北京信息科技大学

毕业设计（论文）附录

题　　目： 民航订票管理系统

学　　院： 计算机学院

专　　业： 计算机科学与技术

学生姓名： **班级/学号**

指导老师/督导老师： 李艳平

起止时间： 2009 年 2 月 23 日 至 2009 年 6 月 19 日

目录

程序代码清单

Login.aspx.cs

```
private void login_Click(object sender, System.EventArgs e)
         {
              if(Page.IsValid)
              {
                   string userName=Request.Form.Get("userName").ToString();
                   string userPwd=Request.Form.Get("userPwd").ToString();
                   SqlConnection con=Class.connection();
                   con.Open();
                   SqlCommand com=new SqlCommand("select count(*) from users where
username='"+userName+"'and userpassword='"+userPwd+"'",con);
                int count=Convert.ToInt32(com.ExecuteScalar());//判断是否登录成功
                   if(count>0)//成功登录
                   {
                        SqlCommand com2=new SqlCommand("select usertype from users where
username='"+userName+"' ",con);//判断用户类型
                        int judge=Convert.ToInt32(com2.ExecuteScalar());
                        if (judge==1)//管理员登录
                        {
                             Session["flag"]=true;
                             UserJudge user=new UserJudge();
                             user.username=userName;
                             Session["admin"]=user;
                             Response.Redirect("admin.aspx");}
                        else//普通用户登录

                        {
                             Session["flag"]=true;
                             UserJudge user=new UserJudge();
                             user.username=userName;
                             Session["user"]=user;
                             Response.Redirect("default.aspx");

                        }
                   }
                   else
                   {

                        Response.Write("<script language='javascript'>alert('用户名或密码错
误!');</script>");


                   }
              }
         }

private void cane1_Click(object sender, System.EventArgs e)
```

```
{    userName.Text="";userPwd.Text="";
}
```

Default.aspx.cs

```
private void Page_Load(object sender, System.EventArgs e)
    {

        TotalLabel.Text=Application["total"].ToString();
        OnlineLabel.Text=Application["online"].ToString();
        if(Session["flag"]==null)
        {
            LoginJudgeLabel.Text="未登录";
        }
        else
        {

            UserJudge user=(UserJudge)Session["user"];
            LoginJudgeLabel.Text="欢迎"+user.username+"光临本网站";
        }
        if(!this.IsPostBack)//是第一次，就执行绑定。
        {  //绑定出发地点
            SqlConnection con=Class.connection();//数据库连接
            con.Open();
            SqlCommand com=new SqlCommand("select distinct planestartplace from plane",con);
            SqlDataReader sdr=com.ExecuteReader();
            this.SPDropDownList.DataSource=sdr;
            this.SPDropDownList.DataTextField="planestartplace";
            this.SPDropDownList.DataValueField="planestartplace";
            this.SPDropDownList.DataBind();
            sdr.Close();
            //绑定到达地点
            SqlCommand com1=new SqlCommand("select distinct planeendplace from plane ",con);
            sdr=com1.ExecuteReader();
            this.EPDropDownList.DataSource=sdr;
            this.EPDropDownList.DataTextField="planeendplace";
            this.EPDropDownList.DataValueField="planeendplace";
            this.EPDropDownList.DataBind();
            sdr.Close();
            con.Close();

        }
```

```
        }

        #region Web 窗体设计器生成的代码
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: 该调用是 ASP.NET Web 窗体设计器所必需的。
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// 设计器支持所需的方法 - 不要使用代码编辑器修改
        /// 此方法的内容。
        /// </summary>
        private void InitializeComponent()
        {
            this.SPDropDownList.SelectedIndexChanged += new
System.EventHandler(this.SPDropDownList_SelectedIndexChanged);
            this.EPDropDownList.SelectedIndexChanged += new
System.EventHandler(this.EPDropDownList_SelectedIndexChanged);
            this.RadioButtonList1.SelectedIndexChanged += new
System.EventHandler(this.RadioButtonList1_SelectedIndexChanged);
            this.Button1.Click += new System.EventHandler(this.Button1_Click);
            this.DataGrid1.ItemCommand += new
System.Web.UI.WebControls.DataGridCommandEventHandler(this.DataGrid1_ItemCommand);
            this.DataGrid1.PageIndexChanged += new
System.Web.UI.WebControls.DataGridPageChangedEventHandler(this.DataGrid1_PageIndexChanged);
            this.Datagrid2.ItemCommand += new
System.Web.UI.WebControls.DataGridCommandEventHandler(this.Datagrid2_ItemCommand);
            this.Datagrid2.PageIndexChanged += new
System.Web.UI.WebControls.DataGridPageChangedEventHandler(this.Datagrid2_PageIndexChanged);
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion
```

```
private void SPDropDownList_SelectedIndexChanged(object sender, System.EventArgs e)
{
    if(!this.IsPostBack)
    {   //绑定出发地点
        SqlConnection con=Class.connection();//数据库连接
        con.Open();
        SqlCommand com=new SqlCommand("select distinct planestartplace from plane",con);
        SqlDataReader sdr=com.ExecuteReader();
        this.SPDropDownList.DataSource=sdr;
        this.SPDropDownList.DataTextField="planestartplace";
        this.SPDropDownList.DataValueField="planestartplace";
        this.SPDropDownList.DataBind();
        sdr.Close();
        con.Close();
    }


}

private void EPDropDownList_SelectedIndexChanged(object sender, System.EventArgs e)
{
    if(!this.IsPostBack)
    {  //绑定到达地点
        SqlConnection con=Class.connection();//数据库连接
        con.Open();
        SqlCommand com1=new SqlCommand("select distinct planeendplace from plane '",con);
        SqlDataReader sdr=com1.ExecuteReader();
        this.EPDropDownList.DataSource=sdr;
        this.EPDropDownList.DataTextField="planeendplace";
        this.EPDropDownList.DataValueField="planeendplace";
        this.EPDropDownList.DataBind();
        sdr.Close();
        con.Close();

    }

}

private void Button1_Click(object sender, System.EventArgs e)
{
    if(this.SPDropDownList.SelectedValue==this.EPDropDownList.SelectedValue)
```

```
                {

                Response.Write("<script language='javascript'>alert('出发地点不能与到达地点相
同！');</script>");
                }
                else
                {
                if(this.RadioButtonList1.SelectedValue=="单程")
                {
                    //查询满足条件的航班
                    this.DataGrid1.DataKeyField="planeid";
                    SqlConnection con=Class.connection();//数据库连接
                    con.Open();
                    this.DataGrid1.DataSource=Class.ExeSelect("select * from plane join planeseattype  on
plane.planeid=planeseattype.planeid where planestartplace='"+this.SPDropDownList.SelectedValue+"'and
planeendplace='"+this.EPDropDownList.SelectedValue+"' and
planest='"+this.STDropDownList.SelectedValue+"'and convert(varchar,planestarttime,120) like
'%"+this.STTextBox.Text+"%'");
                    this.DataGrid1.DataBind();
                    this.Datagrid2.Visible=false;
                    con.Close();
                }
                    if(this.RadioButtonList1.SelectedValue=="往返")//往返
                    {
                        DateTime starttime=Convert.ToDateTime(STTextBox.Text);
                        DateTime endtime=Convert.ToDateTime(ETTextBox.Text);
                        if(DateTime.Compare(starttime,endtime)>0)
                        {

                            Response.Write("<script language='javascript'>alert('返回时间小于出发时间!');</
script>");
                        }
                        else
                        {
                            this.DataGrid1.DataKeyField="planeid";
                            SqlConnection con=Class.connection();//数据库连接
                            con.Open();
                            this.DataGrid1.DataSource=Class.ExeSelect("select * from plane join
planeseattype  on plane.planeid=planeseattype.planeid where
planestartplace='"+this.SPDropDownList.SelectedValue+"'and
planeendplace='"+this.EPDropDownList.SelectedValue+"' and
```

```
planest='"+this.STDropDownList.SelectedValue+"'and convert(varchar,planestarttime,120) like
'%"+this.STTextBox.Text+"%'");
                                this.DataGrid1.DataBind();
                                this.Datagrid2.DataKeyField="planeid";
                                this.Datagrid2.DataSource=Class.ExeSelect("select * from plane join
planeseattype  on plane.planeid=planeseattype.planeid where
planestartplace='"+this.EPDropDownList.SelectedValue+"'and
planeendplace='"+this.SPDropDownList.SelectedValue+"' and
planest='"+this.STDropDownList.SelectedValue+"'and convert(varchar,planestarttime,120) like
'%"+this.ETTextBox.Text+"%'");
                                this.Datagrid2.DataBind();
                                this.Datagrid2.Visible=true;
                                con.Close();
                            }

                        }
                    }

                }



            private void DataGrid1_ItemCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e)
            {



                if(e.CommandName=="AddToOrder")
                {
                    if(Session["flag"]==null)
                    {
                        Response.Redirect("login.aspx");
                    }
                    else
                    {
                        if(this.RadioButtonList1.SelectedValue=="单程")
                        {

                            int OrderNumber=int.Parse(((TextBox)
```

```
(e.Item.Cells[9].FindControl("textbox1"))).Text);
                    string PlaneId=this.DataGrid1.DataKeys[e.Item.ItemIndex].ToString();//插入订单
表
                    decimal PlanePrice=decimal.Parse(((Label)
(e.Item.Cells[8].FindControl("PlanePrice"))).Text);
                    decimal TotalPrice=PlanePrice*OrderNumber;
                    UserJudge user=(UserJudge)Session["user"];
                    SqlConnection con=Class.connection();//连接数据库
                    con.Open();
                    SqlCommand com=new SqlCommand("select userid from users where
username='"+user.username+"'",con);
                    string UserId=com.ExecuteScalar().ToString();
                    string OrderState=Convert.ToString("未确认订单");
                    SqlCommand com1=new SqlCommand("insert into orders
(userid,planeid,planest,ordernumber,orderprice,orderstate) values
('"+UserId+"','"+PlaneId+"','"+this.STDropDownList.SelectedValue+"','"+OrderNumber+"','"+TotalPrice+"','"+
OrderState+"')",con);
                    com1.ExecuteNonQuery();
                    con.Close();
                    Response.Redirect("order.aspx");
                }
                else
                {

                    int OrderNumber=int.Parse(((TextBox)
(e.Item.Cells[9].FindControl("textbox1"))).Text);
                    string PlaneId=this.DataGrid1.DataKeys[e.Item.ItemIndex].ToString();//插入订单
表
                    decimal PlanePrice=decimal.Parse(((Label)
(e.Item.Cells[8].FindControl("PlanePrice"))).Text);
                    decimal TotalPrice=PlanePrice*OrderNumber;
                    UserJudge user=(UserJudge)Session["user"];
                    SqlConnection con=Class.connection();//连接数据库
                    con.Open();
                    SqlCommand com=new SqlCommand("select userid from users where
username='"+user.username+"'",con);
                    string UserId=com.ExecuteScalar().ToString();
                    string OrderState=Convert.ToString("未确认订单");
                    SqlCommand com1=new SqlCommand("insert into orders
(userid,planeid,planest,ordernumber,orderprice,orderstate) values
('"+UserId+"','"+PlaneId+"','"+this.STDropDownList.SelectedValue+"','"+OrderNumber+"','"+TotalPrice+"','"+
```

```
OrderState+"')",con);
                    com1.ExecuteNonQuery();
                    con.Close();
                }


            }


        }
    }

    private void Datagrid2_ItemCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e)
    {



        if(e.CommandName=="AddToOrder1")
        {
            if(Session["flag"]==null)
            {
                Response.Redirect("login.aspx");
            }
            else
            {
                //插入订单表
                int OrderNumber=int.Parse(((TextBox)
(e.Item.Cells[9].FindControl("textbox2"))).Text);//预定数量
                string PlaneId=this.Datagrid2.DataKeys[e.Item.ItemIndex].ToString();
                decimal PlanePrice=decimal.Parse(((Label)
(e.Item.Cells[8].FindControl("PlanePriceLabel"))).Text);
                decimal TotalPrice=PlanePrice*OrderNumber;
                UserJudge user=(UserJudge)Session["user"];
                SqlConnection con=Class.connection();//连接数据库
                con.Open();
                SqlCommand com=new SqlCommand("select userid from users where
username='"+user.username+"'",con);
                string UserId=com.ExecuteScalar().ToString();
                string OrderState=Convert.ToString("未确认订单");
```

```
                    SqlCommand com1=new SqlCommand("insert into orders
(userid,planeid,planest,ordernumber,orderprice,orderstate) values
('"+UserId+"','"+PlaneId+"','"+this.STDropDownList.SelectedValue+"','"+OrderNumber+"','"+TotalPrice+"','"+
OrderState+"')",con);
                    com1.ExecuteNonQuery();
                    con.Close();

                }
                Response.Redirect("order.aspx");
            }









        }


        private void RadioButtonList1_SelectedIndexChanged(object sender, System.EventArgs e)
        {
            if(this.RadioButtonList1.SelectedValue=="单程")
            {
                this.ETLabel.Visible=false;
                this.ETTextBox.Visible=false;
                this.CompareValidator2.Visible=false;
                this.RequiredFieldValidator2.Visible=false;
                this.CompareValidator2.Enabled=false;
                this.RequiredFieldValidator2.Enabled=false;

            }
            else
            {
                this.ETLabel.Visible=true;
                this.ETTextBox.Visible=true;
                this.CompareValidator2.Visible=true;
                this.RequiredFieldValidator2.Visible=true;
                this.CompareValidator2.Enabled=true;
                this.RequiredFieldValidator2.Enabled=true;
```

```
        }

    }

    private void DataGrid1_PageIndexChanged(object source,
System.Web.UI.WebControls.DataGridPageChangedEventArgs e)
    {
        this.DataGrid1.CurrentPageIndex=e.NewPageIndex;
        if(this.RadioButtonList1.SelectedValue=="单程")
        {
            //查询满足条件的航班
            this.DataGrid1.DataKeyField="planeid";
            SqlConnection con=Class.connection();//数据库连接
            con.Open();
            this.DataGrid1.DataSource=Class.ExeSelect("select * from plane join planeseattype  on
plane.planeid=planeseattype.planeid where planestartplace='"+this.SPDropDownList.SelectedValue+"'and
planeendplace='"+this.EPDropDownList.SelectedValue+"' and
planest='"+this.STDropDownList.SelectedValue+"'and convert(varchar,planestarttime,120) like
'%"+this.STTextBox.Text+"%'");
            this.DataGrid1.DataBind();
            this.Datagrid2.Visible=false;
            con.Close();
        }
        if(this.RadioButtonList1.SelectedValue=="往返")//往返
        {
            this.DataGrid1.DataKeyField="planeid";
            SqlConnection con=Class.connection();//数据库连接
            con.Open();
            this.DataGrid1.DataSource=Class.ExeSelect("select * from plane join planeseattype  on
plane.planeid=planeseattype.planeid where planestartplace='"+this.SPDropDownList.SelectedValue+"'and
planeendplace='"+this.EPDropDownList.SelectedValue+"' and
planest='"+this.STDropDownList.SelectedValue+"'and convert(varchar,planestarttime,120) like
'%"+this.STTextBox.Text+"%'");
            this.DataGrid1.DataBind();
            this.Datagrid2.DataKeyField="planeid";
            this.Datagrid2.DataSource=Class.ExeSelect("select * from plane join planeseattype  on
plane.planeid=planeseattype.planeid where planestartplace='"+this.EPDropDownList.SelectedValue+"'and
planeendplace='"+this.SPDropDownList.SelectedValue+"' and
planest='"+this.STDropDownList.SelectedValue+"'and convert(varchar,planestarttime,120) like
'%"+this.ETTextBox.Text+"%'");
```

```
                this.Datagrid2.DataBind();
                this.Datagrid2.Visible=true;
                con.Close();
            }


        }

        private void Datagrid2_PageIndexChanged(object source,
System.Web.UI.WebControls.DataGridPageChangedEventArgs e)
        {
            this.Datagrid2.CurrentPageIndex=e.NewPageIndex;
            if(this.RadioButtonList1.SelectedValue=="单程")
            {
                //查询满足条件的航班
                this.DataGrid1.DataKeyField="planeid";
                SqlConnection con=Class.connection();//数据库连接
                con.Open();
                this.DataGrid1.DataSource=Class.ExeSelect("select * from plane join planeseattype  on
plane.planeid=planeseattype.planeid where planestartplace='"+this.SPDropDownList.SelectedValue+"'and
planeendplace='"+this.EPDropDownList.SelectedValue+"' and
planest='"+this.STDropDownList.SelectedValue+"'and convert(varchar,planestarttime,120) like
'%"+this.STTextBox.Text+"%'");
                this.DataGrid1.DataBind();
                this.Datagrid2.Visible=false;
                con.Close();
            }
            if(this.RadioButtonList1.SelectedValue=="往返")//往返
            {
                this.DataGrid1.DataKeyField="planeid";
                SqlConnection con=Class.connection();//数据库连接
                con.Open();
                this.DataGrid1.DataSource=Class.ExeSelect("select * from plane join planeseattype  on
plane.planeid=planeseattype.planeid where planestartplace='"+this.SPDropDownList.SelectedValue+"'and
planeendplace='"+this.EPDropDownList.SelectedValue+"' and
planest='"+this.STDropDownList.SelectedValue+"'and convert(varchar,planestarttime,120) like
'%"+this.STTextBox.Text+"%'");
                this.DataGrid1.DataBind();
                this.Datagrid2.DataKeyField="planeid";
                this.Datagrid2.DataSource=Class.ExeSelect("select * from plane join planeseattype  on
plane.planeid=planeseattype.planeid where planestartplace='"+this.EPDropDownList.SelectedValue+"'and
```

```
planeendplace='"+this.SPDropDownList.SelectedValue+"' and
planest='"+this.STDropDownList.SelectedValue+"'and convert(varchar,planestarttime,120) like
'%"+this.ETTextBox.Text+"%'");
                    this.Datagrid2.DataBind();
                    this.Datagrid2.Visible=true;
                    con.Close();
                }
```

Orders.aspx.cs

```
private void Page_Load(object sender, System.EventArgs e)
        {


            if(Session["flag"]==null)
            {
                Response.Redirect("login.aspx");
            }
            else
            {

                UserJudge user=(UserJudge)Session["user"];
                LoginJudgeLabel.Text="欢迎"+user.username+"光临本网站";
            }

            if(!this.IsPostBack)

            {
                UserJudge u=(UserJudge)Session["user"];
                SqlConnection con=Class.connection();//数据库连接
                con.Open();
                this.DataGrid1.DataSource=Class.ExeSelect("select * from users join orders on
users.userid=orders.userid join plane on  orders.planeid=plane.planeid join planeseattype on
plane.planeid=planeseattype.planeid  where username='"+u.username+"' and
orders.planest=planeseattype.planest ");
                this.DataGrid1.DataKeyField="orderid";
                this.DataGrid1.DataBind();
                con.Close();
            }
        }

        #region Web 窗体设计器生成的代码
        override protected void OnInit(EventArgs e)
        {
```

```
            //
            // CODEGEN: 该调用是 ASP.NET Web 窗体设计器所必需的。
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// 设计器支持所需的方法 - 不要使用代码编辑器修改
        /// 此方法的内容。
        /// </summary>
        private void InitializeComponent()
        {
            this.DataGrid1.ItemCommand += new
System.Web.UI.WebControls.DataGridCommandEventHandler(this.DataGrid1_ItemCommand);
            this.DataGrid1.ItemDataBound += new
System.Web.UI.WebControls.DataGridItemEventHandler(this.DataGrid1_ItemDataBound);
            this.DataGrid1.SelectedIndexChanged += new
System.EventHandler(this.DataGrid1_SelectedIndexChanged);
            this.Load += new System.EventHandler(this.Page_Load);

        }

        private void DataGrid1_ItemCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e)
        {
            if(e.CommandName=="CancelOrder")//选择取消订单

            {
                string OrderId=this.DataGrid1.DataKeys[e.Item.ItemIndex].ToString();
                UserJudge u=(UserJudge)Session["user"];
            SqlConnection con=Class.connection();
                con.Open();
                SqlCommand com=new SqlCommand("select ordernumber from orders join users on
orders.userid=users.userid join plane on  orders.planeid=plane.planeid join planeseattype on
plane.planeid=planeseattype.planeid  where username='"+u.username+"' and
orders.planest=planeseattype.planest and orderid='"+OrderId+"'",con);
                string OrderNumber=com.ExecuteScalar().ToString();
                SqlCommand com1=new SqlCommand("select ordertime from orders join users on
orders.userid=users.userid join plane on  orders.planeid=plane.planeid join planeseattype on
plane.planeid=planeseattype.planeid  where username='"+u.username+"' and
```

```
orders.planest=planeseattype.planest and orderid='"+OrderId+"'",con);
                    string  OrderTime=com1.ExecuteScalar().ToString();
                if(OrderTime=="")//没有确认订单
                {

                    SqlCommand com2=new SqlCommand("delete from orders where
orderid='"+OrderId+"'",con);
                    //建立适配器
                    SqlDataAdapter sda = new SqlDataAdapter();
                    //建立dataset,用于填充
                    DataSet ds = new DataSet();
                    sda.SelectCommand = com2;
                    //传递查询结果
                    sda.Fill(ds,"orders");
                    this.DataGrid1.DataSource=ds.Tables["orders"];
                    this.DataGrid1.DataKeyField="orderid";
                    this.DataGrid1.DataBind();
                    this.DataGrid1.EditItemIndex=-1;//返回上一级页面
                    this.DataGrid1.DataSource=Class.ExeSelect("select * from users join orders on
users.userid=orders.userid join plane on  orders.planeid=plane.planeid join planeseattype on
plane.planeid=planeseattype.planeid  where username='"+u.username+"' and
orders.planest=planeseattype.planest ");
                    this.DataGrid1.DataKeyField="orderid";
                    this.DataGrid1.DataBind();
                }
                else{
                    SqlCommand com3=new SqlCommand("update planeseattype set
planenumber=planenumber+'"+OrderNumber+"' from planeseattype join plane on
planeseattype.planeid=plane.planeid join orders on plane.planeid=orders.planeid  where
planeseattype.planest=orders.planest and orders.planeid=planeseattype.planeid and
orderid='"+OrderId+"'",con);
                    com3.ExecuteNonQuery();
                    SqlCommand com4=new SqlCommand("delete from orders where
orderid='"+OrderId+"'",con);
                    //建立适配器
                    SqlDataAdapter sda = new SqlDataAdapter();
                    //建立dataset,用于填充
                    DataSet ds = new DataSet();
                    sda.SelectCommand = com4;
                    //传递查询结果
                    sda.Fill(ds,"orders");
```

```
                this.DataGrid1.DataSource=ds.Tables["orders"];
                this.DataGrid1.DataKeyField="orderid";
                this.DataGrid1.DataBind();
                this.DataGrid1.EditItemIndex=-1;//返回上一级页面
                this.DataGrid1.DataSource=Class.ExeSelect("select * from users join orders on
users.userid=orders.userid join plane on  orders.planeid=plane.planeid join planeseattype on
plane.planeid=planeseattype.planeid  where username='"+u.username+"' and
orders.planest=planeseattype.planest ");
                this.DataGrid1.DataKeyField="orderid";
                this.DataGrid1.DataBind();
            }
            con.Close();
        }

            if(e.CommandName=="SubmitOrder")//确认订单
            {

                DateTime dt1 = DateTime.Now;
                //                string strDate = dt1.ToString("yyyy-MM-dd");
                //                DateTime dt2 = DateTime.Parse("02:00:00");
                //                DateTime strTime = dt1.AddTicks(dt2.Ticks);
                string OrderId=this.DataGrid1.DataKeys[e.Item.ItemIndex].ToString();
                UserJudge user=(UserJudge)Session["user"];
                SqlConnection con=Class.connection();//连接数据库
                con.Open();
                SqlCommand com=new SqlCommand("select ordernumber from orders join users
on orders.userid=users.userid join plane on  orders.planeid=plane.planeid join planeseattype on
plane.planeid=planeseattype.planeid  where username='"+user.username+"' and
orders.planest=planeseattype.planest and orderid='"+OrderId+"'",con);
                string OrderNumber=com.ExecuteScalar().ToString();
                SqlCommand com1=new SqlCommand("update planeseattype set
planenumber=planenumber-'"+OrderNumber+"' from planeseattype join plane on
planeseattype.planeid=plane.planeid join orders on plane.planeid=orders.planeid  where
planeseattype.planest=orders.planest and orders.planeid=planeseattype.planeid and
orderid='"+OrderId+"'",con);
                com1.ExecuteNonQuery();
                string OrderState=Convert.ToString("已确认订单");
                SqlCommand com2=new SqlCommand("update orders set ordertime='"+dt1+"',
orderstate='"+OrderState+"' where orderid='"+OrderId+"'",con);
                com2.ExecuteNonQuery();
                this.DataGrid1.DataSource=Class.ExeSelect("select * from users join orders on
```

```
users.userid=orders.userid join plane on  orders.planeid=plane.planeid join planeseattype on
plane.planeid=planeseattype.planeid  where username='"+user.username+"' and
orders.planest=planeseattype.planest ");
                        this.DataGrid1.DataKeyField="orderid";
                        this.DataGrid1.DataBind();
                              con.Close();

//                        ((Label)e.Item.Cells[12].FindControl("SubmitSuccess")).Text="提交成功,请于"+strDate+"
"+strTime.ToString("HH:mm:ss")+"前付款";









                    }
                }
private void DataGrid1_SelectedIndexChanged(object sender, System.EventArgs e)
                {

                }

                private void DataGrid1_ItemDataBound(object sender,
System.Web.UI.WebControls.DataGridItemEventArgs e)
                {

                        if(e.Item.ItemIndex   >   -1)
                        {
                              ((LinkButton)(e.Item.Cells[14].Controls[0])).Attributes.Add("onclick","return
confirm('确认删除吗？');");
                        }
                    }
```

外文文献译文

一 介绍 SQL Server 和连接字符串

一个希望有多个同时连接用户的站点必须使用一个比 Access 合适的数据源，Access 并不真的为高性能应用程序服务。接下来将说明如何从 Microsoft SQL Server 这种企业级关系数据库管理系统（RMDS）获取数据。

SQL Server 完全版包括了三个部分。第一部分是引擎，用于组织数据及针对命令进行读取和写入。第二部分是用于对数据库操作的开发人员的工具，例如 Query Analyzer 和 Data Transformation Services。最后一部分是用于管理数据的工具，包括备份程序和复制模式。

虽然完全版对大型企业来讲有着不可估量的好处，但是许多开发人员并不需要全套工具。幸运的是，微软提供了一个 SQL Server 引擎的免费版，叫做 SQL Server Express（SSE）。

虽然它具有一个使用 T-SQL 命令来导入模式和数据的命令行工具（osql.exe）,但是它还不包括在 SQL Server 完全版中具有的丰富图形化的工具。不过，你可以方便地使用 Visual Studio 或 Visual Web Developer 来使用 SSE 开发数据库。SSE 有一个限制，只处理本地的连接（不可能在 Web 服务器以外的不同机器上运行 SSE）。对于具有某些嗜好的人和学生的网站来说，SSE 是一个极佳的选择。你可能会使用基于 SQL Server2000 的 MSDE，而 SSE 基于 SQL Server 的 Yukon 版。SSE 将在全书中使用，它的下载说明在第 1 章中已经给出。

除非有特别说明，这里所介绍的所有技术均可应用于 SQL Server 的三种形式 （完全版产品，SSE 和 MSDE），因此，专业术语 SQL Server 包括了这三种形式。

因为 SSE 只是一个引擎而并无内置的开发工具（到 2004 年），所以必须使用其他的工具创建、修改、或者填充数据库。为了这个目的我们要使用的主要工具是 Visual Studio 或者 Visual Web Developer，提供了 Database Explorer 作为其界面的一部分。请注意 Database Explorer 可以更改本地数据库中的数据和模式，但仅仅能更改远程数据库中的数据（而非结构）。在 Visual Studio 中，Database Explorer 被称作 Server Explorer，因为它包括了一些处理非数据库服务器的附加功能。我们在本书的开始简要地使用 osql.exe 命令行工具来导入将在本书通篇使用的初始数据库。修改数据库结构的第三种方式是打开 Access，然后在 SSE 数据库中链接一个外部表。

使用 SQL Server 需要熟悉一些词汇。SQL Server 被安装在一台称为服务器的机器上，用其机器名来引用，如果 SQL Server 和请求软件在同一台机器上，这台机器可以用（Local）来引用。引擎可以在一台机器上安装多次，每一次安装叫做一个实例。SSE 安装一个名为（Local）\SQL Express 的实例。在一个实例内，你就可以创建数据库。数据库有具有字段和记录的表。数据库有供数据使用者使用的一组表、字段和约束，称为视图。存储过程（SPROCs）可以对数据执行任务。一个 SQL Server 实例自动安装一个用户名为 sa 的帐户，这个账户当对用户使用 SQL Server 认证时具有对所有对象进行所有动作的权限。当使用 Windows 认证（也称混合认证）时，登录进 Windows 的用户也同时登录进了 SSE。更多关于认证方面的信息将在稍后介绍。

1 使用 SQL Server 数据库的准备

在研究数据库上花时间将会减少你设计用数据的页面时产生的错误。在编写使用 SQL Server 的页面前，请检查你是否已掌握了如下几类信息:

服务器，实例和数据库名称------确认服务器名、实例和数据库名的拼写正确。如果在服务器上只有一个完全版 SQL Server 的实例，你不需要使用实例名。但是即使在服务器上只有一个 SSE 的实例，你也得用 My Server\SQL Express 来显式地引用。明确是测试实际数据还是测试数据库的部署备份。

安全信息------你需要知道自己的用户 ID 和密码，以便对开发数据库的访问进行认证。同时请确认

SSE 是使用 Windows 认证还是 SQL 认证（第 1 章的安装指明 SSE 应使用 Windows 认证）。

数据库模式------理解数据库模式。获得表和字段名、自动生成或锁定的字段、相关性以及约束的确切拼写。特别要注意对象名的下划线和空格。从管理员找出是直接使用表还是使用视图或者存储过程（SPROCs）。后者可能需要特殊数据类型的参数。检查元数据的 SQL 语法将出现在后面。

测试 SQL 语句（可选）------你可能对 SQL 语句的语法或逻辑有疑问。你将会发现使用例如 SQL 的查询分析器之类的开发工具测试语句要比在这些语句第一次出现在 ASPX 页面上时才测试要有效的多。

2 连接字符串

AccessDataSource 控件（在最后一章讨论）和 SqlDataSource 控件之间语法主要的不同是应用数据库的方式不同。对 MDB 来说，我们只提供文件名和路径名。SqlDataSource 使用连接字符串将服务器名、数据库名和登录信息连接起来。连接字符串的语法和我们通常所熟知的 Visual Basic 或 C#的语法有所区别，这也是许多学生犯错误的原因。一个典型的连接字符串如下：

```
ConnectionString=”Server=MyServer;Database=MyDatabase;User ID=MyID;Password=MyPass”
```

改写成多行格式能提高可读性，如下：

```
  ConnectionString=”
    Server=MyServer;
    Database=MyDatabase;
    User ID=MyID;
    Password=MyPass”
```

首先，注意语法。源代码中整个字符串处于双引号之中。在 VWD 的属性窗口指定字符串时，你不需要引号；VWD 会自动添加。在引号中间是一些诸如 Criteria=Value 格式的成对信息。分号将这些信息分隔。注意引号不被用于值。还要注意即使判句中有空格，（例如 User ID），它们也不需要引号或括号。这种语法不难理解，但是因为与 VB、C#和 SQL 语言不同，犯错误是很平常的。让我们花点时间将这个字符串拆解开来。

在这个连接字符串中有两个值：数据库标识符（服务器、实例和数据库）和安全值（用户 ID、密码和安全设置）。数据库标识符以服务器值开始，该值是运行 SQL Server 的机器的网络名称。（在 xp 中机器名可通过开始->我的电脑;右键点击并选择属性->计算机名->完整的计算机名称来获得。在 Windows2000 中可以在桌面的我的电脑上单击右键->属性->网络认证）。如果知道数据库服务器与 ASP.NET 运行的网络服务器使用同一台机器，可以将服务器名称指定为（Local）来表明服务器是 ASP.NET 页面运行时的本地机器。可以用 Instance=My Instance 来添加实例。更普通的是，使用 MyServer\My-Instance 或者（Local）\MyInstance 来添加实例到服务器名。可能还会看到使用句点（句号）来表示本地机器，例如\SQL Express。

在默认情况下，SSE 安装自己的实例。单独引用（local）将导致失败。你必须使用（local）\SQL Express 来引用 SSE。

安全设置在稍后讨论。现在，了解 Windows 认证，使用属性 IntegratedTurusted_connection=true 代替用户 ID 和密码属性。SQL 认证需要两个值：user=MyUserName;password=MyPassword,MyUserName 和 MyPassword 被你自己的证书代替。本文使用 Windows 认证。

如果你熟悉 ASP 早期版本，可能对提供程序感到疑惑。默认提供程序是 .NET FrameWork Data Provider for SQL Server。

二 了解 SQL Server 的安全性

SQL Server 的安装有两种安全模式选择。它们之间的区别在于哪种软件执行认证。认证是检验想要连接到 SQL Server 的用户身份的过程。一旦执行了认证，SQL Server 便能验证该用户是否具有连接一个

被请求的资源的许可，例如数据库。如果用户有连接数据库的许可，SQL Server 允许连接请求成功。否则，就失败。这个验证用户许可的过程称作认证。

Windows 认证（通常被称为信任认证或者完整性安全）使用进行连接请求过程的 Windows 用户身份执行对数据库的认证。这种情况下，连接字符串不需提供显式的用户名和密码。ASP.NET 以一个”ASPNET”的本地用户来运行（或者在 IIS6.0 中以一用户名为”Network Service”来运行），所以当使用 Windows 认证时，SQL 将检查该用户是否具有使用数据库的许可。在这种情况下，所有 ASP.NET 应用程序以这个相同的用户名来运行，所以安全模式对这些应用程序不区别对待。虽然可以在单独的 ASP.NET 进程中运行每一个应用程序（单独的用户运行每一个程序），或者模拟有连接请求的浏览器客户的 Windows 用户身份，这些内容超出了本书的范畴。客户设想的模拟情况然而是在 Web 应用程序中关于 Windows 认证最常见的使用方式。

SQL 认证对 SQL Server 内配置的用户来检查显式提供的用户名和密码（不涉及操作系统）。在这种情况下，对于在 ASP.NET 进程中运行的每一个应用程序来说以单独的证书来连接数据库是很容易的。这样就把应用程序合理地隔开了（应用程序 A 没有应用程序 B 的用户名和密码便不能连接到应用程序 B 的数据库）。这是部署的 Web 应用程序的最普遍的认证模式，特别是在共享宿主的情况下。它的小缺陷是要应用程序保留用于连接的用户帐户的密码，并且如果恶意的人窃取了该密码，将会危及数据库的安全。但是，你将在本书稍后看到 ASP.NET 提供了一种以加密的格式保存在 Web.config 文件中存储 SQL 认证密码的安全方式，这样就降低了密码被获取的风险。

混合模式是 SQL Server 的配置，它既允许 Windows 认证，又允许 SQL 认证。

在安装 SQL Server 或 SSE 时，你就需要选饿一种认证模式。在 SQL Server 中，安装步骤有向导帮助。而在 SSE 中，Windows 认证是默认方式，要想安装 SQL 认证，必须显式地配置。本文采用 Windows 认证。

如果你的 SQL Server 或 SSE 安装完毕，就能打开注册表查看指定的认证模式（当然需要备份），找到 HKey_Local_Machine/Software/Microsoft SQL Server 并搜素 LoginMode。值为 1 的注册子键表示 Windows 认证，值为 2 表示混合认证模式。

这些模式之间的区别被总结于下表

	Windows 认证	SQL 认证（用于本书）
替换名称	信任认证 完整性安全	没有，混合模式认证允许使用 Windows 或 SQL 认证
典型环境	内网	因特网
用户和认证过程列表的位置	Windows	SQL Server
SSE 安装	默认安装	需要指定安装
连接字符串	Trusted_connection=true 或 Integrated Security=true	user=username; password=password
ASP.NET Web 应用程序的用户	ASP.NET 进程、ASPNET（IIS 5.x）或 Network Service（IIS 6）	SQL 用户
实力	较好的安全性；对用户在 SQL 事件和 Windows 事件中的活动进行跟踪	无需创建新帐户即可在宿主机上部署；独立于操作系统宿主的内网站点只需普通技术为应用程序提供更加灵活的方式以不同的证书来连接每个数据库
缺点	Web 应用程序的 Windows 认	密码存储在 Web 应用程序中

	证有可能会将 OS 中的权限范围设置过大	（在 Windows 认证中则不是）。确认密码存储在 Web.config 文件中并已加密。允许使用 sa 证书的 Web 应用程序的低级操作。总是为 ASP.NET Web 应用程序创建新的证书并给予所需权限。

现在，知道了 SQL 使用安全的方式，我们考虑数据使用者（DataSource 控件）将如何满足需求。首先，使用从 VWD 和 VWD Web 服务器（Cassini）获取的数据，主要是在设计和测试阶段。第二，在部署后你应该从 IIS 访问数据。这两个数据使用者有不同的用户名。VWD 和 VWD Web 服务器使用登录进 Windows 的人员的名称，IIS 程序使用名称 ASP.NET。

如果 SQL Server 使用 Windows 认证，SqlDataSource 控件需要在连接字符串中包含以下代码：Integrated Security=true（或 Trusted_connection=true）。这个参数将指示 SQL Server 基于请求者的 Windows 登录帐户对数据请求进行认证。如果你是安装 SSE 的登录用户，证书将被授予访问 SSE 的权限。使用 VWD 和 VWD Web 服务器将一切顺利，因为 VWD Web 服务器的用户被认为是登陆到 Windows 的程序员并且拥有 SSE 帐户。然而，即使应用程序在 VWD 之内运行正常，当站点移至 IIS 后，也不会正常。IIS 是在名为 ASPNET 的用户帐户下运行（在 IIS6/Windows2003 Server 中的 Network Service）。因此，运行 IIS 机器的管理员必须给 ASP.NET 添加用户并授予其许可。这个过程超出了本书的范围，在许多 IIS 管理员的手册中有详细解释。总而言之，如果你的 SQL Server 使用的是 Windows 认证，就能使用 VWD 和 VWD Web 服务器做本书的练习。只有在授予访问数据库的 ASP.NET 进程帐户许可后，你的页面才可以在 IIS 上运行。

如果你的 SQL Server 使用 SQL 认证，SQL 将使用自己的认证。这个过程不依靠 Windows 的用户列表。连接字符串包括两个参数：user=username；password=password。你可以从 VWD、VWD Web 服务器或 IIS 中使用页面，因为在 Windows 中没有创建用户帐户的需要。但是，我们还是需要使用 SQL server 的帐户。惟一的默认帐户是 sa。在部署之前，你应该创建 SQL Server 的另一个帐户，该帐户只有执行 ASPX 页面的权限。如果不创建 sa 以外的替换帐户（密码保护 sa）将会使你的站点处于最知名、最易于利用的安全漏洞当中。任何黑客都知道尝试空密码的 userID='sa'来登录。

对于这两种认证模式，当使用前面提及的连接字符串时，用户将以初始帐户登录 SQL Server。帐户 sa 表示系统管理员，从名称上看出，它具有对所有对象的所有权限。在 SQL Server 的当前版本中，你不能使用密码为空的 sa 帐户来安装服务。在 SSE 中，必须以参数 SAPWD='MyStringPassword'来安装。安全性高的密码表示至少不为空。最后使用不少于 7 位的字符并使用字母、数字和符号的混合形式。在学生练习之外的大多数情况下，需要为每个数据库和应用程序设置一个指定帐户。避免一个应用程序具有访问其他应用程序数据的权限。

三 探究新奇的数据库的结构

在使用数据库之前，了解数据库的结构（模式）很重要。至少，你要知道将要使用的表或查询的名称，以及它们的列（字段）名。如果你打算写入或改变数据，你应该了解关系以便数据依赖不被破坏。注意，如果这是你第一次查看数据库结构，你将会看到比你想象更多的信息，但是，很容易忽视无关信息。几万年前人类的智力便能将麦子和糠分离出来；他们也能将谷物转化成代码。

首先，考虑除 ASP.NET2.0 页面之外的可以使用的工具来帮助进行探索。例如，如果使用完全版的 SQL Server，就有了 SQL 企业管理器，用于显示和浏览数据库组件。完全版的 Visual Studio 也有 QueryBuilder（在 Beta2 中，有数据库模式设计器），微软希望在 SQL Express 中添加一个名为

XM（Express Manager）作为 Web 下载的工具。即使在 VWD 中，我们已经实践过如何使用数据探测器添加连接并发现数据库的表和列。但是如果你没有任何可用工具，或者想探究更深，可以使用 SQL 命令来查看模式。程序员通常在一些临时页面上这样做；最后，这并不是部署的一部分。开始这些语句会有一个小窍门，但是它们是按照一个合理的形式来执行的。在 Kauffman 等人著的 Beginning SQL Programming(ISBN 1-861001-80-0)的第 16 章中，对这一主题进行了深入的讲解，包括所有的术语。

SQL Server 包括了模式表，用于描述它们包括的数据的结构，一旦知道了表和列的名称，就可以像读取其他数据一样读取模式表。下表列出了最常用的模式表。

模式表名（所有以 INFORMATION_SCHEMA 开头的模式名）	最常用的列
.SCHEMATA	Catalog_Name（数据库名）
.TABLES	Table_Name Table_Catalog(哪个数据库包含该表) Table_Type(基表或视图)
.COLUMNS(显示所有表的所有列)	Column_Name Table_Name(列的位置) Data_Type Column_Default
.VIEWS	Table_Name(实际上这就是视图名) View_Definition
.ROUTINES(存储过程)	Routine_Name Routine_Definition

外文文献原文

Introduction to SQL Server and Connection Strings

A site that expects to have more than a few simultaneous users will have to employ a more scalable datasource than Access, which isn't really intended for high-performance applications. I will explain how to use data from Microsoft SQL Server an enterprise-strength Relational Database Management System (RDMS) later.
The full version of SQL Server includes three main parts. First is the engine that actually organizes the data and reads or writes in response to commands. Second is a suite of developer's tools to work with the database, such as Query Analyzer and Data Transformation Services. Last are the tools for managing data, ranging from backup utilities to replication schemes.
Although the full version has invaluable benefits to large-scale enterprises, many developers don't need this entire suite of tools. Fortunately Microsoft makes available a free version of the SQL Server engine called SQL Server Express (SSE). Although it comes with a command-line tool (osql.exe) for importing schema and data using T-SQL commands, it does not yet include the rich graphical tools included with the full version of SQL Server (keep posted — it probably will by mid-2005). However, you can easily use Visual Studio or Visual Web Developer to develop databases using SSE. SSE enforces a restriction that only local connections are served (it is not possible to run SSE on a different machine than the Web server. For many hobbyist-level or student Web sites, SSE is a perfectly suitable option. You may have worked with an older incarnation named MSDE, which was based on SQL Server 2000; SSE is based on the Yukon version of SQL Server. SSE is used throughout this book; the instructions for downloading itare explained in Chapter 1.
Unless specifically denoted otherwise, all of the techniques presented here apply for all three forms of SQL Server (full product, SSE and the older MSDE); thus, the generic term SQL Server encompasses all three.
Because SSE is only an engine and does not come with built-in developer tools (as of 2004), you must use other tools to create, change, or populate a database. The primary tool we will use for this purpose is Visual Studio or Visual Web Developer, which provides a Database Explorer as part of its interface. Note that the Database Explorer can change data and schema in local databases, but it can change only the data in remote databases (not the structure). In Visual Studio, the Database Explorer is called the Server Explorer because it includes some additional capabilities for working against non-database servers. We also use the osql.exe command-line tool briefly in the beginning of this book to import the initial databases we will employ throughout this book. A third alternative for database structure modification is to open Access and link to an external table in your SSE database.
Working with SQL Server requires familiarity with some vocabulary. SQL Server is installed on a machine that becomes a server and can be referenced by its machine name. If SQL Server is on the same machine as the requesting software, the machine can be referred to as (local). The engine can be installed more than once on a machine, and each installation is called an instance. SSE installs as an instance named(local)\SQLExpress. Within an instance, you can create databases. Databases have tables with fields and records. Databases can also have views that are a set of tables, fields, and constraints available to data consumers. Stored procedures (SPROCs) can carry out tasks on the data. A SQL Server instance auto-matically installs one account for a user named "sa" that has rights to all actions on all objects when using SQL Server authentication for users. When using Windows authentication (also called mixed authentication, the user logged on to Windows is also logged on to SSE. More information on authentication follows later.

Preparing to Use a SQL Server Database

Time spent up front studying your database will reduce mistakes when you design pages to use the
data. Check that you have the following kinds of information in hand before writing a page that uses SQL Server:

□Server, instance, and database names-Confirm the exact spelling of the server name,
instance, and database name. If there is only one instance of the full SQL Server on a server, you

do not need to use an instance name. But even if there is only one instance of SSE on the server, you must refer to it explicitly as MyServer\SQLExpress. Clarify whether you will be testing against live data or a development copy of the database.

□Security information-You will need to know your user ID and password to authenticate access to the database for development. Also, check whether the SSE uses Windows or SQL authentication. (The Chapter 1 install specified that SSE should use Windows authentication.)

□Database schema-Understand the schema of the database. Obtain the exact spelling of tables and field names, autogenerated or locked fields, dependencies, and constraints. Carefully note the presence of underscore characters and spaces in object names. Find out from the administrator-whether you will be using tables directly or whether you will be using views and stored procedures (SPROCs). The latter may require parameters of specific data types. The SQL syntax to check these metadata appears later .

□Test your SQL statements (optional)-You may have doubts about the syntax or logic of your SQL statements. You will find it more efficient to test the statements using a development tool such as SQL's Query Analyzer rather than checking them for the first time in an ASPX page.

Connection Strings

The major difference between the syntax of the AccessDataSource control (discussed in the last chapter) and the SqlDataSource control lies in the way the database to use is specified. For an MDB, we merely provide the file's name and pathname. The SqlDataSource uses a connection string to hold the name of the server, database, and login information. A connection string has different syntax than we are accustomed to in Visual Basic or C# and is the source of mistakes for many students. A typical connection string follows.

```
ConnectionString=”Server=MyServer;Database=MyDatabase;User ID=MyID;Password=MyPass”
```

Alternate formatting on multiple lines improves readability, as in the following:

```
ConnectionString=”
        Server=MyServer;
        Database=MyDatabase;
        User ID=MyID;
        Password=MyPass”
```

First, note the syntax. The entire string resides within double quotation marks in the source code. When specifying the string in the property window of VWD, you do not need the quotes; VWD will add them. Inside the quotes are a series of pairs in the format Criteria=value. Semicolons separate these pairs. Note that quotes are not used around the values. Also note that even though there are spaces in some of the criteria (for example, user ID), they are not encased in quotes or brackets. This syntax is not difficult to understand, but because it is different from the VB, C#, and SQL languages, mistakes are common. Let us take a moment to dissect the component parts of this string.

Within this connection string are two kinds of values: database identifiers (server, instance, and database) and security values (user ID, password, and security settings). The database identifiers start with a server value that is the network name of the machine hosting SQL Server. (The machine name is available in XP at Start ->My Computer; right-click and select Properties ->Computer Name tab ->Full Computer Name, For Windows 2000, right-click My Computer on the desktop ->Properties -> Network Identification.) If you know that the database server will be running on the same machine as the Web server where ASP.NET will run, you can also specify the server name as "(local)" to indicate that the server is the local machine on which the ASP.NET page is running. The instance can be added with Instance=Mylnstance. More commonly the instance is appended to the server name as MyServer\My 工nstance or(local)\My 工nstance. You may also see code that uses a period (full stop) to represent the local machine, as in.\SQLExpress.

SSE, by default, installs with its own instance. Referring to(local)alone will fail.
You must refer to your SSE as(local)\SQLExpress.

Security settings are discussed later . For now, understand that for Windows Authentication, use the attribute IntegratedTrusted_connection=true instead of the user ID and password attributes. SQL authentication requires two values: user=MyuserName; password=MyPassword, where MyUserName and MyPassword would be replaced with your own credentials. This text uses Windows authentication.
If you are familiar with earlier versions of ASP, you may be wondering about the provider. The default provider for the SqlDataSource control is the .NET Framework Data Provider for SQL Server.

Understanding Security in SQL Server

An installation of SQL Server has two options for security schemes. They differ by which software performs the authentication. Authentication is the process of verifying the identity of the user that wishes to connect to SQL Server. Once authentication has been performed, SQL Server can verify whether that user has permission to connect to a requested resource such as a database. If the user has permission to connect to the database, SQL Server allows the connection request to succeed. Otherwise, it fails. This process of verifying permission for a user is sometimes called authorization.

☐ **Windows Authentication** (also called Trusted Authentication or Integrated Security) uses the Windows user identity of the process making the connection request to perform authorization against the database. In this case, the connection string does not provide an explicit username or password. The ASP.NET process runs as a local user named "ASPNET" (or on IIS 6.0 as a user named "Network Service"), so when using Windows Authentication, SQL checks whether this user has permission to use the database. In this case, all ASP.NET applications run as this same user, so this security scheme does not provide isolation between applications. Although it is possible to run each application in a unique ASP.NET process (each as a separate user) or to impersonate the Windows user identity of the browser client making the connection request, these topics are outside the scope of this book. The client impersonation scenario, however, is the most common usage for Windows Authentication in a Web application.

☐ **SQL Authentication** checks an explicitly provided username and password against a set of known users configured within SQL Server (without any reference to the operating system). In this case, it is easily possible for each application running in the ASP.NET process to connect to a database with a unique set of credentials, providing reasonable isolation between applications(application A cannot connect to application B's database without application B's username and password). This is the most common authentication scheme used for deployed Web applications, especially in shared hosting scenarios. Its one minor drawback is that it requires the application to retain the password of the user account used to connect, and there is always a risk that the security of your database could be compromised if a malicious person obtained this pass- word. However, as you will see later in this book, ASP.NET provides a safe way to store the SQL Authentication password in the Web.config file in an encrypted format, which mitigates the risk that the password will be obtained.

☐ **Mixed Mode** is a configuration of SQL Server that allows either Windows Authentication or SQL Authentication.

You select which mode of authentication to use when you install SQL Server or SSE. In SQL Server, there is a wizard to assist in the security setup. For SSE, you will get Windows Authentication by default. To install with SQL Authentication, you must configure it explicitly. This text uses Windows Authentication.

If your SQL Server or SSE is already installed, you can figure out which authentication scheme was specfied by opening RegEdit (back it up, of course) and looking in HKey_Local Machine/software/Microsoft/Microsoft SQL server and searching for LoginMode. A Registry subkey value of 1 means Windows Authentication, and a value of 2 means Mixed Authentication mode.

The differences are summarized in the following table.

	Windows Authentication	SQL Authentication (Used in This Text)
Alternate Name	Trusted Authentication Integrated Security	None, but Mixed Mode Authentication

		allows Windows or SQL Authentication
Typical Environment	Intranet	Internet
Location of List of Users and Authentication Process	Windows	SQL Server
SSE Install	Default	Requires special setup
Connection String	Trusted connection=true Or Integrated Security=true	user=username; password=password
User for ASP.NET Web Apps	ASP.NET process, either ASPNET (IIS5.x) or Network Service (IIS6)	SQL User
Strength	Generally better security; activity of a user can be traced across both SQL events and Windows events	Deploys to host without creating new account; independent from operating system Normal technique for hosted intranet sites More flexibility for applications to use different credentials to connect to each of their databases
Weakness	A Windows credential for a Web app is more likely to accidentally give rights to more in the OS than necessary	Passwords are stored in the Web app(in Windows authentication they are not). Be sure your passwords are stored in the Web.config file and encrypted. Allows poor practice of a Web app using sa credentials. Always create a new credential for the ASP.NET Web app and give it only those rights needed.

Now, with the way SQL employs security, consider how the data consumer (our DataSource control) will meet the requirements. First, there is the use of data from VWD and VWD Web Server (Cassini), mainly during design time and testing. Second, you will want to access the data from IIS after deployment.

These two data consumers have different usernames. VWD and VWD Web Server use the name of the person logged on to Windows. The IIS process uses the name ASPNET.
If your SQL Server uses Windows authentication, your SqlDataSource control needs to include the following in its connection string: Integrated Security=true (or, synonomously, Trusted_connection=true).This parameter will instruct SQL Server to authenticate the request for data based on the requestor's Windows login. If you were the user logged in when you installed SSE, your credentials were given access rights to SSE. Using VWD and VWD Web Server will work fine because the user of VW Web Server is considered the programmer that logged on to Windows and thus has an account on SSE. However, even though your application works inside of

VWD, it may not work when the site is moved to IIS. IIS operates under an explicit user account named ASPNET (or Network Services in IIS6/Windows 2003 Server). Therefore, the administrator of the machine that hosts IIS must add ASP.NET as a user and give it permissions. That procedure is beyond the scope of this book but is explained in detail in most manuals for IIS administrators. In summary if your SQL Server is using Windows authentication, you can do this book's exercises with VWD and VWD Web Server. Your pages will work on IIS only after granting the ASP.NET process account permission to your database.
If your SQL Server uses SQL authentication, SQL will do its own authentication. There is no dependence on Windows' list of users. Include in your connection string the two parameters user=username; password=password. Now you can use your page from VWD, VWD Web Server, or IIS, because there is no requirement for setting up user accounts in Windows. However, we need to use an account in SQL Server. The only default account is sa. Prior to deployment, you should create another account in SQL Server that has only the rights needed to execute your ASPX pages. Failure to create an alternate account to sa (and password protect sa) will expose your site to one of the most widely known and easily exploited security failures. Any hacker knows to try a log-in of userID='sa' with an empty password.
For both authentication schemes, when you use the connection strings previously mentioned, a user is logged in to SQL Server to the only initial account. It is named sa, which stands for systems administrator, and, as the name implies, it has all rights to all objects. With current versions of SQL Server, you can not install the service with an sa having a NULL password. For SSE, you must install with the parameter SAPWD="MyStrongPassword"·Strong password in this case means a minimum of not NULL.Better, use at least seven characters and ensure a mix of letters, numbers, and symbols. In most situations beyond a student exercise, you will want a specific account for each database and application. Avoid an application having rights to the data of any other application.

Discovering the Structure of an Unfamiliar Database

It is important to understand the structure (schema) of a database before you begin to use it. At a minimum, you must know the names of tables or queries that you want to use, as well as the names of their columns (fields). If you intend to write or change data, you should also have an understanding of the relationships so that data dependencies are not broken. Note that if this is the first time you have looked at the structure of a database, you will see more information than you could have imagined. However, it is easy to ignore the extraneous information. Human minds have been separating the wheat from the chaff for tens of thousands of years; they convert well from grain to code.
First, consider tools other than ASP.NET 2.0 pages that might be available to you that can help with discovery For example, if you are using the full SQL Server, you have the SQL Enterprise Manager that can display and navigate across the components of a database. The full Visual Studio also has the QueryBuilder (and in Beta 2, a database schema designer). Microsoft expects to add to SQL Express a tool named XM (Express Manager) as a Web download. Even in our VWD we have practiced how to use the Data Explorer to add a connection and discover a database's tables and columns. But if you don't have any of these tools available, or want to dig deeper, you can discover the schema using SQL commands. Programmers usually do this on some temporary pages; in the end, this is not part of deployment. These statements are a little tricky to start, but they do follow a logical pattern. This topic is covered in depth, including all of the nomenclature, in Chapter 16 of Beginning SQL Programming by Kauffman et al., ISBN 1-861001-80-0.
SQL Server contains schema tables that describe the structure of the data they hold. Schema tables can be read like other data once you know their table and column names. The following list gives those most commonly used.

Schema Table Name (all schema names begin with INFORMATION_ SCHEMA)	Most Useful Columns
.SCHEMATA	Cataloges Name (database name)
.T ABLES	Table Name Table Catalog (which database holds the table) Table_ Type (base table or a view)

.COLUMNS (shows all columns of all tables)	Column Name Table_ Name (location of the column) Data_Type Column Default
.VTEWS	Table_ Name (actually this is the View Name) View Definition
.ROUTINES (stored procedures)	Routine Name Routine Definition