第2章　AJAX Control Toolkit中的文本输入处理

ASP.NET AJAX Control Toolkit提供了多个与文本输入相关的控件，用户使用这些控件能够实现多样式文本输入功能，如添加水印提示、拒绝非法字符、多样式验证、智能密码强度提示、在线智能输入建议、弹出式日历选择输入、控制并验证用户输入格式、可选择输入等。本章节介绍应用ASP.NET AJAX Control Toolkit中的控件（如TextBoxWatermark、FilteredTextBox、ValidatorCallout、PasswordStrength、AutoComplete、Calendar、PopupControl等）实现上述功能的方法。

2.1　添加水印提示的TextBoxWatermark控件

当用户在网页中的文本输入框中输入内容时，一些文本框能够显示提示信息。当焦点落在该文本框上时，显示的提示信息又会自动消失。这就是被称为水印提示的功能，效果如图2.1所示。使用ASP.NET AJAX Control Toolkit中的TextBoxWatermark控件，就能够实行该功能。

图2.1　水印提示功能

声明TextBoxWatermark扩展器控件的语法类似如下：

```
<ajaxToolkit:TextBoxWatermarkExtender
    ID="tweTextBox" runat="server"
    TargetControlID="显示水印提示的TextBox控件的ID值"
    WatermarkText="水印提示"
    WatermarkCssClass="水印提示的样式" />
```

注意：在 ASP.NET AJAX Control Toolkit 程序集中，若需要使用 TextBoxWatermark 控件，则需要使用其的全称“TextBoxWatermarkExtender”。其实，上述程序代码中已经显示了这种差别。另外，若 ASP.NET AJAX Control Toolkit 程序集中的其他控件也存在和 TextBoxWatermark 控件类似的情况时，将不再做特殊说明。

另外，TextBoxWatermark控件包含3个常用的属性：TargetControlID、WatermarkCssClass和WatermarkText。具体说明如表2-1所示。

表2-1　TextBoxWatermark控件的属性及其说明

属性	说明
TargetControlID	使用该控件的ASP.NET服务器端控件的ID值。
WatermarkText	水印提示
WatermarkCssClass	水印提示的样式

WatermarkText属性指定TextBox控件（TextBoxWatermark控件的TargetControlID属性指定的控件）的水印提示。若要在TextBox控件中显示水印提示“请输入用户名称”，则只要把TextBoxWatermark控件的WatermarkText属性的值设置为“请输入用户名称”即可。

在下述代码实例中，WaterMark.aspx页面演示了为TextBox控件添加水印提示的功能。

TextBox控件的ID属性的值为tbInput，TextBoxWatermarkExtender控件的ID属性的值为tweInput。tweInput控件为tbInput控件显示水印提示信息“请输入数据……”。

```
<!-- AjaxTextInput/WaterMark.aspx页面 -->
<%@ Page Language="C#" StylesheetTheme="ASPNETAjaxWeb"%>
<head runat="server"><title>添加水印提示</title></head>
<asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager><br />
<asp:TextBox ID="tbInput" runat="server" SkinID="tbSkin"
    Width="300px"></asp:TextBox>
<ajaxToolkit:TextBoxWatermarkExtender ID="tweInput" runat="server"
    TargetControlID="tbInput" WatermarkText="请输入数据……"
    WatermarkCssClass="Watermark">
    </ajaxToolkit:TextBoxWatermarkExtender>
```

上述代码实例的执行结果如图2.2所示。

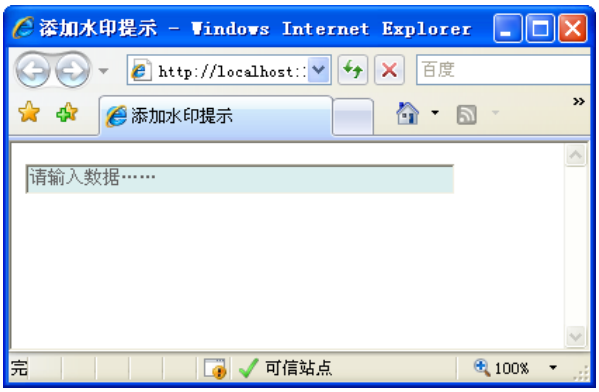

图2.2　演示水印提示的功能

另外，tweInput控件还设置了水印提示的样式（即WatermarkCssClass属性的值）为“Watermark”。该样式定义了水印提示的背景颜色和文本颜色，其程序代码如下。

```
.Watermark
{
    background-color:Gray;
    color:#666666;
}
```

2.2　拒绝非法字符的FilteredTextBox控件

许多网站在注册新用户或申请新邮箱时，用户或邮箱的名称一般不能包含一些特殊字符，如“@”、“#”、“/”等。在ASP.NET Web应用程序中，程序员可以通过程序的后台功能（如.cs文件中的功能）或正则表达式来判断用户是否输入了这些特殊字符，从而防止用户输入这些特殊字符。然而，上述实现方法往往是比较复杂。ASP.NET AJAX Control Toolkit中的FilteredTextBox控件能实现拒绝非法字符或过滤指定的字符的功能。声明FilteredTextBox扩展器控件的语法类似如下。

```
<ajaxToolkit:FilteredTextBoxExtender
    ID="fteTextBox" runat="server"
    TargetControlID="被过滤字符的TextBox控件的ID值"
```

```
    FilterType="过滤类型"
    ValidChars="合法字符集合" />
```

另外，FilteredTextBox控件包含5个常用的属性：TargetControlID、FilterType、FilterMode、ValidChars和InvalidChars。具体说明如表2-2所示。

表2-2　FilteredTextBox控件的属性及其说明

属性	说明
TargetControlID	使用该控件的ASP.NET服务器端控件的ID值。
FilterType	过滤类型，可以为Numbers、LowercaseLetters、UppercaseLetters和Custom。
FilterMode	过滤模式。
ValidChars	合法字符集合。
InvalidChars	非法字符集合。

FilterType属性指定了过滤字符的类型。它的值可以为“Numbers”、“LowercaseLetters”、“UppercaseLetters”和“Custom”。其中，“Numbers”表示数字；“LowercaseLetters”表示小写英文字母；“UppercaseLetters”表示大写英文字母；“Custom”表示自定义字符集合。另外，前3个值可以任意组合起来使用，值之间使用逗号（,）分割。譬如，“Numbers,LowercaseLetters”表达式指定数字和小写英文字母。

FilterMode属性指定过滤字符的模式，它的值可以为“ValidChars”或者“InvalidChars”。默认值为“ValidChars”。ValidChars属性指定合法字符集合，InvalidChars属性指定非法字符集合。如果FilterType属性的值为“Custom”，那么ValidChars属性可以定义任意合法字符集合、InvalidChars属性可以定义任意非法字符集合。

在下述代码实例中，Filter.aspx页面演示了为TextBox控件（ID属性的值为tbInput）拒绝非法字符的功能。FilteredTextBoxExtender控件的ID属性的值为fteInput。它的FilterMode属性的值为“ValidChars”，FilterType属性的值为“UppercaseLetters,LowercaseLetters,Numbers”。因此，tbInput控件只能接收数字、英文字母（包括大写和小写）。

```
<!-- AjaxTextInput/Filter.aspx页面 -->
<%@ Page Language="C#" StylesheetTheme="ASPNETAjaxWeb" %>
<head runat="server"><title>拒绝非法字符</title></head>
<asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager>
<br />请在下面输入框中输入大写英文字母、小写英文字母和数字。<br />
<asp:TextBox ID="tbInput" runat="server" SkinID="tbSkin"
   Width="300px"></asp:TextBox>
<ajaxToolkit:FilteredTextBoxExtender ID="fteInput" runat="server"
   TargetControlID="tbInput" FilterMode="ValidChars"
   FilterType="UppercaseLetters,LowercaseLetters,Numbers">
   </ajaxToolkit:FilteredTextBoxExtender>
```

上述代码实例的执行结果如图2.3所示。

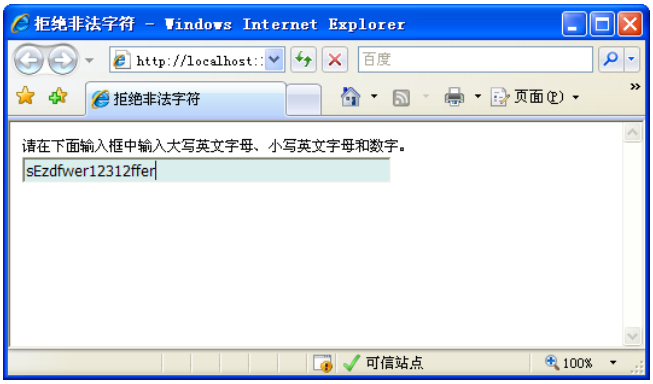

图2.3　演示拒绝非法字符的功能

2.3　多样式验证的ValidatorCallout控件

ASP.NET中的验证控件能够在网页或对话框上显示验证结果。虽然ASP.NET验证能够达到预期的效果，但是验证结果不一定能够非常醒目或更加友好地提供给用户。ASP.NET AJAX Control Toolkit中的ValidatorCallout控件能够实现以非常醒目或更加友好好方式把验证信息显示给用户的功能，效果如图2.4所示。

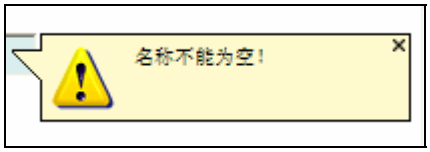

图2.4　多样式验证效果图

声明ValidatorCallout扩展器控件的语法类似如下：

```
<ajaxToolkit:ValidatorCalloutExtender
    runat="Server"
    ID="vceValidator"
    TargetControlID="验证控件的ID属性的值"
    Width="350px"
    HighlightCssClass="样式"
    WarningIconImageUrl="警告按钮图像的地址"
    CloseImageUrl="关闭按钮图像的地址" />
```

另外，ValidatorCallout控件包含多个常用的属性，如TargetControlID、Width、Animations、CloseImageUrl、WarningIconImageUrl等。具体说明如表2-3所示。

表2-3　ValidatorCallout控件的属性及其说明

属性	说明
TargetControlID	使用该控件的ASP.NET服务器端控件的ID值。
Width	宽度。
CloseImageUrl	指定Close图像。
WarningIconImageUrl	指定警告图像。
HighlightCssClass	指定验证结果的样式。
Animations	显示或隐藏验证结果的动画。

OnShow	显示验证结果时的动画。
OnHide	隐藏验证结果时的动画。

Width属性指定控件的宽度。CloseImageUrl和WarningIconImageUrl属性分别指定显示验证结果面板中的关闭按钮图像和警告按钮图像。Animations属性指定显示或隐藏验证结果的动画。

注意：ValidatorCallout 控件只能应用于验证控件，不能应用于 TextBox 控件。

在下述代码实例中，Validator.aspx页面演示了为TextBox控件（ID属性的值为tbInput）添加多样式验证的功能。Validator.aspx页面声明了2个RequiredFieldValidator控件和1个RegularExpressionValidator控件，ID属性的值分别为rfInputBlank、rfInputValue和revInput。

其中，rfInputBlank控件验证tbInput控件的内容不能为空。rfInputValue控件验证tbInput控件的内容不能为wmeInput控件（TextBoxWatermarkExtender类型的控件）的水印值“请输入名称”。revInput控件验证tbInput控件内容的最大长度为50，其正则表达式为“.{1,50}”。

注意：Validator.aspx 页面声明了 3 个 ValidatorCalloutExtender 控件，ID 属性的值分别为 vceInputBlank、vceInputValue 和 vceInputRegex。它们分别以多样式形式显示 rfInputBlank、rfInputValue 和 revInput 控件的验证结果。

```
<!-- AjaxTextInput/Validator.aspx页面 -->
<%@ Page Language="C#" StylesheetTheme="ASPNETAjaxWeb" %>
<head runat="server"><title>多样式验证</title></head>
<asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager>
<br />请在下面输入框中输入名称：<br />
<asp:TextBox ID="tbInput" runat="server" SkinID="tbSkin"
   Width="300px"></asp:TextBox>
<asp:RequiredFieldValidator ID="rfInputBlank" runat="server"
   ControlToValidate="tbInput" Display="none"
   ErrorMessage="名称不能为空！"></asp:RequiredFieldValidator>
<asp:RequiredFieldValidator ID="rfInputValue" runat="server"
   ControlToValidate="tbInput" Display="none" InitialValue="请输入名称"
   ErrorMessage="名称不能为空！"></asp:RequiredFieldValidator>
<asp:RegularExpressionValidator ID="revInput" runat="server"
   ControlToValidate="tbInput" Display="none"
   ErrorMessage="名称的长度最大为50，请重新输入。"
   ValidationExpression=".{1,50}"></asp:RegularExpressionValidator>
<ajaxToolkit:TextBoxWatermarkExtender ID="wmeInput" runat="server"
   TargetControlID="tbInput" WatermarkText="请输入名称"
   WatermarkCssClass="Watermark">
   </ajaxToolkit:TextBoxWatermarkExtender>
<ajaxToolkit:ValidatorCalloutExtender ID="vceInputBlank" runat="server"
   TargetControlID="rfInputBlank" HighlightCssClass="Validator">
   </ajaxToolkit:ValidatorCalloutExtender>
<ajaxToolkit:ValidatorCalloutExtender ID="vceInputValue" runat="server"
   TargetControlID="rfInputValue" HighlightCssClass="Validator">
   </ajaxToolkit:ValidatorCalloutExtender>
<ajaxToolkit:ValidatorCalloutExtender ID="vceInputRegex" runat="server"
   TargetControlID="revInput" HighlightCssClass="Validator">
```

```
        </ajaxToolkit:ValidatorCalloutExtender>
    <asp:Button ID="btnSure" runat="server" Text="提交" />
```

上述代码实例的执行之后，单击【提交】按钮，如图2.5所示。此时，用户未输入任何内容。

图2.5　演示多样式验证的功能

2.4　智能密码强度提示的PasswordStrength控件

智能密码强度提示功能是非常实用的一种密码提示功能。它能够告诉用户自己所输入密码的安全性的强弱，效果图如图2.6所示。单个ASP.NET服务器端控件不能提供该功能。值得幸运的是，ASP.NET AJAX Control Toolkit中的PasswordStrength控件能够实现智能密码强度提示的功能。

图2.6　智能密码强度提示效果图

声明PasswordStrength控件的语法类似如下：

```
    <ajaxToolkit:PasswordStrength ID="PS" runat="server"
        TargetControlID="TextBox控件的ID属性的值"
        DisplayPosition="显示位置"
        StrengthIndicatorType="强度提示方式"
        PreferredPasswordLength="10"
        PrefixText="Strength:"
        TextCssClass="提示文本样式"
        MinimumNumericCharacters="0"
        MinimumSymbolCharacters="0"
        RequiresUpperAndLowerCaseCharacters="false"
        TextStrengthDescriptions="很差;差;一般;好;很好"
        TextStrengthDescriptionStyles="样式1;样式2;样式3;样式4;样式5"
        CalculationWeightings="50;15;15;20" />
```

另外，PasswordStrength控件包含多个常用属性，如TargetControlID、DisplayPosition、StrengthIndicatorType、HelpStatusLabelID等。具体说明如表2-4所示。

表2-4　PasswordStrength控件的属性及其说明

属性	说明
TargetControlID	使用该控件的ASP.NET服务器端控件的ID值。
HelpStatusLabelID	显示帮助文本的Label控件的ID属性的值。
DisplayPosition	强度提示显示的位置。
HelpHandlePosition	显示帮助文本的Label控件的位置。
StrengthIndicatorType	强度提示的类型，可以为Text或BarIndicator。
PreferredPasswordLength	密码的首选长度。
PrefixText	如果强度提示类型为Text，该属性指定提示文本的前缀。
MinimumNumericCharacters	数字字符的最小数量。
MinimumSymbolCharacters	特殊字符（symbol）的最小数量。
RequiresUpperAndLowerCaseCharacters	是否要求大小写混合形式。
TextStrengthDescriptions	强度提示的描述文本，它是一个分号分割的字符串列表。
TextCssClass	如果强度提示类型为Text，该属性指定提示文本的样式。
BarBorderCssClass	图提示的边框样式。
BarIndicatorCssClass	图提示的文本样式。
HelpHandleCssClass	帮助文本的Label控件的样式。
TextStrengthDescriptionStyles	分号分割的样式列表。每一种样式将应用于其对应的强度提示的文本。
CalculationWeightings	密码组成部分所占的比重，其值的格式为“A;B;C;D”。其中，A表示长度的比重，B表示数字的比重，C表示大写的比重，D表示特殊字符的比重。A、B、C、D四个值的和必须为100，默认值为“50;15;15;20”。

StrengthIndicatorType属性指定强度提示的类型，它的值可以为“Text”或“BarIndicator”。如果为“Text”，则使用文本显示提示信息。如果为“BarIndicator”，则使用图像显示提示信息。MinimumNumericCharacters和MinimumSymbolCharacters属性分别指定数字和特殊字符的最小数量。

TextStrengthDescriptions属性指定描述密码强度的字符串，它的值由分号（;）分割。下述代码实例描述了5种强度（很差、差、一般、好和很好）的密码。

```
<ajaxToolkit:PasswordStrength ID="ps" runat="server"
    TextStrengthDescriptions="很差;差;一般;好;很好"
    ……
/>
```

TextStrengthDescriptionStyles属性指定用于每一种强度提示信息的样式，它的值由分号（;）分割。下述代码实例分别为5种强度指定了相应的样式。

```
<ajaxToolkit:PasswordStrength ID="ps" runat="server"
    TextStrengthDescriptions="很差;差;一般;好;很好"
    TextStrengthDescriptionStyles="样式1;样式2;样式3;样式4;样式5"
    ……
/>
```

CalculationWeightings属性指定密码组成的各个部分所占的比重，它的值的格式为“A;B;C;D”。其中，A表示长度的比重，B表示数字的比重，C表示大写的比重，D表示特殊字符的比重。

注意：CalculationWeightings 属性中的 A、B、C 和 D 这 4 个值都必须为整数，且它们的和为 100。该属性的默认值为“50;15;15;20”。

在下述代码实例中，PasswordStrength.aspx页面演示了为TextBox控件（ID属性的值为

tbInput）提供了智能密码强度提示的功能。PasswordStrength控件的ID值为psInput。该控件设置数字字符和特殊字符的最小数量都为2，密码组成的各个部分所占的比重（CalculationWeightings属性的值）为“40;20;20;20”。

```
<!-- AjaxTextInput/PasswordStrength.aspx页面 -->
<%@ Page Language="C#" StylesheetTheme="ASPNETAjaxWeb" %>
<head runat="server"><title>智能密码强度提示</title></head>
<asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager><br />
<asp:TextBox ID="tbInput" runat="server" SkinID="tbSkin"
   Width="300px"></asp:TextBox><br />
<asp:Label ID="lbHelp" runat="server"></asp:Label>
<ajaxToolkit:PasswordStrength ID="psInput" runat="server"
   TargetControlID="tbInput" DisplayPosition="RightSide"
   TextCssClass="PasswordStrengthText" HelpHandlePosition="BelowLeft"
   HelpStatusLabelID="lbHelp" MinimumNumericCharacters="2"
   MinimumSymbolCharacters="2" PreferredPasswordLength="10"
   RequiresUpperAndLowerCaseCharacters="true"
   StrengthIndicatorType="Text"
   TextStrengthDescriptions="很差;差;一般;好;很好"
   CalculationWeightings="40;20;20;20"></ajaxToolkit:PasswordStrength>
```

上述代码实例的执行结果如图2.7所示。

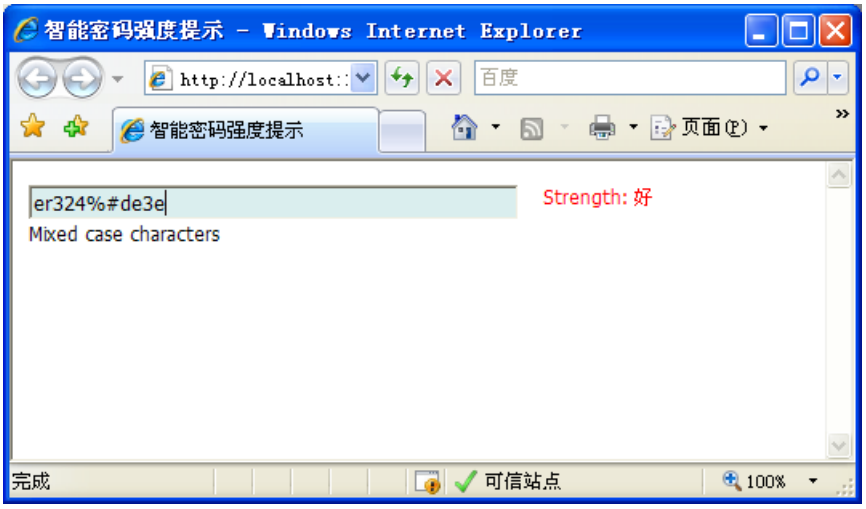

图2.7　演示智能密码提示的功能

注意：为了更加清楚说明智能密码提示的效果，特意没有把 tbInput 控件的 TextMode 属性的值设置为 Password。

2.5　在线智能输入建议的AutoComplete控件

在线智能输入建议是目前网站上一个非常流行的功能。该功能能够根据用户的输入显示一个在线提示列表。因此，用户只要输入部分关键字，就能够从在线提示列表中选择所需要的关键字。典型的效果如图2.8所示。

图2.8　在线智能输入建议效果图

如果使用ASP.NET和javascript技术实现在线智能输入建议这一个功能，是非常复杂且比较繁琐的事情。ASP.NET AJAX Control Toolkit中的AutoComplete控件能够实现在线智能输入建议的功能。声明AutoComplete扩展器控件的语法类似如下：

```
<ajaxToolkit:AutoCompleteExtender
    runat="server" ID="ace"
    TargetControlID="TextBox控件"
    ServiceMethod="获取建议的方法的名称"
    ServicePath="获取建议的Web服务"
    MinimumPrefixLength="2"
    CompletionInterval="1000"
    EnableCaching="true"
    CompletionSetCount="20"
    CompletionListCssClass="提示列表的样式"
    CompletionListItemCssClass="未选择项的样式"
    CompletionListHighlightedItemCssClass="选择项的样式"
    DelimiterCharacters=";, :">
    <Animations>
        <OnShow> ... </OnShow>
        <OnHide> ... </OnHide>
    </Animations>
</ajaxToolkit:AutoCompleteExtender>
```

另外，AutoComplete控件包含多个常用属性，如TargetControlID、MinimumPrefixLength、ServiceMethod、ServicePath、ContextKey等。具体说明如表2-5所示。

表2-5　AutoComplete控件的属性及其说明

属性	说明
TargetControlID	使用该控件的ASP.NET服务器端控件的ID值。
MinimumPrefixLength	获取建议文本的最小字符数量。
CompletionInterval	获取建议文本之前等待的时间。
EnableCaching	是否使用缓存。
CompletionSetCount	建议文本的数量。
CompletionListCssClass	建议列表的样式。
CompletionListItemCssClass	未被选择的每一项建议文本的样式。
CompletionListHighlightedItemCssClass	被选择的建议文本的样式。
DelimiterCharacters	分割字符集合。
FirstRowSelected	指定建议文本中的第一项是否被选择。

Animations	建议文本的动画。
OnShow	显示建议文本时的动画。
OnHide	隐藏建议文本时的动画。
ServiceMethod	Web服务方法的名称。
ServicePath	Web服务的路径（相对路径）。
ContextKey	设置Web服务方法的prefixText参数的值。
UseContextKey	指定是否使用ContextKey参数。

MinimumPrefixLength属性指定一个整数值。当用户输入内容的长度大于或等于该值时，AutoComplete控件将为输入框显示在线提示列表。CompletionInterval属性设置显示在线提示列表之前等待的时间。CompletionSetCount属性指定在线提示列表一次返回建议的最大数量。

ServicePath属性指定获取在线建议的Web服务；ServiceMethod属性指定获取在线建议的Web服务中的方法的名称。该方法必须满足以下代码实例中的签名。

```
[System.Web.Services.WebMethod]
[System.Web.Script.Services.ScriptMethod]
public string[] GetCompletionList(string prefixText, int count)
{
    ...
}
```

ServiceMethod属性指定的方法必须满足以下3个条件。

- 参数列表必须为“string prefixText, int count”。其中，prefixText参数的值等于ContextKey属性的值，count参数的值等于CompletionSetCount属性的值。
- 方法的返回类型必须为“string[]”，且使用“public”修饰。
- 必须为方法添加“System.Web.Script.Services.ScriptMethod”属性，使得脚本能够调用该方法。

注意：ServiceMethod 属性指定的方法的名称可以随意命名，没有硬性规定。

下述实例代码创建了一个名称为AjaxService的Web服务，并在该Web服务中引入了4个新的命名空间：System.Data、System.Web.Script.Services、AjaxControlToolkit和System.IO。AjaxService Web服务定义了一个类型为string[]的静态变量autoCompleteTextList，它用来保存在线建议的内容。定义AjaxService Web服务的程序代码如下。

```
///引入新的命名空间
using System.Data;
using System.Web.Script.Services;
using AjaxControlToolkit;
using System.IO;
[System.Web.Script.Services.ScriptService()]   ///添加脚本服务
public class AjaxService:System.Web.Services.WebService
{
    public static string[] autoCompleteTextList = null;
    public AjaxService(){}
    ……
}
```

在下述程序代码中，AjaxService Web服务定义了一个名称为GetTextList的Web方法。该方法从data.txt文件（存放在AjaxTextInput应用程序的根目录下）中获取在线建议。具体实现步骤如下：

（1）判断prefixText和count参数是否合法。如果不合法，则中止方法。

（2）如果autoCompleteTextList变量的值为空，则从data.txt文件中获取数据。其中，以按行方式读取data.txt文件，并保存到临时数组tempTextList，并且对tempTextList数组进行排序。最后将排序好的内容设置为autoCompleteTextList变量的值。

（3）使用二叉树搜索法在autoCompleteTextList变量中搜索prefixText参数的值所在位置，并保存搜索的索引。

（4）如果未搜索到prefixText参数的值，则把索引设置为0。

（5）根据索引和count参数的值获取在线建议内容，并复制到matchResultList数组中。

（6）最后返回matchResultList数组，该数组的内容就是在线建议列表中的内容。

```
[System.Web.Services.WebMethod()]
[System.Web.Script.Services.ScriptMethod()]
public string[] GetTextList(string prefixText,int count)
{   ///检测参数是否为空
    if(string.IsNullOrEmpty(prefixText) == true || count <= 0) return null;
    if(autoCompleteTextList == null)
    {   ///获取data.txt文件的数据
        StreamReader reader = new StreamReader(Server.MapPath("data.txt"));
        ///按行方式读取data.txt文件的数据
        ArrayList list = new ArrayList();
        string rowString = reader.ReadLine();
        while(rowString != null)
        {   ///读取一行
            list.Add(rowString);
            rowString = reader.ReadLine();
        }
        reader.Close();
        ///将获取的保存到临时数组中
        string[] tempTextList = new string[list.Count];
        int i = 0;
        foreach(string s in list){tempTextList[i++] = s;}
        ///对数组进行排序
        Array.Sort(tempTextList,new CaseInsensitiveComparer());
        autoCompleteTextList = tempTextList;
    }
    ///定位二叉树搜索的起点
    int index = Array.BinarySearch(autoCompleteTextList,prefixText,
        new CaseInsensitiveComparer());
    if(index < 0)
    {   ///修正起点
        index = ~index;
    }
    ///搜索符合条件的数据
    int matchCount = 0;
    for(matchCount = 0; matchCount < count
        && matchCount + index < autoCompleteTextList.Length;
        matchCount++)
    {   ///查看开头字符串相同的项
```

```
            if(autoCompleteTextList[index + matchCount].StartsWith(prefixText,
                StringComparison.CurrentCultureIgnoreCase) == false)
            {break;}
        }
        ///处理搜索结果
        string[] matchResultList = new string[matchCount];
        if(matchCount > 0)
        {   ///复制搜索结果
            Array.Copy(autoCompleteTextList,index,matchResultList,0,
                matchCount);
        }
        return matchResultList;
    }
```

另外，UseContextKey属性表示是否使用ContextKey属性的值。如果不使用，则需要把UseContextKey属性的值设置为false。

在下述代码实例中，AutoSuggest.aspx页面演示了为TextBox控件（ID属性的值为tbInput）添加在线智能输入建议的功能。AutoCompleteExtender控件的ID值为aceInput，该控件为tbInput控件添加在线建议。ServicePath属性的值为AjaxService.asmx，ServiceMethod属性的值为上述代码实例中创建的方法GetTextList。

```
    <!-- AjaxTextInput/AutoSuggest.aspx页面 -->
    <%@ Page Language="C#" StylesheetTheme="ASPNETAjaxWeb" %>
    <head runat="server"><title>在线智能输入建议</title></head>
    <asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager><br />
    <asp:TextBox ID="tbInput" runat="server" SkinID="tbSkin"
        Width="300px"></asp:TextBox>
    <ajaxToolkit:AutoCompleteExtender ID="aceInput" runat="server"
        MinimumPrefixLength="1" ServicePath="AjaxService.asmx"
        ServiceMethod="GetTextList"
        TargetControlID="tbInput"></ajaxToolkit:AutoCompleteExtender>
```

上述代码实例的执行之后，在输入框中输入字符“A”，此时将显示在线建议列表，如图2.9所示。

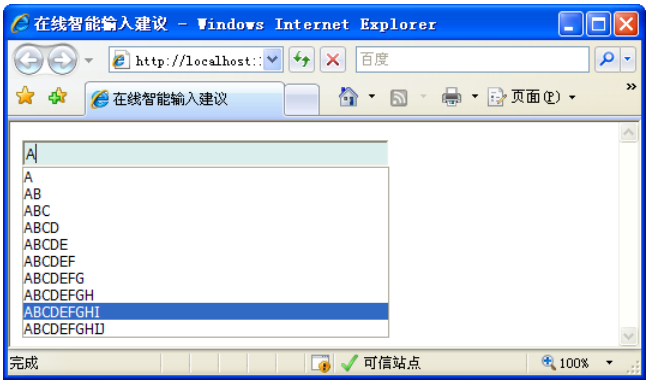

图2.9　演示在线智能输入建议的功能

2.6　弹出式日历选择输入的Calendar控件

当应用程序要求用户输入日期时，往往要规定用户输入信息的格式（即用户的输入必须满足日期格式）。否则，应用程序不能接受用户的输入内容。如果在后台代码文件中判断用户输入内容的格式，那么这是一个相对比较复杂的事情。因此，为了避免这种验证，应用程序可以提供一个日历控件让用户来选择所输入的日期。ASP.NET AJAX Control Toolkit中的Calendar控件能够实现弹出式日历选择输入的功能。声明Calendar控件的语法类似如下：

```
<ajaxToolkit:Calendar ID=cDate runat="server"
   TargetControlID="Date1"
   CssClass="ClassName"
   Format="MMMM d, yyyy"
   PopupButtonID="Image1" />
```

另外，Calendar控件包含4个常用的属性：TargetControlID、CssClass、PopupButtonID和Format。具体说明如表2-6所示。

表2-6　Calendar控件的属性及其说明

属性	说明
TargetControlID	使用该控件的ASP.NET服务器端控件的ID值。
CssClass	样式类。
Format	日期的格式字符串。
PopupButtonID	弹出日历控件的ASP.NET服务器端控件的ID值。

Format属性指定日期的格式，如“yyyy-MM-dd”、“yyyy/MM/dd”、“MM/dd/yyyy”等。其中，“yyyy”表示4位年份，“MM”表示2位月份，“dd”表示2位日期。

在下述代码实例中，Calendar.aspx页面为TextBox控件（ID属性的值为tbInput）演示了弹出式日历选择输入的功能。Calendar控件的ID属性的值为ceInput。当用户单击imgPopup控件（属于Image类型的控件）时，将弹出一个日历控件，用户可以在该日历控件选择所需要输入的日期。

```
<!-- AjaxTextInput/Calendar.aspx页面 -->
<%@ Page Language="C#" StylesheetTheme="ASPNETAjaxWeb" %>
<head runat="server"><title>弹出式日历选择输入</title></head>
<asp:ScriptManager ID="sm" EnablePartialRendering="true"
   runat="server"></asp:ScriptManager><br />
<asp:TextBox ID="tbInput" runat="server" SkinID="tbSkin"
   Width="200px"></asp:TextBox>
<asp:Image ID="imgPopup" runat="server"
   ImageUrl="~/App_Themes/ASPNETAjaxWeb/Images/view.PNG" />
<ajaxToolkit:CalendarExtender ID="ceInput" runat="server"
   Format="yyyy-MM-dd" FirstDayOfWeek="Default"
   TargetControlID="tbInput"
   PopupButtonID="imgPopup"></ajaxToolkit:CalendarExtender>
```

上述代码实例的执行之后，用户单击【】（imgPopup控件）时，弹出一个日历控件，如图2.10所示。

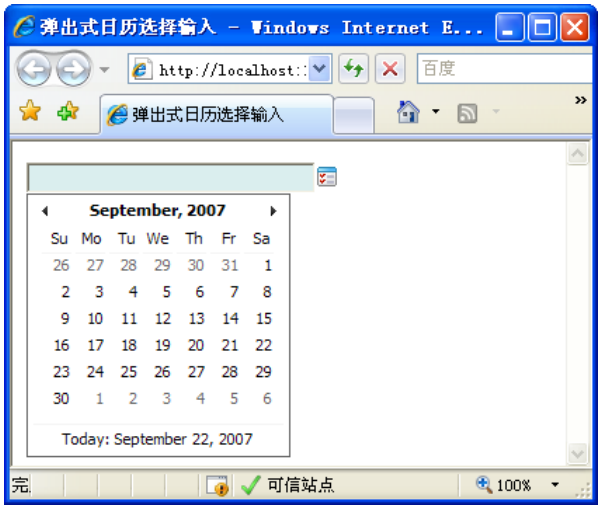

图2.10　Calendar控件演示弹出式日历选择输入的功能

2.7　弹出式日历选择输入的PopupControl控件

和Calendar控件一样，ASP.NET AJAX Control Toolkit中的PopupControl控件也能够实现弹出式日历选择输入的功能。声明PopupControl控件的语法类似如下：

```
<ajaxToolkit:PopupControlExtender ID="pce" runat="server"
    TargetControlID="TextBox控件"
    PopupControlID="被弹出的控件"
    Position="Bottom" />
```

另外，PopupControl控件包含多个常用属性，如TargetControlID、Position、OffsetX、OffsetY、PopupControlID等。具体说明如表2-7所示。

表2-7　PopupControl控件的属性及其说明

属性	说明
TargetControlID	使用该控件的ASP.NET服务器端控件的ID值。
PopupControlID	弹出控件的ID属性的值。
Position	弹出控件与目标控件相对的位置，其值可以为Left、Right、Top、Bottom或Center。
OffsetX	X偏移量。
OffsetY	Y偏移量。
CommitScript	设置弹出控制的结果时执行的脚本代码。
CommitProperty	目标控件的属性，弹出控件的值将设置为该属性的值。
Animations	弹出控件时的动画。
OnShow	弹出控件时的动画。
OnHide	隐藏控件时的动画。

PopupControlID属性指定被弹出的控件，往往是一个Panel控件。Position属性指定控件弹出的位置，它的值可以为Left、Right、Top、Bottom或Center。OffsetX和OffsetY属性分别指定控件弹出位置的偏移量。

注意：CommitProperty 属性指定的值是目标控件的某一个属性，弹出控件的结果将作为该属性的值。CommitProperty 属性的用法将在后续章节中进行介绍。

在下述代码实例中，PopupCalendar.aspx页面为TextBox控件（ID属性的值为tbInput）演示了弹出式日历选择输入的功能。PopupControlExtender控件的ID属性的值为pceInput，它弹出一个ID属性的值为pDate的Panel控件。其中，pDate控件放置了一个Calendar控件（ID属性的值为cDate），用户可以在日历控件中选择所需要的日期。

注意：为了不让用户感觉该页面的刷新操作，特意把 cDate 控件放置在 UpdatePanel 控件中。

```
<!-- AjaxTextInput/PopupCalendar.aspx页面 -->
<%@ Page Language="C#" StylesheetTheme="ASPNETAjaxWeb" %>
<head runat="server"><title>弹出式日历选择输入</title></head>
<asp:ScriptManager ID="sm" EnablePartialRendering="true"
   runat="server"></asp:ScriptManager><br />
<asp:TextBox ID="tbInput" runat="server" SkinID="tbSkin"
   Width="300px"></asp:TextBox>
<ajaxToolkit:PopupControlExtender ID="pceInput" runat="server"
   PopupControlID="pDate" OffsetX="0" OffsetY="0" Position="Bottom"
   TargetControlID="tbInput" ></ajaxToolkit:PopupControlExtender>
<asp:Panel ID="pDate" runat="server">
<asp:UpdatePanel ID="upDate" runat="server"><ContentTemplate>
<asp:Calendar ID="cDate" runat="server" BackColor="White"
   BorderColor="Black" BorderStyle="Solid" CellSpacing="1"
   Font-Names="Verdana" Font-Size="9pt" ForeColor="Black" Height="250px"
   NextPrevFormat="ShortMonth" SelectedDate="2007-08-07" Width="330px"
   OnSelectionChanged="cDate_SelectionChanged">
   <SelectedDayStyle BackColor="#333399" ForeColor="White" />
   <TodayDayStyle BackColor="#999999" ForeColor="White" />
   <DayStyle BackColor="#CCCCCC" />
   <OtherMonthDayStyle ForeColor="#999999" />
   <NextPrevStyle Font-Bold="True" Font-Size="8pt" ForeColor="White" />
   <DayHeaderStyle Font-Bold="True" Font-Size="8pt" ForeColor="#333333"
      Height="8pt" />
   <TitleStyle BackColor="#333399" BorderStyle="Solid" Font-Bold="True"
      Font-Size="12pt" ForeColor="White" Height="12pt" />
</asp:Calendar>
</ContentTemplate></asp:UpdatePanel>
</asp:Panel>
```

上述代码实例的执行之后，当用户把鼠标放置在输入框中时，输入框的下方将显示一个日历控件，用户可以在该日历控件选择所需要输入的日期，如图2.11所示。

图2.11　PopupControl控件演示弹出式日历选择输入的功能

然而，仅仅使用上述代码实例还不能获取用户选择的日期。为了把用户选择的日期显示在输入框中，特意为cDate控件添加了cDate_SelectionChanged(object sender,EventArgs e)事件。该事件调用pceInput控件的Commit()方法把用户选择的日期提交到其目标控件。

注意：虽然 cDate_SelectionChanged(object sender,EventArgs e)事件把弹出控件的结果传递到输入框，但是要在输入框中显示该结果，则需要把 ScriptManager 控件 sm 的 EnablePartialRendering 属性的值设置为 true，即及时更新输入框中的内容。

```
<script runat="server" type="text/C#">
protected void cDate_SelectionChanged(object sender,EventArgs e)
{   ///将用户选择的日期提交给TextBox。
    ///注意：ScriptManager控件的EnablePartialRendering属性的值设置为true。
    pceInput.Commit(cDate.SelectedDate.ToShortDateString());
}
</script>
```

2.8　控制并验证用户输入格式的MaskedEdit控件

在ASP.NET AJAX Control Toolkit中，MaskedEdit控件是功能非常强大的一个控件，它能够控制用户输入的字符和内容格式。如果用户输入字符或内容的格式不满足事先的设置，则控件将不会接收用户的输入。如果MaskedEdit控件设置了格式，那么它将为用户显示设置的格式，如图2.12所示。图2.12中的格式为“XXXX-XX-XX”。因此，用户只能按照该格式进行输入。

图2.12　MaskedEdit控件控制用户输入的格式

在ASP.NET AJAX Control Toolkit中，MaskedEdit控件和MaskedEditValidator控件往往一起使用。MaskedEdit控件控制用户输入的字符和内容格式，MaskedEditValidator控件验证

用户输入的内容是否满足指定的内容和格式。其中，声明MaskedEdit扩展器控件的语法类似如下：

```
<ajaxToolkit:MaskedEditExtender
    TargetControlID="TextBox控件"
    Mask="能够接收的字符集合，如9,999,999.99"
    MessageValidatorTip="true"
    OnFocusCssClass="样式"
    OnInvalidCssClass="用户输入内容不合法时的样式"
    MaskType="屏蔽的类型"
    InputDirection="文字的方向"
    AcceptNegative="接收负号（-）的方式"
    DisplayMoney="显示货币符合的方式"
    ErrorTooltipEnabled="True"/>
```

另外，MaskedEdit控件包含多个常用属性，如TargetControlID、MaskType、Filtered、AcceptNegative、UserDateFormat、UserTimeFormat等。具体说明如表2-8所示。

表2-8　MaskedEdit控件的属性及其说明

属性	说明
TargetControlID	使用该控件的ASP.NET服务器端控件的ID值。
MaskType	屏蔽的类型。
Mask	指定接收的字符和格式。
AcceptAMPM	是否接受“AM/PM”字符串。默认值为false。
AcceptNegative	是否接受负号（-）。其值可以是None、Left或Right。
AutoComplete	是否使用自动完成功能。
AutoCompleteValue	自动完成时使用的值。
ClearTextOnInvalid	当内容不合法时，是否自动清空文本。
ClipboardEnabled	是否允许粘贴或拷贝。
ErrorTooltipEnabled	是否使用错误提示功能。
Century	默认的世纪。
ClearMaskOnLostFocus	当TextBox控件失去焦点时，是否清空屏蔽字符。
ClipboardText	剪切板的文本。
DisplayMoney	货币符号显示方式。
Filtered	合法字符集合。
InputDirection	输入文本的方向。
MessageValidatorTip	当用户输入内容时，显示的提示文本。
PromptChararacter	提示字符集合。
CultureName	文化名称。
ErrorTooltipCssClass	错误提示文本的样式。
OnFocusCssClass	当焦点落在TextBox控件上时，TextBox控件的样式。
OnInvalidCssClass	当内容不合法时，TextBox控件的样式。
OnFocusCssNegative	当焦点落在TextBox控件、且内容为负值时，TextBox控件的样式。
OnBlurCssNegative	当TextBox控件失去焦点、且内容为负值时，TextBox控件的样式。
UserDateFormat	日期格式。
UserTimeFormat	时间格式。
CultureDateFormat	本地化日期格式。
CultureAMPMPlaceholder	与当前页的文化相关。
CultureCurrencySymbolPlaceholder	与当前页的文化相关。
CultureDatePlaceholder	与当前页的文化相关。

CultureDecimalPlaceholder	与当前页的文化相关。
CultureThousandsPlaceholder	与当前页的文化相关。
CultureTimePlaceholder	与当前页的文化相关。

MaskType属性指定屏蔽的类型，即验证的类型。它的值可以是以下5个值。

- None，未指定验证类型。
- Number，数字验证类型。
- Date，日期验证类型。
- Time，时间验证类型。
- DateTime，日期和时间验证类型。

Mask属性指定接收的字符和格式，它的值是一个字符串，如表示日期的“9999-99-99”（年份为4位的数字、月份和日期都为2位的数字，分割符号为“-”）、表示时间的“99:99:99”等。在Mask属性中，它可以使用通配符表示一类字符，具体说明如表2-9所示。

表2-9 Mask中的通配符及其说明

通配符	说明
9	表示数字（0～9）
L	表示一个字母，可以大写或小写。
$	表示一个字母或空格。
C	表示一个自定义字符。
A	表示一个字母或一个自定义字符。
N	表示一个数字或一个自定义字符。
?	表示任意一个字符。

在Mask属性中，除了使用通配符之外，还可以使用分割符，具体说明如表2-10所示。

表2-10 Mask中的分割符及其说明

通配符	说明
/	日期分割符。
:	时间分割符。
.	小数点。
,	千位分割符。
\	转义字符。
{	重复值的开始部分。
}	重复值的结束部分。

使用表2-8中的通配符和表2-9中的分割符可以组成多种多样的格式字符串，常用的实例如下：

- “99:99:99”，表示时间字符串。
- “9999-99-99”，表示日期字符串。
- “999999”，表示由6个数字组成的字符串。
- “9\/9”，表示由2个数字组成的、中间被字符“/”分割的字符串。

DisplayMoney属性能够指定显示货币符号（如￥、$等）的方式，具体描述如下：

- None，不显示货币符号。
- Left，在最左边显示货币符号。
- Right，在最右边显示货币符号。

AcceptNegative属性指定能否接受负号（-），具体描述如下：

- None，不显示负号。
- Left，在最左边显示负号。

- Right，在最左边显示负号。

InputDirection属性指定文本的方向，它的值可以为LeftToRight（表示从左到右）和RightToLeft（表示从右到左）。在下述实例代码中，MaskedEditExtender控件指定验证类型为日期类型（由MaskType属性指定），格式字符串为“9999-99-99”。因此，用户只能输入日期格式的数据。

```
<ajaxToolkit:MaskedEditExtender ID="meeInput" runat="server"
   Mask="9999-99-99" MaskType="Date" TargetControlID="tbInput">
   </ajaxToolkit:MaskedEditExtender>
```

在ASP.NET AJAX Control Toolkit中，MaskedEditValidator控件也是一个功能非常强大的控件，它能够验证用户输入的内容是否满足事先指定的格式。声明MaskedEditValidator控件的语法类似如下：

```
<ajaxToolkit:MaskedEditValidator
   ControlExtender="MaskedEditExtender控件"
   ControlToValidate="被验证的TextBox控件"
   IsValidEmpty="False"
   MaximumValue="最大值"
   EmptyValueMessage="空值时的提示消息"
   InvalidValueMessage="不合法时的提示消息"
   MaximumValueMessage="超过最大值的提示消息"
   MinimumValueMessage="超过最小值的提示消息"
   MinimumValue="最小值"
   EmptyValueBlurredText="*"
   InvalidValueBlurredMessage="*"
   MaximumValueBlurredMessage="*"
   MinimumValueBlurredText="*"
   Display="Dynamic"
   TooltipMessage="提示消息"/>
```

另外，MaskedEditValidator控件包含多个常用属性，如ControlToValidate、InitialValue、ControlExtender、IsValidEmpty、MaximumValue、MinimumValue等。具体说明如表2-11所示。

表2-11　MaskedEditValidator控件的属性及其说明

属性	说明
ControlToValidate	被验证控件的ID属性的值。
ControlExtender	与其相对应MaskedEditExtender控件指定的TextBox控件的ID属性的值。
AcceptAMPM	是否接受“AM/PM”字符串。默认值为false。
InitialValue	与其相对应TextBox控件的初始值。
IsValidEmpty	与其相对应TextBox控件的值为空是否合法。
MaximumValue	最大值。
MinimumValue	最小值。
ValidationExpression	验证内容的正则表达式。
ClientValidationFunction	客户端验证函数的名称。
TooltipMessage	TextBox控件具有焦点时显示的消息。
EmptyValueMessage	TextBox控件具有焦点，该控件的内容为空时显示的消息。
EmptyValueBlurredText	TextBox控件不具有焦点，该控件的内容为空时显示的消息。
InvalidValueMessage	TextBox控件具有焦点，验证不合法时显示的消息。
InvalidValueBlurredMessage	TextBox控件不具有焦点，验证不合法时显示的消息。

MaximumValueMessage	TextBox控件具有焦点，控件的内容超过最大值时显示的消息。
MaximumValueBlurredMessage	TextBox控件不具有焦点，控件的内容超过最大值时显示的消息。
MinimumValueMessage	TextBox控件具有焦点，控件的内容超过最小值时显示的消息。
MinimumValueBlurredText	TextBox控件不具有焦点，控件的内容超过最小值时显示的消息。

在下述代码实例中，MaskedEdit.aspx页面为TextBox控件（ID属性的值为tbInput）演示了控制并验证用户输入格式的功能。MaskedEditExtender和MaskedEditValidator控件的ID属性的值分别为meeInput和mevInput。meeInput控件指定用户只能输入“9999-99-99”格式的日期，mevInput控件将验证用户输入的格式是否正确。如果不正确，则显示提示消息。

```
<!-- AjaxTextInput/MaskedEdit.aspx页面 -->
<%@ Page Language="C#" StylesheetTheme="ASPNETAjaxWeb"%>
<head runat="server"><title>控制用户输入格式并验证</title></head>
<asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager><br />
<asp:TextBox ID="tbInput" runat="server" SkinID="tbSkin"
    Width="300px"></asp:TextBox>
<ajaxToolkit:MaskedEditExtender ID="meeInput" runat="server"
    Mask="9999-99-99" MaskType="Date" TargetControlID="tbInput">
    </ajaxToolkit:MaskedEditExtender>
<ajaxToolkit:MaskedEditValidator ID="mevInput" runat="server"
    ControlToValidate="tbInput" ControlExtender="meeInput"
    IsValidEmpty="false" EmptyValueMessage="日期不能为空"
    InvalidValueMessage="日期的格式不正确"
    TooltipMessage="请输入日期，格式为：yyyy-MM-dd"
    Display="Dynamic"></ajaxToolkit:MaskedEditValidator>
```

上述代码实例的执行结果如图2.13所示。

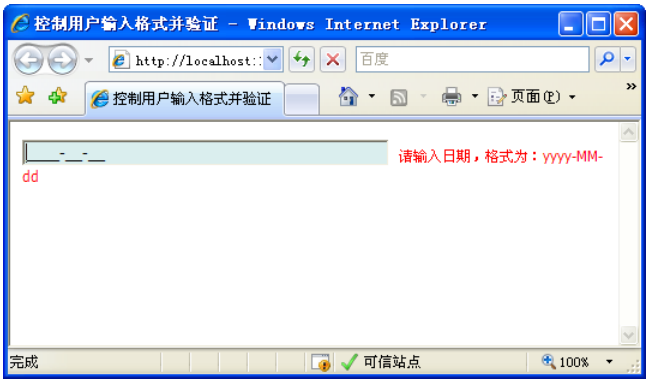

图2.13　演示控制并验证用户输入格式的功能

2.9　可选择输入的PopupControl控件

ASP.NET AJAX Control Toolkit中的PopupControl控件不但可以实现弹出式日历选择输入功能，也可以实现输入框的可选择输入的功能。

在下述代码实例中，SelectText.aspx页面为TextBox控件（ID属性的值为tbInput）演示了可选择输入的功能。PopupControlExtender控件的ID属性的值为pceInput，它弹出一个ID属性为pText的Panel控件。其中，pText控件放置了一个RadioButtonList控件（ID属性的值为rblText），用户可以在单项列表控件中选择所需要的数据项。

注意：为了不让用户感觉该页面的刷新操作，特意把 rblText 控件放置在 UpdatePanel 控件中。

```
<!-- AjaxTextInput/SelectText.aspx页面 -->
<%@ Page Language="C#" AutoEventWireup="true"
    CodeFile="SelectText.aspx.cs" StylesheetTheme="ASPNETAjaxWeb"
    Inherits="SelectText" %>
<head runat="server"><title>可选择输入</title></head>
<asp:ScriptManager ID="sm" EnablePartialRendering="true"
    runat="server"></asp:ScriptManager><br />
<asp:TextBox ID="tbInput" runat="server" SkinID="tbSkin"
    Width="300px"></asp:TextBox>
<ajaxToolkit:PopupControlExtender ID="pceInput" runat="server"
    PopupControlID="pText" OffsetX="0" OffsetY="0" Position="Bottom"
    TargetControlID="tbInput" ></ajaxToolkit:PopupControlExtender>
<asp:Panel ID="pText" runat="server">
<asp:UpdatePanel ID="upText" runat="server"><ContentTemplate>
    <asp:RadioButtonList ID="rblText" runat="server" CssClass="Text"
        RepeatDirection="Vertical"
        OnSelectedIndexChanged="rblText_SelectedIndexChanged"
        AutoPostBack="True" RepeatLayout="Flow"></asp:RadioButtonList>
</ContentTemplate></asp:UpdatePanel>
</asp:Panel>
```

上述代码实例的执行，当用户把鼠标放置在输入框中时，输入框的下方将显示一个单项列表控件，用户可以在该单项列表控件选择所需要输入的数据项，如图2.14所示。

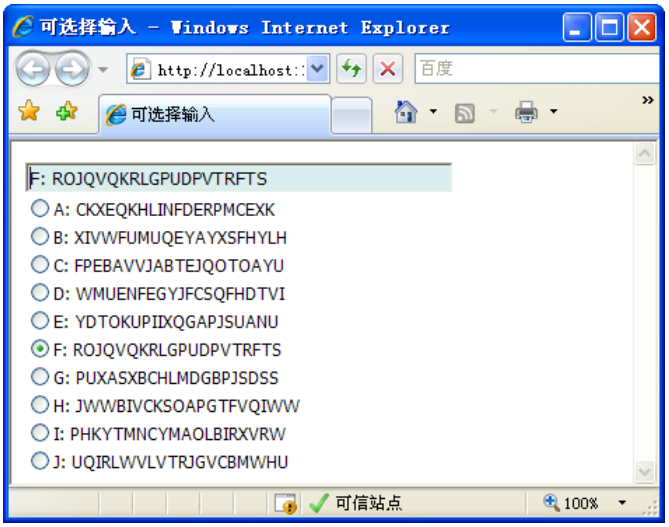

图2.14　演示可选择输入的功能

然而，仅仅使用上述代码实例还不能获取用户选择的数据项。为了把用户选择的数据项显示在输入框中，特意为rblText控件添加了rblText_SelectedIndexChanged(object sender, EventArgs e)事件。该事件调用pceInput控件的Commit()方法把用户选择的数据项提交到其目标控件。

注意：虽然 rblText_SelectedIndexChanged(object sender,EventArgs e)事件把弹出控件的结果传递到输入框，但是要在输入框中显示该结果，则需要把 ScriptManager 控件 sm 的 EnablePartialRendering 属性的值设置为 true，即及时更新输入框中的内容。

```
protected void rblText_SelectedIndexChanged(object sender,EventArgs e)
{   ///将用户选择的值提交给TextBox。
    ///注意：ScriptManager控件的EnablePartialRendering属性的值设置为true。
    pceInput.Commit(rblText.SelectedItem.Text);
}
```

在下述程序代码中，Page_Load(object sender, EventArgs e)事件初始化SelectText.aspx页面，即调用BindPageData()函数创建rblText控件的数据项。其中，每一个数据项均为一个随机字符串，由CreateRandomString()函数创建。

```
Random random = new Random();
protected void Page_Load(object sender, EventArgs e)
{   ///初始化页面的数据
    if(!Page.IsPostBack){BindPageData();}
}
/// <summary>
/// 创建被选项控件的数据源
/// </summary>
private void BindPageData()
{   ///创建数据表
    DataTable dt = new DataTable();
    dt.Columns.Add(new DataColumn("ID",typeof(int)));
    dt.Columns.Add(new DataColumn("Name",typeof(string)));
    ///添加表的数据
    for(int i = 0; i < 10; i++)
    {
        DataRow row = dt.NewRow();
        row["ID"] = i;
        row["Name"] = ((char)('A' + i)).ToString() + ": "
            + CreateRandomString();
        dt.Rows.Add(row);
    }
    ///绑定并显示数据
    rblText.DataSource = dt;
    rblText.DataTextField = "Name";
    rblText.DataValueField = "ID";
    rblText.DataBind();
}
```

在下述程序代码中，CreateRandomString()函数创建一个长度为20的、由大写英文字母组成的随机字符串。

```
/// <summary>
/// 产生随机字符串
/// </summary>
private string CreateRandomString()
{   ///创建长度为20的随机字符串
    System.Text.StringBuilder sbString = new System.Text.StringBuilder();
```

```
        for(int i = 0; i < 20; i++)
        {
            sbString.Append((char)random.Next((int)'A',(int)'Z'));
        }
        return sbString.ToString();
    }
```