
    cmd.ExecuteNonQuery();
```

SqlDataReader读取数据

常用属性

- FieldCount

```
    string strCnn =
ConfrigurationManager.ConnectionStrings["StudentCnnString"].ConnectionStrings;
    //定义连接对象
    SqlConnection cnn = new SqlConnection(strcnn);

    SqlCommand cmd = new SqlCommand();//创建命令对象
    cmd.Connection = cnn;//设置数据连接属性
    cmd.CommandText = "selecct * from Info";
    try
    {
        if(cnn.State==ConnectionState.Closed)
            cnn.open();
        stuReader = cmd.ExecuteReader();
        for(int i=0;i<stuReader.FieldCount;i++)
        {
            Response.Write(stuReader.GetName(i));
        }
        while(stuReader.Read())
        {
            for(int j=0;j<stuReader.FieldCount;j++)
            {
                Response.Write(stuReader.GetValue(i));
            }
        }
    }
    catche (Exception ex)
    {

    }
    finally
    {
        if(stuReader.IsClosed == false)
            stuReader.Close();
        if(cnn.State == ConnectionState.Open)
            cnn.Close();
```

```
    }
```

断开模式访问

DataSet数据集

属性方法	说明
DataSetName属性	获取或设置DataSet对象名称
Tables属性	获取数据集的数据表集合
Clear方法	删除对象中的所有表
Copy方法	复制结构和数据

`DataSet 对象名 = new DataSet();`或`DataSet 对象名 = new DataSet("数据集名称");`

SqlDataAdapter

常用方法

- Fill方法

```
using(SqlConnection cnn = new SqlConnection(strCnn))
{
    SqlDataAdapter daStu = new SqlDataAdapter(" select * from xx",cnn);
    DataSet dsStu = new DataSet();
    try
    {
        //调用Fill方法填充DataSet的数据表StuInfo
        daStu.Fill(daStu,"StuInfo");
        //绑定到GridView上显示
        GridView1.DataSource = daStuTables["Stuinfo"];
        GridView1.DataBind();
    }
    catch (...)
}
```

数据源控件

SqlDataSource数据源控件