第 6 章　Web 窗体的高级控件

上一章中讲解了 ASP.NET 中常用的基本控件，ASP.NET 不仅提供了常用的基本控件如标签控件、文本框控件等，还提供了高级的 Web 窗体的控件。这些控件能够轻松实现更多在 ASP 开发中难以实现的效果。

6.1　登录控件

对于目前常用的网站系统而言，登录功能是必不可少的，例如论坛、电子邮箱、在线购物等。登录功能能够让网站准确的验证用户的身份。用户能够访问该网站时，可以注册并登录，登录后的用户还能够注销登录状态以保证用户资料的安全性。ASP.NET 就提供了一系列的登录控件方便登录功能的开发。

6.1.1　登录控件（Login）

登录控件是一个复合控件，它包含用户名和密码文本框，以及一个询问用户是否希望在下一次访问该页面时记起其身份的复选框。当用户勾选此选项时，下一次用户访问此网站后，将自动进行身份验证 。创建一个登录控件代码，系统会自动生成相应的 HTML 代码，示例代码如下所示。

```
<asp:Login ID="Login1" runat="server">
</asp:Login>
```

上述代码则创建了一个登录控件，如图 6-1 所示。开发人员可以通过属性的设置更改登录控件的样式等，如图 6-2 所示。

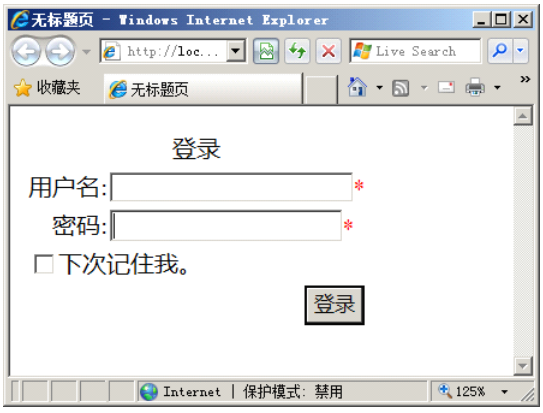

图 6-1　默认登录窗口

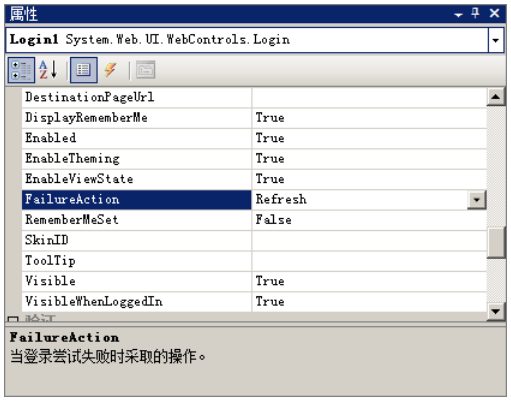

图 6-2　登录框属性的设置

开发人员能够使用登录控件执行用户登录操作而无需复杂的代码实现，登录控件常用的属性如下所示。

- Orientation：控件的一般布局。
- TextLayout：标签相对于文本框的布局。
- CreatUserIconUrl：用户创建用户连接的图标的 URL。

- CreatUserText：为“创建用户”连接显示的文本。
- CreatUserUrl：创建用户页的 URL。
- HelpPageIconUrl：用于帮助页连接的图标的 URL。
- HelpPageText：为帮助连接显示的文本。
- HelpPageUrl：帮助页的 URL。
- PasswordRecoveryIconUrl：用于密码回复连接的图标的 URL。
- PasswordRecoveryUrl：为密码回复连接显示的文本。
- PasswordRecoveryText：密码回复页的 URL。
- MembershipProvider：成员资格提供程序的名称。
- FailuteText：当登录尝试失败时显示的文本。
- InstructionText：为给出说明所显示的文本。
- LoginButtonImageUrl：为“登录”按钮显示的图像的 URL。
- LoginButtonText：为“登录”按钮显示的文本。
- LoginButtonType：“登录”按钮的类型。
- PasswordLableText：密码标识文本框内的文本。
- RememberMeText：为“记住我”复选框所显示的文本。
- TitleText：为标题显示的文本。
- UserName：用户名文本框内的初始值。
- UserNameLableText：标识用户名文本框的文本。
- DestinationPageUrl：用户成功登录时被定向到的 URL。
- DisplayRememberMe：是否显示“记住我”复选框。
- Enabled：控件是否处于启动状态。
- RememberMeSet：“记住我”复选框是否初始化被选中。
- VisibleWhenLoggedIn：是否控件在用户登录时保持可见。
- PasswordRequiredErrorMessage：密码为空时在验证摘要中显示的文本。
- UserNameRequiredErrorMessage：用户名为空时在验证摘要中显示的文本。

同样，登录控件还包括许多常用的事件，登录控件常用的事件如下所示：

- Authenticate：当用户使用登录控件登录到网站时，引发该事件。
- LoggedIn：对用户进行身份验证后引发该事件。
- LoggingIn：对用户进行身份验证前引发该事件。
- LoginError：对用户进行用户身份验证失败时引发该事件。

开发人员能够在页面中拖动相应的登录控件实现登录操作，使用登录控件进行登录操作可以直接进行用户的信息的查询而无需复杂的登录实现。

6.1.2 登录名称控件（LoginName）

登录名称控件（LoginName）是一个用来显示已经成功登录的用户的控件。在 Web 应用程序开发中，开发人员常常需要在页面中通知相应的用户已经登录，如用户在商品网站上进行登录，登录成功后可以在相应的页面中提示“您已登录，您的用户名是 XXX”等，这样不仅能够提高用户的友好度，也能够让开发人员在 Web 应用程序中方便的对用户信息做收集整理。

开发人员能够方便的在应用程序中拖动 LoginName 控件用于用户名的呈现，拖动到页面中，系统生

成的 HTML 代码如下所示。

```
<asp:LoginName ID="LoginName1" runat="server" />
```

上述代码则实现了一个登录名称控件，开发人员能够将该控件放置在页面中的任何位置进行页面呈现，当用户登录后，该控件能够获取用户的相应信息并呈现用户名在控件中。登录控件页面效果如图 6-3 所示。

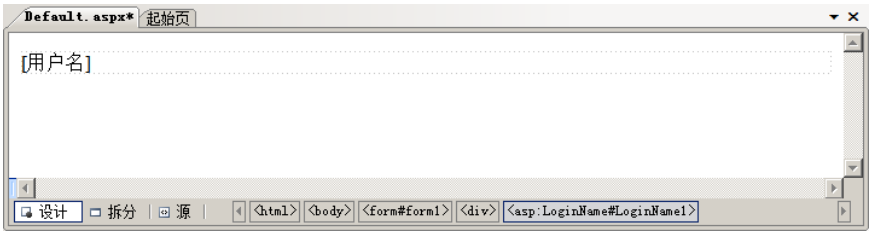

图 6-3　登录名称控件

注意：LoginName 控件只能够在<body>标记内的<form>标记中使用，该控件不能够使用于<title>、<style>等标记中。

在 LoginName 控件中，最常用的属性为 FormatString 属性，该属性用于格式化用户名输出。在控件的 FormatString 属性中，“{0}”字符串用于显式用户名，开发人员能够配置相应的字符串进行输出，例如配置成“您好，{0}，您已经登录！”，可以在相应的占位符中呈现相应的用户名，如图 6-4 所示。

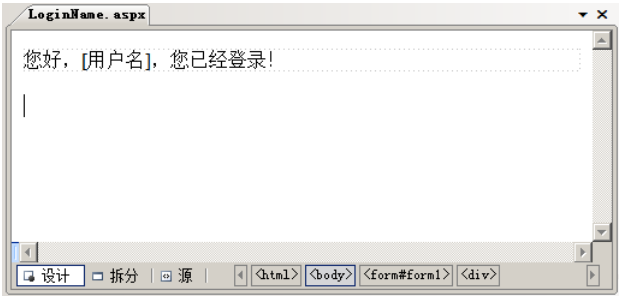

图 6-4　格式化输出用户名

正如图 6-4 所示，当对 LoginName 进行格式化规定后，用户名能够被格式化输出，例如当用户 soundbbg 登录在 Web 应用后，该控件会呈现“您好，soundbbg，您已登录！”。开发人员只需要通过简单的配置就能够实现复杂的登录显示功能操作的实现。

6.1.3　登录视图控件（LoginView）

在应用程序的开发过程中，通常需要对不同的身份和权限的用户进行不同登录样式的呈现，开发人员可以为用户配置内置对象以呈现不同的页面效果。但是在页面请求时，还需要对用户的身份进行验证。在 ASP.NET 2.0 之后的版本中，系统提供了 LoginView 控件用于不同用户权限之间的视图的区分。

在开发一个应用程序时，开发人员希望应用程序能够实现功能当用户在网站中没有登录时，用户看到的视图是没有登录时的视图，包括网站的风格、系统的提示信息等。而当用户登录后，用户看到的视图是登录后的视图，同样包括网站的风格、系统的提示信息等。LoginView 控件为开发人员提供了不同权限的用户进行不同视图的查看的功能，开发人员能够拖动 LoginView 控件在页面中以编辑不同的页面进行开发。

拖动一个 LoginView 控件在页面中，开发人员能够通过编辑不同的模板进行不同权限的页面的编写，

拖动 LoginView 控件后系统生成的 HTML 代码如下所示。

```
<asp:LoginView ID="LoginView1" runat="server">
</asp:LoginView>
```

上述代码为默认的 LoginView 控件的代码，开发人员需要通过编写相应的模板以便不同的用户查看不同的页面，在 LoginView 控件中，包括两个最常用的模板，这两个模板及其作用分别如下所示。

- AnonymousTemplate：匿名模板，当用户没有进行登录时，该模板会呈现在匿名用户面前。
- LoggedInTemplate：已登录模板，当用户已经登录成功后，该模板会呈现在已经登录的用户面前。

开发人员可以通过编写相应的模板进行页面的呈现，当用户没有登录时，用户可以看见 AnonymousTemplate 模板中的内容而无法看见 LoggedInTemplate 模板的内容；而如果用户已经登录，则登录过后的用户能够看见 LoggedInTemplate 模板的内容而无法看见 AnonymousTemplate 模板的内容，如图 6-5 所示。

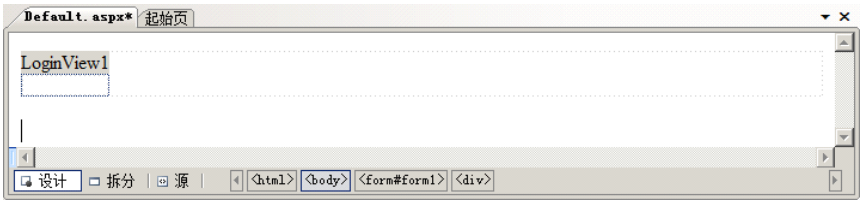

图 6-5　LoginView 控件

在 AnonymousTemplate 模板中，该模板通过获取和判断 PageUser 属性的 Name 属性进行判断。如果 PageUser 属性的 Name 属性为空时，AnonymousTemplate 模板则不会向通过身份验证的用户的呈现相应的页面。开发人员可以通过编写 AnonymousTemplate 模板和 LoggedInTemplate 模板进行不同用户的样式呈现，示例代码如下所示。

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:LoginView ID="LoginView1" runat="server">
            <LoggedInTemplate>
                这是一个登录用户可以访问的页面..
            </LoggedInTemplate>
            <AnonymousTemplate>
                这是一个匿名用户可以访问的页面..
            </AnonymousTemplate>
        </asp:LoginView>
    </div>
    </form>
</body>
```

上述代码为不同权限的用户配置了不同的模板，当不同权限的用户访问页面时，其看到的页面样式也是不同的。在 LoginView 控件中，还能够为不同权限和身份的用户配置不同的模板，开发人员能够为不同的用户分配不同的角色。当用户被分配了不同的角色后，用户能够通过相应的角色访问相应的模板，例如普通用户可以访问普通用户模板，VIP 用户可以访问 VIP 模板而管理员可以访问管理员模板。

在 LoginView 控件中，单击 RoleGroup 集合，可以添加相应的 LoginView 控件的 RoleGroup 集合，如图 6-6 所示。

图 6-6　添加 RoleGroup 集合

这里添加了两个 RoleGroup 集合，该 RoleGroup 集合分别包含 admin 和 VIP 两种用户类别，当用户为 admin 或 VIP 是，可以通过相应的权限绑定进行不同模板的访问，创建后示例代码如下所示。

```
<asp:LoginView ID="LoginView1" runat="server">
    <RoleGroups>
        <asp:RoleGroup Roles="admin">
            <ContentTemplate>
                这是一个管理员用户可以访问的页面..
            </ContentTemplate>
        </asp:RoleGroup>
        <asp:RoleGroup Roles="vip">
            <ContentTemplate>
                这是一个 VIP 用户可以访问的页面
            </ContentTemplate>
        </asp:RoleGroup>
    </RoleGroups>
    <LoggedInTemplate>
        这是一个登录用户可以访问的页面..
    </LoggedInTemplate>
    <AnonymousTemplate>
        这是一个匿名用户可以访问的页面..
    </AnonymousTemplate>
</asp:LoginView>
```

当有不同身份的用户访问该控件时，控件能够通过用户的身份进行不同模板的呈现，这样就方便了开发人员对不同身份和权限的用户进行网站应用程序和模板的访问限制了。

注意：当一个用户拥有的身份或权限不在列表的权限中时，该用户会默认访问 LoggedInTemplate 模板，并且无论是 LoggedInTemplate 模板还是 RoleGroup 模板，都不会对匿名用户呈现。

6.1.4　登录状态控件（LoginStatus）

登录状态控件（LoginStatus）用于显式用户验证时的状态，LoginStatus 包括“登录”和“注销”两种状态，对于 LoginStatus 控件的状态是由相应的 Page 对象的 Request 属性中的 IsAuthenticated 属性进行决定。开发人员能够直接将 LoginStatus 控件拖放在页面中，从而让用户能够通过相应的状态进行登录或注销操作，LoginStatus 控件默认 HTML 代码如下所示。

```
<asp:LoginStatus ID="LoginStatus1" runat="server" />
```

上述代码就呈现了一个 LoginStatus 控件，LoginStatus 控件默认的呈现形式是以文本的形式呈现的，如图 6-7 所示。

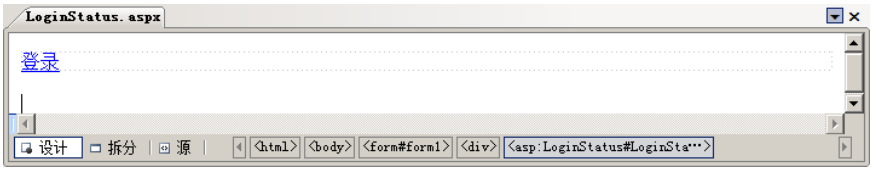

图 6-7　LoginStatus 控件呈现形式

正如图 6-7 所示，LoginStatus 控件默认的呈现形式是以文本的形式呈现的。当用户没有在网站上进行登录操作时，该控件会呈现登录字样给用户以便用户进行登录操作，当用户登录后，LoginStatus 控件会为用户提供注销字样以便用户进行注销操作。开发人员还能够为 LoginStatus 控件指定以图片形式进行登录和注销，LoginStatus 控件常用的属性如下所示。

- LoginImageUrl：设置或获取用于登录连接的图像 URL。
- LoginText：设置或获取用于登录连接的文本。
- LogoutAction：设置或获取一个值用于用户从网站注销时执行的操作。
- LogoutImageUrl：设置或获取一个值用于登出图片的显示。
- LogoutPageUrl：设置或获取一个值用于登出连接的图像 URL。
- LougoutText：设置或一个值用于登出连接的文本。
- TagKey：获取 LoginStatus 控件的 HtmlTextWriterTag 的值。

开发人员可以配置 LoginImageUrl 以及 LogoutImageUrl 属性进行登录、登出的图片显示，使用图片进行登录登出操作能够提高用户体验，示例代码如下所示。

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:LoginStatus ID="LoginStatus1" runat="server" LoginImageUrl="~/login.jpg"
            LogoutImageUrl="~/logout.jpg" />
    </div>
    </form>
</body>
```

上述代码指定了当用户没有登录时，相应的登录操作以图片的形式呈现在页面中，同样当用户登录后，注销操作也会以图片的形式呈现在页面中，如图 6-8 所示。

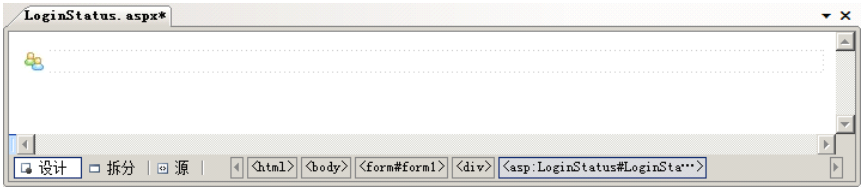

图 6-8　图片形式呈现

LoginStatus 控件还包括两个常用事件，这两个事件分别为 LoggingOut 和 LoggedOut。当用户单击注销按钮时会触发 LoggingOut 事件，开发人员能够在 LoggingOut 事件中编写相应的事件以清除用户的身份信息，这些信息包括 Session、Cookie 等。开发人员还能够在 LoggedOut 事件中规定在用户离开网站时所必须执行的操作。

6.1.5 密码恢复控件（PasswordRecovery）

当用户进行 Web 应用程序访问时，在有些情况下会丢失用户密码，这样就需要通过 Web 应用程序恢复自己的密码。在应用程序开发中，为了提高系统的安全性和用户信息的私密性，开发人员常常需要编写诸多代码来保存用户的信息并进行用户请求的检测。ASP.NET 中提供了密码恢复控件以便开发人员能够在 Web 应用中轻松的能够让用户自行进行密码回复。

开发人员能够拖动一个 PasswordRecovery 控件在页面中，系统能够在主窗口中创建一个 PasswordRecovery 控件所必须的声明，示例代码如下所示。

```
<asp:PasswordRecovery ID="PasswordRecovery1" runat="server">
</asp:PasswordRecovery>
```

开发人员能够使用 PasswordRecovery 控件进行相应的配置，包括自动太用格式、视图配置、转换成模板以及网站管理等，如图 6-9 所示。

对于 PasswordRecovery 控件而言，开发人员能够单击 PasswordRecovery 控件的属性进行相应的配置，例如选择自动套用格式，单击【自动套用格式】按钮进行格式的选取，如图 6-10 所示。

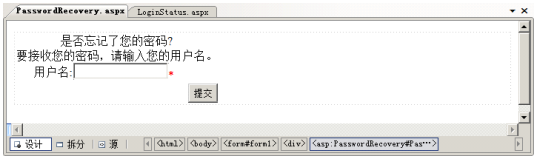

图 6-9　默认的 PasswordRecovery 控件

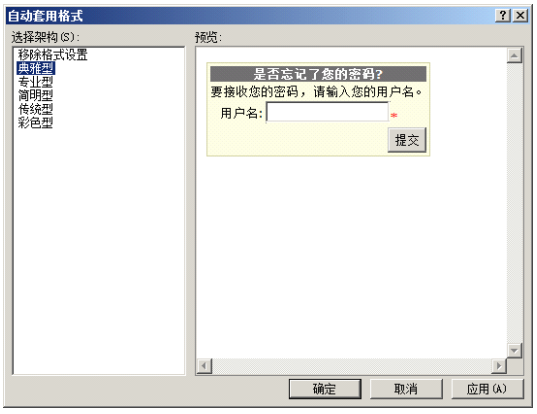

图 6-10　选择默认格式

开发人员可以选择自动套用格式进行模板的编写，以提高用户体验，开发人员还能够自行编写模板进行 PasswordRecovery 控件的样式控制，选择相应的样式后，系统会自行生成样式控制代码，示例代码如下所示。

```
<asp:PasswordRecovery ID="PasswordRecovery1" runat="server" BackColor="#F7F7DE"
    BorderColor="#CCCC99" BorderStyle="Solid" BorderWidth="1px"
    Font-Names="Verdana" Font-Size="10pt">
    <TitleTextStyle BackColor="#6B696B" Font-Bold="True" ForeColor="#FFFFFF" />
</asp:PasswordRecovery>
```

开发人员能够通过修改上面的颜色进行样式控制。在 PasswordRecovery 控件中，除了能够自动套用和开发 PasswordRecovery 控件的格式外，开发人员还能够为 PasswordRecovery 控件相应的功能进行样式控制。PasswordRecovery 控件包括三个基本功能，分别为【用户名】、【密码提示问题】和【成功模板】。

在用户使用 PasswordRecovery 控件进行密码恢复时，首先需要输入用户名进行用户名的匹配。如果用户名匹配后 PasswordRecovery 控件要求用户进行问题答案的填写。如果答案正确，PasswordRecovery 控件能够为用户显示【成功模板】。

开发人员能还够分别为三个功能进行模板创建。在默认情况下，开发人员不能够进行模板的编辑，开发人员可以选择 PasswordRecovery 控件中【管理】菜单中的【转换为模板】选项进行相应的模板转换，

如图 6-11 所示。

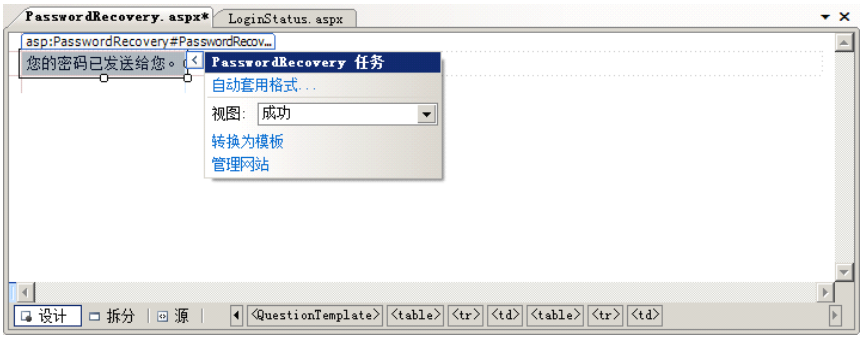

图 6-11　转换为模板

当转换为模板之后，开发人员就能够在模板中编写相应的文档或样式控制提高用户体验的友好度。在编写相应的模板后，该控件中的三个功能会分别被生成为模板形式而存在，示例代码如下所示。

```
<QuestionTemplate>
    <table border="0" cellpadding="1" cellspacing="0" style="border-collapse: collapse;">
        <tr>
        <td>
            <table border="0" cellpadding="0">
        <tr>
            <td  align="center"  colspan="2"  style="color:  White;  background-color:  #6B696B;  font-weight:
bold;">
            标识确认</td>
        </tr>
        <tr>
            <td align="center" colspan="2">要接收您的密码，请回答下列问题。只有当填写了相应的问题后，
            您的用户密码才能够被恢复</td>
        </tr>
        <tr>
            <td align="right">用户名:</td>
            <td>
            <asp:Literal ID="UserName" runat="server"></asp:Literal>
            </td>
        </tr>
        <tr>
            <td align="right">问题:</td>
            <td>
            <asp:Literal ID="Question" runat="server"></asp:Literal>
            </td>
        </tr>
        <tr>
            <td align="right">
            <asp:Label ID="AnswerLabel" runat="server" AssociatedControlID="Answer">答案:</asp:Label>
            </td>
            <td>
            <asp:TextBox ID="Answer" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator ID="AnswerRequired" runat="server"
            ControlToValidate="Answer" ErrorMessage="需要答案。"
```

```
            ToolTip="需要答案。" ValidationGroup="PasswordRecovery1">*</asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td align="center" colspan="2" style="color: Red;">
            <asp:Literal ID="FailureText" runat="server" EnableViewState="False"></asp:Literal>
            </td>
        </tr>
        <tr>
            <td align="right" colspan="2">
            <asp:Button id="SubmitButton" runat="server" commandname="Submit"
            text="提交" validationgroup="PasswordRecovery1" />
            </td>
        </tr>
            </table>
            </td>
        </tr>
    </table>
</QuestionTemplate>
```

上述代码实现了【提问模板】中的模板信息和样式，当用户进入提问功能时会呈现该模板。当用户输入用户名时，系统会查找相应的用户信息并跳转到提问页面。如果用户回答自己提问的问题并回答正确后，PasswordRecovery 控件会将密码发送到相应的邮箱中，而如果用户回答出错，PasswordRecovery 控件就保留密码，以提高系统的安全性。

6.1.6 密码更改控件（ChangePassword）

在应用程序开发中，开发人员需要编写密码更改控件让用户能够快速的进行密码更改。在应用程序的使用中，用户会经常需要更改密码，更改密码有很多的可能性。例如用户进行登录后发现自己的用户信息可能被其他人改动过，就有可能怀疑密码泄露的问题，这样用户就可以通过更改密码进行密码的更换。另外，如果用户在注册时的密码是系统自动生成的密码，用户同样需要在密码更改控件中修改生成的密码以便用户记忆。

在 ASP.NET 中提供了密码更改控件以便开发人员能够轻易的完成密码更改功能。拖放一个密码更改控件在页面，系统会自动生成相应的 HTML 代码，示例代码如下所示。

```
        <asp:ChangePassword ID="ChangePassword1" runat="server">
        </asp:ChangePassword>
```

ChangePassword 控件包括密码、新密码和确认新密码，如图 6-12 所示。

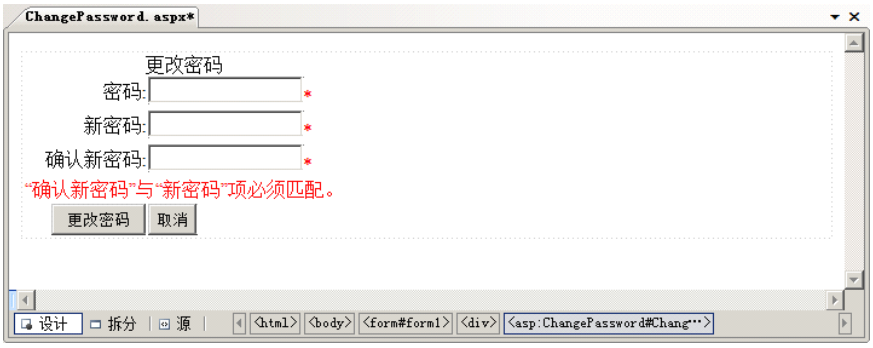

图 6-12　ChangePassword 控件

当用户需要更改密码时，用户必须先填写旧密码进行密码的验证，如果用户填写的旧密码是正确的密码，则系统会将新密码替换旧密码以便用户下次登录时使用新密码。如果用户填写的旧密码不正确，则系统会认为可能是一个非法用户而不允许更改密码。ChangePassword 控件同样允许开发人员自动套用格式或者通过编写模板进行 ChangePassword 控件的样式布局，如图 6-13 所示。

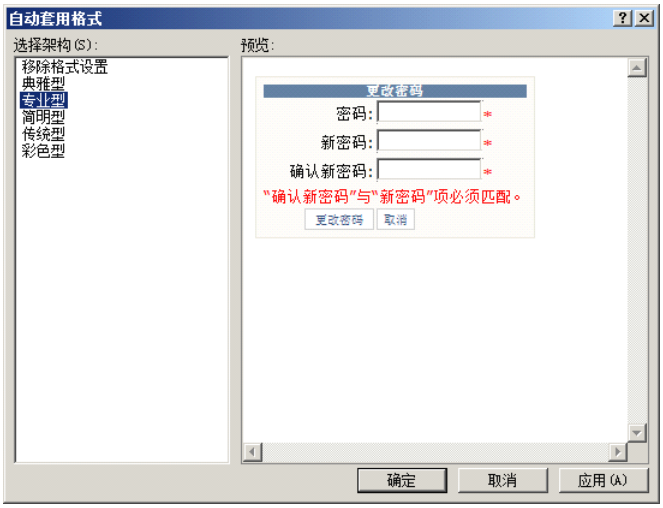

图 6-13　自动套用格式

开发人员能够自动套用格式进行更改密码控件的呈现，不仅如此，开发人员还能够单击右侧的功能导航进行模板的转换，转换成模板后开发人员就能够进行模板的自定义。ChangePassword 控件可以使用 Web.config 中的 membership 配置节进行成员资格配置，所以 ChangePassword 控件能够实现不同场景的不同功能，这些功能如下所示。

- 用户登录情况：开发人员能够使用 ChangePassword 控件允许用户在不登录的情况下进行密码的更改。
- 更改用户密码：开发人员能够使用 ChangePassword 控件让一个登录的用户进行另一个用户的密码的更改。

在 ChangePassword 控件中，开发人员可以通过配置 ChangePassword 控件的相应属性进行 ChangePassword 控件的样式或者是功能的设置，这样能够保证在一定的安全范围内进行安全的用户信息操作。ChangePassword 控件常用的属性如下所示。

- CancelButtonImageUrl：配置取消按钮控件的图片文本，该属性可以为按钮控件指定一个图片按钮进行呈现。
- CancelButtonStyle：配置取消按钮控件的样式和外观的属性集。
- ChangePasswordButtonType：配置更改密码控件的类型。
- ChangePasswordFailtureText：配置更改密码失败时所呈现的错误信息。
- ConfirmNewPassword：获取用户输入的重复密码的值。
- ConfirmPasswordCompareErrorMessage：当用户输入密码和输入验证密码出现错误时提示的错误消息。
- ConfimPasswordRequiredErrorMessage：当用户没有输入“确认新密码”时在控件中提示的错误消息。
- ContinueButtonImageUrl：为继续按钮配置一个图片文本，该属性可以为按钮控件指定一个图片

按钮进行呈现。

- ContinueButtonStyle：为继续按钮配置样式或属性集。

开发人员能够配置相应的ChangePassword控件的属性进行不同的ChangePassword控件的样式呈现，以及功能实现。在 ChangePassword 控件中，有许多属性都是包括按钮或表格的样式的呈现的属性，这里就不再一一列举。

6.1.7 生成用户控件（CreateUserWizard）

生成用户控件（CreateUserWizard）为 MembershipProvider 对象提供了用户界面，使用该控件能够方便的让开发人员在页面中生成相应的用户，同时当用户访问该应用程序时，用户能够通过使用 CreateUserWizard 控件的相应的功能进行注册，如图 6-14 所示。

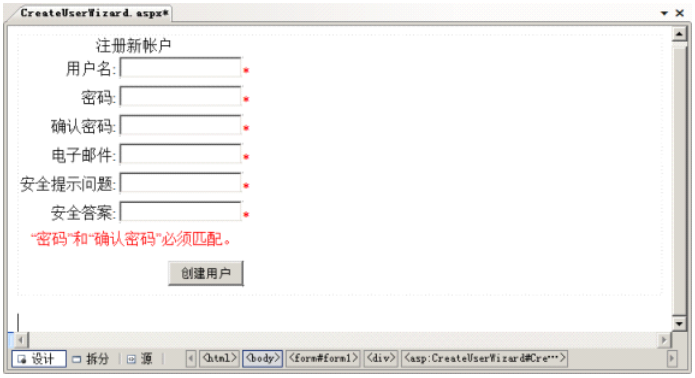

图 6-14 CreateUserWizard 控件

正如图 6-14 所示，CreateUserWizard 控件默认包括多个文本框控件以便用户的输入，这里包括用户名、密码、确认密码、电子邮件、安全提示问题和问题答案等项目。其中用户名、密码、确认密码用于身份验证和数据插入为系统提供用户信息，而电子邮件和安全答案用于当用户忘记密码或更改密码时向用户发送相应的邮件以便提高系统身份认证的安全性。

开发人员能够将 CreateUserWizard 控件拖放在主窗口中进行页面呈现，这样就能够实现用户注册功能，当开发人员拖动 CreateUserWizard 在主窗口中是，系统会自动生成 HTML 代码，示例代码如下所示。

```
<asp:CreateUserWizard ID="CreateUserWizard1" runat="server">
    <WizardSteps>
        <asp:CreateUserWizardStep runat="server" />
        <asp:CompleteWizardStep runat="server" />
    </WizardSteps>
</asp:CreateUserWizard>
```

上述代码创建了一个 CreateUserWizard 控件进行用户注册功能的实现，开发人员还能够为 CreateUserWizard 控件中相应的模板进行样式控制。例如当用户注册完毕后，用户会跳转到一个页面提示“账户注册完毕，请登录”等等，这样就能提高用户体验。单击【自定义完成步骤】按钮或在快捷窗口下拉菜单中选择【完成】选项就能够进行完成模板的实现，如图 6-15 所示。

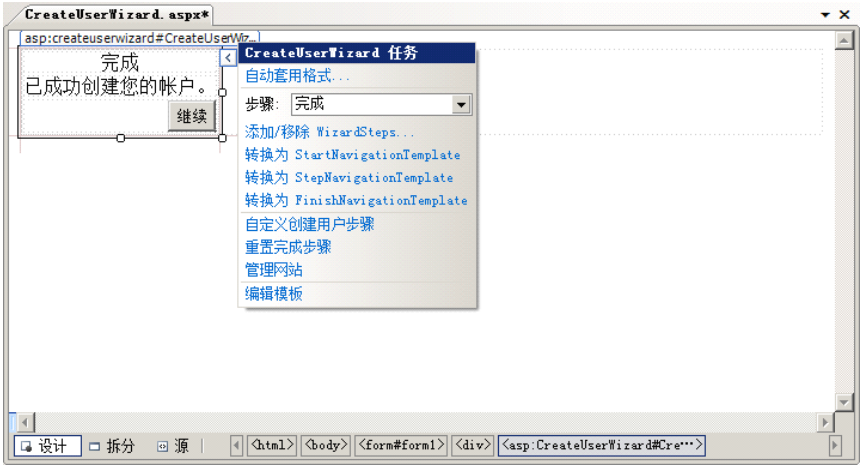

图 6-15　完成模板的编写

开发人员能够在完成步骤中编辑模板以便能够进行更多的提示和更好的用户体验，编辑完成模板后，系统会自动更改相应的代码，示例代码如下所示。

```
<asp:CreateUserWizard ID="CreateUserWizard1" runat="server" ActiveStepIndex="1">
    <WizardSteps>
        <asp:CreateUserWizardStep runat="server" />
        <asp:CompleteWizardStep runat="server">
        <ContentTemplate>
            <table border="0">
                <tr>
                    <td align="center">恭喜您！注册完毕！</td>
                </tr>
                <tr>
                    <td>已成功创建您的账户，请登录。</td>
                </tr>
                <tr>
                    <td align="right">
                    <asp:Button ID="ContinueButton" runat="server" CausesValidation="False"
                    CommandName="Continue" Text="继续" ValidationGroup="CreateUserWizard1" />
                    </td>
                </tr>
            </table>
        </ContentTemplate>
        </asp:CompleteWizardStep>
    </WizardSteps>
</asp:CreateUserWizard>
```

上述代码创建了一个完成注册的模板，开发人员还可以通过编写自定义创建用户模板以便更加方便的创建用户。CreateUserWizard 控件还包括其他模板，这些模板能够方便开发人员进行更高的页面呈现，这些模板及其说明如下所示。

- HeadTemplate：获取或设置标题区的模板内容。
- SideBarTemplate：获取或设置侧边栏的模板内容。
- StartNavigationTemplate：获取或设置起始步骤中导航区域的模板内容。
- StepNavigationTemplate：获取或设置不同步骤中导航区域的模板内容。

- FinishNavigationTemplate：获取或设置结束步骤中导航区域的模板内容。
- ContentTemplate：获取或设置在创建用户模板和完成模板中的模板内容。

开发人员还能够通过 HeadTemplate、SideBarTemplate 等模板进行高级的 CreateUserWizard 控件的页面呈现和样式控制，这样不仅能够提高用户体验和友好度，还能够清晰的让用户按照步骤执行操作，降低了错误的出现率。

6.2 网站管理工具

在使用高级用户控件时，开发人员需要使用网站管理工具进行相应的控件配置和网站管理，网站管理工具包括安全、应用程序配置和提供应用程序配置。开发人员能够在管理工具中进行用户访问的权限，以及应用程序配置等高级网站管理。

6.2.1 启动管理工具

在 ASP.NET 应用程序开发中，通常都是通过手动进行 Web.config 配置文件的更改。在 ASP.NET 应用程序中，系统提供了网站管理工具用于系统的用户、用户权限以及系统的配置的管理，开发人员能够很容易的进行 ASP.NET 应用程序的管理。

在应用程序中，特使需要使用到用户及网站管理的用户控件中，在侧边的快捷操作栏中都会包括一个【网站管理】选项，单击【网站管理】选项能够启动网站管理工具以便能够进行相应的网站管理，如果没有使用相应的控件进行 ASP.NET 网站管理，开发人员可以在导航菜单栏中右击当前项目，在下拉菜单中选择【ASP.NET 配置】选项进行网站管理，如图 6-16 所示。

当开发人员选择【ASP.NET 配置】选项后，Visual Studio 2008 会创建一个虚拟服务器用于管理工具的执行。在管理工具中，开发人员能够配置安全、应用程序配置和提供应用程序配置进行高级的 ASP.NET 应用程序配置。如图 6-17 所示。

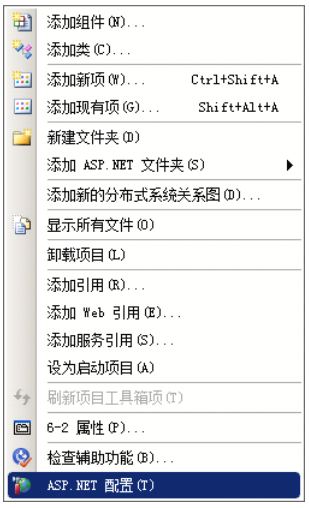

图 6-16 选择 ASP.NET 配置

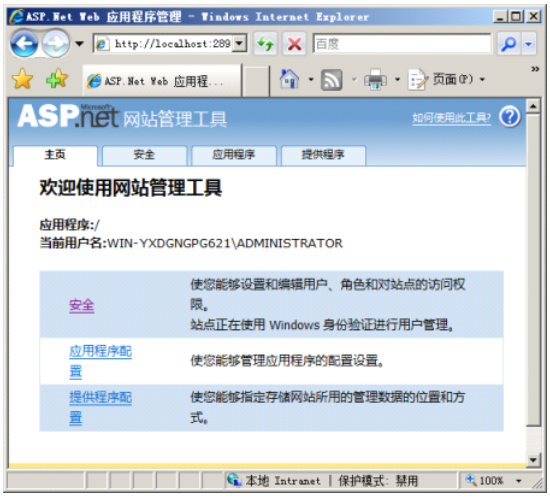

图 6-17 启动网站管理工具

注意：在使用 ASP.NET 管理工具进行网站管理时，推荐关闭 Web.config 文件或停止使用该文件，因为管理工具可能会在配置和运行中更改该文件。

6.2.2 用户管理

开发人员能够单击【安全】标签进行相应的应用程序安全的管理。安全管理包括用户管理、用户角色管理以及用户的访问规则管理，如图 6-18 所示。

使用网站管理工具能够进行用户管理，用户管理功能仅对表单验证有效。如果当前的验证方式是基于 Windows 默认的身份验证时，则会在用户栏中显式【当前身份验证类型为 Windows，因此禁用了此工具中的用户管理】文本，开发人员能够选择身份验证类型进行身份验证类型的配置，如图 6-19 所示。

图 6-18 用户管理

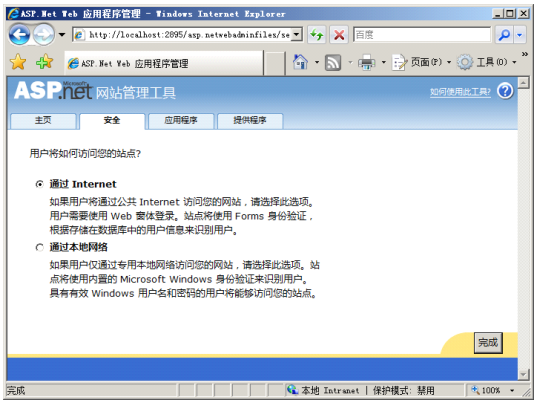

图 6-19 身份验证类型配置

在身份验证类型配置中，允许开发人员进行用户访问的配置，这里包括两个用户访问配置，这两个用户访问配置如下所示。

- 通过 Internet：如果用户将通过公共 Internet 访问该网站时网站，可以选择此选项作为配置。用户需要使用 Web 窗体登录并且站点将使用 Forms 进行身份验证，从而根据存储在数据库中的用户信息来识别用户。
- 通过本地网络：如果用户仅通过专用本地网络访问该的网站，可以选择此选项，站点将使用内置的 Microsoft Windows 身份验证来识别用户。

开发人员能够根据应用程序的功能的不同进行不同用户访问的配置，通常情况下可以选择【通过 Internet】选项进行用户的访问配置，单击【完成】按钮后，系统会呈现相应的用户管理信息，如图 6-20 所示。

图 6-20 用户管理

当配置【用户访问】选项为【通过 Internet】选项时，在用户管理中会生成相应的统计和功能，开发人员能够在用户选项卡中选择创建用户或管理用户，并在相应的选项卡中显示用户的统计。

6.2.3 用户角色

在 ASP.NET 应用程序开发中，需要对不同的用户进行用户角色的管理，例如该用户可能是一个学

生，也可能是一个学生甚至是一个管理员。使用 ASP.NET 管理工具能够快速的创建用户角色以便管理不同角色的用户。在角色选项卡中选择【启动角色】选项即可启动角色，启动角色后，开发人员就能够在角色中创建和管理角色，单击【创建或管理角色】按钮进行角色管理，如图 6-21 所示。

图 6-21　创建新角色

单击【添加角色】按钮就能够在 ASP.NET 应用程序中创建相应的角色，创建的角色可以在用户注册和用户登录时进行选择和管理，如图 6-22 所示。创建完成后，开发人员能够选择相应的用户角色进行用户角色的管理，如图 6-23 所示。

图 6-22　创建角色

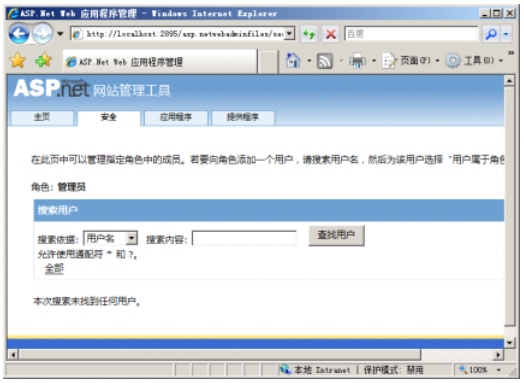

图 6-23　管理角色

开发人员能够为相应角色的用户进行信息的修改和删除。在进行用户管理前，开发人员还能够进行用户的搜索。ASP.NET 网站管理工具支持开发人员使用通配符进行用户搜索和筛选，这样就能够提高用户筛选的效率以便在大量用户前提下进行相应的用户角色的管理。

6.2.4　访问规则管理

在 ASP.NET 管理工具中，开发人员能够为用户进行访问规则的管理，这在应用程序开发中是非常

必要的。在 ASP.NET 应用程序的开发中，开发人员通常是不允许用户进入到后台管理页面中的，这是非常重要的，一个普通用户无法进入到相应的后台中进行管理。而对于管理员而言，管理员能够进入到后台进行相应的管理。

在应用程序开发中，将管理员和用户分开开发是非常不明智的选择，同样这样开发也会造成应用程序维护困难。在 ASP.NET 管理工具中，可以为相应的用户角色配置相应的访问权限。选择【访问规则】选项卡，开发人员能够创建访问规则和管理访问规则，单击【创建访问规则】按钮可以进行访问规则的创建，如图 6-24 所示。

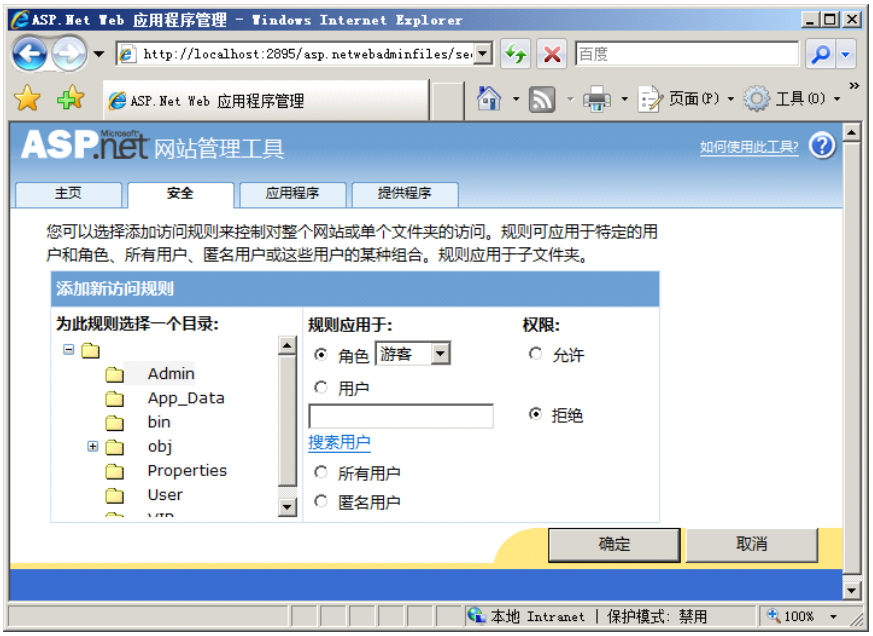

图 6-24　创建访问规则

开发人员能够在访问规则管理器中选择相应的目录进行访问规则的创建，正如图 6-24 所示。首先在左侧选择了 Admin 文件夹，由于该文件是一个机密文件夹，所以游客用户是不能够进行访问的。开发人员能够右侧的规则管理中的下拉菜单中选择【游客】选项并在【权限】选项中选择【拒绝】选项以禁止【游客】权限的用户的访问。

开发人员还能够为其他目录如 VIP、User 等目录进行访问权限的添加，如图 6-25 所示。在添加完成访问规则后，开发人员能够在选项卡中选择【管理访问规则】选项进行访问规则的管理，开发人员可以通过在左侧选择相应的文件夹目录进行访问规则的管理，如图 6-26 所示。

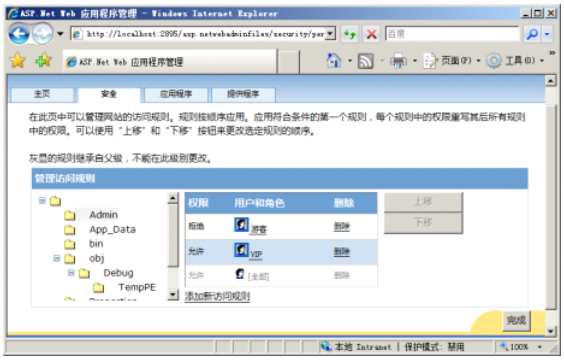

图 6-25　管理访问规则

图 6-26　选择访问规则

当开发人员选择不同的文件夹时，其访问规则呈现的也不同，开发人员能够在访问规则管理面板中删除相应的规则以修改角色的访问权限。

2.6.5 应用程序配置

在 Web.config 文件中，开发人员可以手动的进行应用程序管理和配置，在 ASP.NET 管理工具中，开发人员可以使用管理工具进行应用程序配置，如图 6-27 所示。

图 6-27　应用程序配置

使用应用程序配置能够创建应用程序设置和管理应用程序设置，创建的应用程序设置会保存在 Web.config 文件的 appSettings 配置节中，示例代码如下所示。

```
<appSettings>
    <add key="sql" value="0" />
</appSettings>
```

appSettings 配置节中的信息能够在应用程序中通过编程进行获取，这样就提高了应用程序的灵活性。除了能够配置 appSettings 配置节中的应用程序设置，开发人员还能够通过应用程序管理面板进行 SMTP 邮件配置，如图 6-28 和图 6-29 所示。

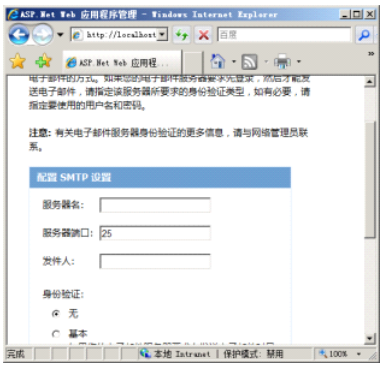

图 6-28　配置端口

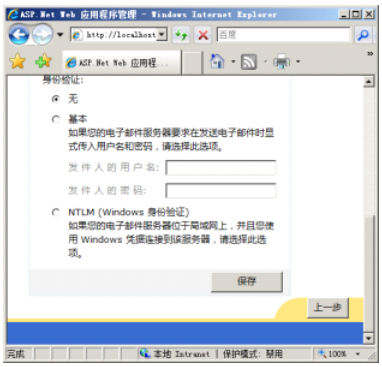

图 6-29　配置邮件

配置邮件后，登录等高级控件就能够通过该配置进行邮件的发送，当用户进行密码更改和密码索取时，相应的控件能够通过邮件配置进行密码和信息的发送。在 ASP.NET 应用程序配置中，还能够配置应用程序状态、配置调试和跟踪、定义默认错误页等功能而无需手动修改 Web.config，极大的方便了开发人员在 ASP.NET 应用程序开发中的应用程序配置，以及系统调配。

6.3 使用登录控件

使用登录控件前，需要进行相应的应用程序配置进行登录控件的使用，因为登录控件等高级控件的使用都是基于 ASP.NET 应用程序配置而存在的，这些控件不能够独立的运行。在实现相应的操作时，这些控件还需要使用默认的方法和配置信息进行方法操作，登录控件的使用非常简单，这里挑选两个重要的控件进行讲解。

6.3.1 生成用户控件（CreateUserWizard）

在用户访问网站时，需要通过注册才能够进行用户信息的保存和获取。在用户注册时，可以使用生成用户控件（CreateUserWizard）进行用户注册功能的实现，CreateUserWizard 控件 HTML 代码如下所示。

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:CreateUserWizard ID="CreateUserWizard1" runat="server" BackColor="#F7F6F3"
        BorderColor="#E6E2D8" BorderStyle="Solid" BorderWidth="1px"
        Font-Names="Verdana" Font-Size="0.8em">
        <SideBarStyle BackColor="#5D7B9D" BorderWidth="0px" Font-Size="0.9em"
        VerticalAlign="Top" />
        <SideBarButtonStyle BorderWidth="0px" Font-Names="Verdana" ForeColor="White" />
        <ContinueButtonStyle BackColor="#FFFBFF" BorderColor="#CCCCCC"
        BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana"
        ForeColor="#284775" />
        <NavigationButtonStyle BackColor="#FFFBFF" BorderColor="#CCCCCC"
        BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana"
        ForeColor="#284775" />
        <HeaderStyle BackColor="#5D7B9D" BorderStyle="Solid" Font-Bold="True"
        Font-Size="0.9em" ForeColor="White" HorizontalAlign="Center" />
        <CreateUserButtonStyle BackColor="#FFFBFF" BorderColor="#CCCCCC"
        BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana"
        ForeColor="#284775" />
        <TitleTextStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
        <StepStyle BorderWidth="0px" />
        <WizardSteps>
        <asp:CreateUserWizardStep runat="server" />
        <asp:CompleteWizardStep runat="server">
        <ContentTemplate>
            <table border="0">
                <tr>
```

```
                    <td align="center" colspan="2">恭喜您！注册完毕！</td>
                </tr>
                <tr>
                    <td>已成功创建您的账户，请登录。</td>
                </tr>
                <tr>
                    <td align="right" colspan="2">
                    <asp:Button ID="ContinueButton" runat="server" CausesValidation="False"
                    CommandName="Continue" Text="继续" ValidationGroup="CreateUserWizard1" />
                    </td>
                </tr>
            </table>
        </ContentTemplate>
        </asp:CompleteWizardStep>
        </WizardSteps>
        </asp:CreateUserWizard>
    </div>
    </form>
</body>
```

上述代码在页面中呈现了 CreateUserWizard 控件并在页面中进行样式控制。当用户进行注册时，用户可以单击该控件并进行注册操作，运行后如图 6-30 后台 6-31 所示。

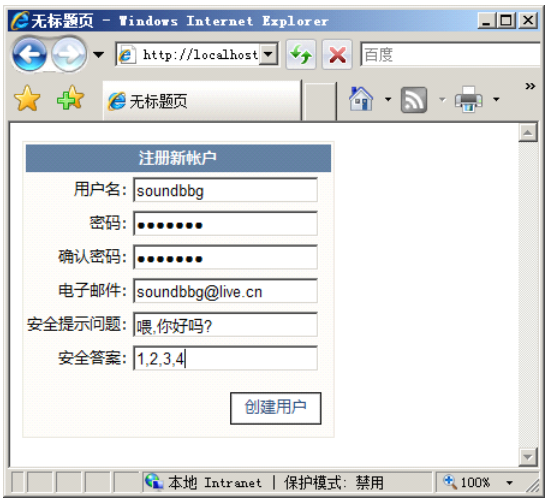

图 6-30　创建用户

图 6-31　创建成功

注意：在创建用户时，可能会遇到“密码最短长度为 7,其中必须包含以下非字母数字字符 1”的错误，如果出现这个错误说明密码强度不够，密码中必须包含~!@#$%^&*()_+字符串的一个，如果希望用户能够输入弱密码，可以修改 minRequiredNonalphanumericCharacters 的值为 0 即可。

当开发人员再次进入 ASP.NET 网站管理工具中时，开发人员能够在网站管理工具发现这两个用户已经被统计了并且可以为相应的用户进行管理操作，如图 6-32 所示。

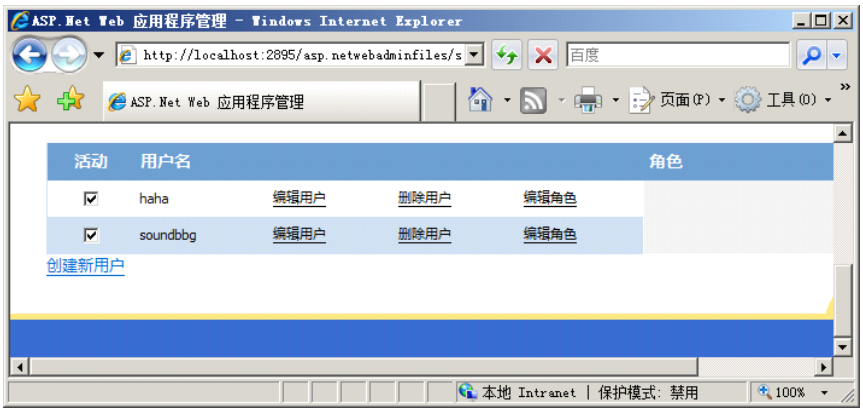

图 6-32　管理工具对用户的管理

在管理工具中，管理员可以对用户进行编辑、删除和角色管理，以便对注册的用户进行更加深入权限划分和信息编辑。

6.3.2　密码更改控件（ChangePassword）

当用户忘记密码后，可以通过使用密码控件进行密码的获取。在使用密码控件进行密码获取时，首先需要输入用户名进行用户身份验证，如图 6-33 所示。验证完成后系统会将相应的用户名匹配的问题呈现在用户界面中，用户需要填写相应的问题进行密码的获取，如图 6-34 所示。

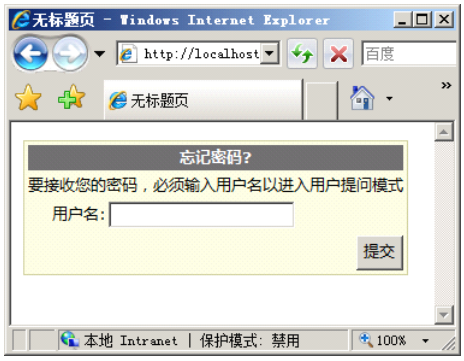

图 6-33　输入用户名

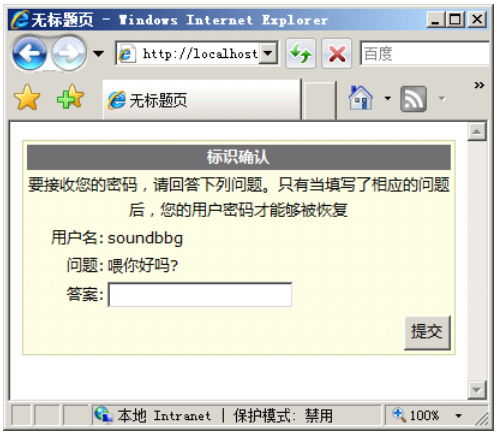

图 6-34　标识确认

用户必须在标识确认前进行用户名的输入，输入用户名后才能够跳转到标识确认模板进行问题的回答，当用户回答正确后，系统将会发送一份邮件到用户的邮箱中提示用户已经找回了相应的密码。

注意：在更改密码控件中，必须在 ASP.NET 管理工具中配置 SMTP 邮件发送的相应项用于邮件的发送，否则系统不会发送邮件到用户页面。

6.4　小结

本章讲解了 ASP.NET 中的高级控件用于 ASP.NET 应用程序的开发，虽然 ASP.NET 高级控件能够极大的简化开发人员的应用程序开发并通过 ASP.NET 管理工具进行高级控件的配置便开发人员对复杂

的应用的开发，但是 ASP.NET 高级控件同样包括一定的局限性，就是不够自主化。在后面的实例章节中会讲解如何通过手动创建一个登录、注册模块，以及如何在项目中使用模块。本章还包括：

- 启动管理工具：讲解了如何快速的启动管理工具。
- 访问规则管理：讲解了如何对用户的角色进行访问规则的管理。
- 应用程序配置：讲解了如何进行应用程序的配置。

在本章中详细讲解的登录控件中，必须在 ASP.NET 管理工具中开启相应的用户访问权限才能够让登录控件良好的运行。在最后一节中，演示了登录控件的使用和运行方法。