

Ajax 和 Silverlight4 技术

第 7 章

学习要点

- (1) Ajax 的基本概念及原理。
- (2) 基于 Ajax 框架的应用开发原理和方法。
- (3) 了解 Silverlight 和 XAML 基础知识
- (4) 掌握 Silverlight 基本控件及其应用
- (5) 了解 Silverlight 事件处理
- (6) 了解 Silverlight 界面布局
- (7) 了解 Silverlight 数据服务
- (8) 理解 Silverlight 自定义控件和模板及其应用

本书第 1 章已经介绍过 Ajax 技术。它由 Jesse James Garrett 于 2005 年 2 月提出之前,实际上像 Google 公司在 Google Map 中已经使用这种技术来处理相关数据的查询和浏览,只不过没有给它以如此响亮的名字。Ajax 是 Asynchronous Javascript and XML 的简称,该技术的目标是让用户动态地与页面进行交互,加快服务器的响应速度,减少用户的等待时间,是一种创建交互式 Web 应用程序的开发技术。

为了便于读者理解 Ajax 技术的基本原理,并学会应用 Ajax 技术,本章将介绍如何利用 XMLHttpRequest 对象进行 Ajax 开发,实现与网页服务器之间的异步数据交换,随后针对 ASP.NET 平台下传统的 Web Form 程序开发和基于 Ajax 框架的程序开发进行对比研究,并通过具体案例讲解其内容和原理。

同时随着互联网行业的飞速发展,越来越多的公司认识到只有不断地提高企业的核心竞争力和创新能力,才能在激烈的市场竞争当中脱颖而出。Silverlight 的横空出世为企业应用提出了新的解决方案,可轻松构建更为直观、易于使用、反应更迅速并且可以脱机使用的应用程序,并可帮助企业提供多元化的重要业务效益,包括提高销量、提高品牌忠诚度、延长网站逗留时间、较频繁的重访问、减少带宽成本、减少支持求助以及增强客户关系等。

7.1 Ajax 概述

Ajax 通过异步数据交换和处理,可显著提高 Web 应用程序运行效率,给 Web 开发者带来了新的希望。Ajax 并不是一门新的语言或技术,它实际上是几项技术按一定的方式组合在一起共同协作中发挥各自的作用。具体来说,Ajax 基于下列核心技术:

- XHTML: 对应 W3C 的 XHTML 规范,目前是 XHTML 1.0。
- CSS: 对应 W3C 的 CSS 规范,目前是 CSS 2.0。
- DOM: 这里的 DOM 主要是指 HTML DOM。
- JavaScript: 对应于 ECMA 的 ECMAScript 规范。
- XML: 对应 W3C 的 XML DOM、XSLT、XPath 等规范。
- XMLHttpRequest: 对应 WHATWG (Web Hypertext Application Technology Working Group) 的 Web Applications 2.0 规范的一部分(<http://whatwg.org/specs/web-apps/current-work/>)。

Ajax 的工作原理相当于在用户和服务器之间加了一个中间层即 Ajax 引擎,使用户操作与服务器响应异步化。并不是所有的用户请求都提交给服务器,像一些数据验证和简单的数据处理等都交给 Ajax 引擎自己来做,只有确定需要从服务器读取新数据时再由 Ajax 引擎代为向服务器提交请求。其应用程序模型如图 7-1 所示。

Ajax 的核心是 JavaScript 对象 XMLHttpRequest。该对象在 Internet Explorer 5 中首次引入,它是一种支持异步请求的技术。XmlHttpRequest 使开发者可以使用 JavaScript 向服务器提出异步请求并处理响应,而不阻塞用户。

目前实现 Ajax 技术的方法主要有:①直接基于 XMLHttpRequest 对象;②利用各种 Ajax 框架,简化 Ajax 开发。

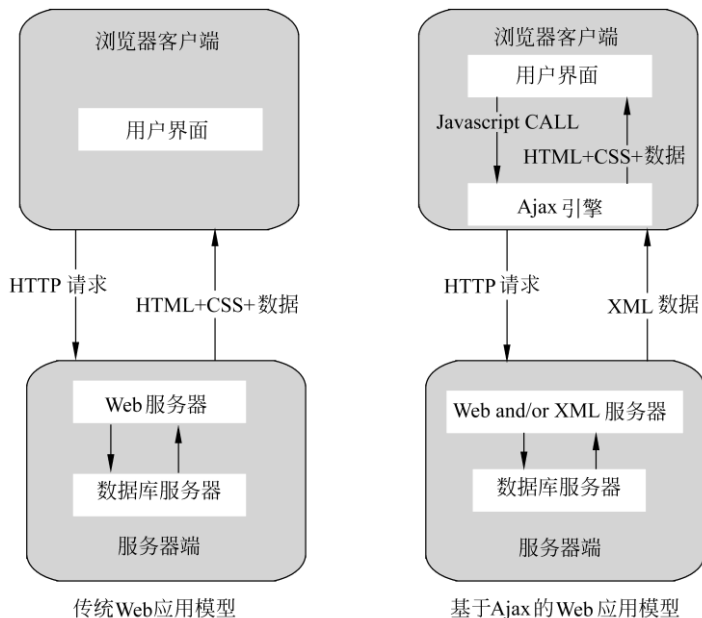


图 7-1 Ajax 与传统 Web 应用程序模型对比

7.2 用 XMLHttpRequest 实现 Ajax 技术

Ajax 的一个最大的特点是无需刷新页面便可向服务器传输或读写数据（又称无刷新更新页面），这一特点主要得益于 XMLHTTP 组件中的 XMLHttpRequest 对象。最早应用 XMLHTTP 的是微软 IE（IE 6.0 以上）允许开发人员在 Web 页面内部使用 XMLHttpRequest ActiveX 组件扩展自身的功能，开发人员可以不用从当前的 Web 页面导航而直接传输数据到服务器上或者从服务器取数据。在这种情况下，XMLHttpRequest 对象相当于起到了图 7-1 中 Ajax 引擎的作用，利用该对象减少了无状态连接的痛苦，还可以排除下载冗余 HTML，从而提高服务器的响应速度。其他浏览器 Mozilla、Firefox、Opera、Safari 等也都支持 XMLHttpRequest 对象。

XMLHttpRequest 是 Ajax 开发的基础，体现了异步调用的核心。XMLHttpRequest 对象的方法和属性如表 7-1 和表 7-2 所示。

表 7-1 XMLHttpRequest 对象方法

方 法	描 述
abort()	停止当前请求
getAllResponseHeaders()	返回完整的 headers 字符串
getResponseHeader("headerLabel")	返回单个的 header 标签字符串
open("method","URL",[syncFlag[, "userName"[, "password"]]])	设置请求的方法、目标 URL 和其他参数
send(content)	发送请求
setRequestHeader("label", "value")	设置 header 并和请求一起发送

表 7-2 XMLHttpRequest 对象属性

属 性	描 述
onreadystatechange	状态改变的事件触发器
readyState	对象状态（integer）：0=未初始化；1=读取中；2=已读取；3=交互中；4=完成
responseText	从服务器返回的数据文本
responseXML	从服务器返回兼容 DOM 的 XML 文档对象
status	服务器返回的状态码。如：404 表示“文件未找到”、200 表示“成功”
statusText	服务器返回的状态文件信息

用 XMLHttpRequest 进行 Ajax 开发的基本步骤主要包括发送 XMLHttpRequest 对象请求和获取响应信息进行数据处理两个步骤。

使用 XMLHttpRequest 对象发送请求的基本步骤是：

- （1）创建 XMLHttpRequest 对象；
- （2）指定处理函数：给 XMLHttpRequest 对象的 onreadystatechange 属性赋值，指示哪个函数处理 XMLHttpRequest 对象状态的变化；
- （3）指定请求的属性。open 方法的三个参数分别指定将发送请求的方法（通常是 GET 或 POST）、目标资源 URL 串以及是否异步请求；
- （4）发送请求到服务器：send 方法把请求传送到指定的目标资源，send 方法接受一个参数，通常是一个串或 DOM 对象。这个参数会作为请求体的一部分传送到目标 URL。向

send 方法提供参数时,要确保 open 中指定的方法是 POST。如果没有数据要作为请求体的一部分发送,则使用 null。

XMLHttpRequest 对象在大部分浏览器上已经实现而且拥有一个简单的接口允许数据从客户端传递到服务端,但并不会打断用户当前的操作。使用 XMLHttpRequest 传送的数据可以是任何格式,虽然从名字上建议是 XML 格式的数据。

【例 7.1】该例代码演示了利用 XMLHttpRequest 对象获取远程数据并显示结果的整个过程。

```
<HTML>
<HEAD><title>Welcome</title>
<script language="javascript">
    var vXMLHttpRequest;
    function CreateXMLHttpRequest() { //创建 XMLHttpRequest 对象,需考虑浏览器兼容性
        if (window.XMLHttpRequest) { // Mozilla、Firefox、Safari 等浏览器
            vXMLHttpRequest = new XMLHttpRequest();
            if (vXMLHttpRequest.overrideMimeType) {
                vXMLHttpRequest.overrideMimeType("text/xml");
            }
        } else if (window.ActiveXObject) { // IE 浏览器
            try {
                vXMLHttpRequest = new ActiveXObject("Msxml2.XMLHTTP");
            }
            catch (e) {
                vXMLHttpRequest = new ActiveXObject("Microsoft.XMLHTTP");
            }
        }
        if (!vXMLHttpRequest) { window.alert("Browser not support XMLHttpRequest!"); }
        return vXMLHttpRequest;
    }
    function ExcuteWelcome(vName) {
        vXMLHttpRequest = CreateXMLHttpRequest();
        vXMLHttpRequest.onreadystatechange = Excute_Callback; //指定处理函数
        var url = "WelcomeResult.aspx?Name=" + vName;
        vXMLHttpRequest.open("GET", url); //设置请求方法和目标
        vXMLHttpRequest.send(null); //发送请求
    }
    function Excute_Callback() {
        if (vXMLHttpRequest.readyState == 4) { //对响应数据进行处理
            if (vXMLHttpRequest.status == 200) { //信息返回成功
                var s = vXMLHttpRequest.responseText;
                MyResult.innerHTML = s; //显示数据
            }
        }
    }
}
</script>
```

```
</HEAD>
<body>
    <form id="Form1" method="post" runat="server">
        <div>请输入您的姓名: </div>
        <input id="username" type="text" size="13">
        <input type="button" value="单击" onclick=" ExcuteWelcome (document.
        getElementById('username').value)">
        <div id="MyResult"></div>
    </form>
</body>
</HTML>
```

以上代码实现在文本框输入姓名, 单击 button 按钮后, XMLHttpRequest 异步访问 WelcomeResult.aspx 获取问候语并显示。

其中函数 CreateXMLHttpRequest 根据浏览器版本不同, 完成 XMLHttpRequest 对象创建工作; 函数 ExcuteWellcome 首先创建 XMLHttpRequest 对象, 然后通过 onreadystatechange 属性指定当属性 readyState 状态发生改变时由函数 Excute_Callback 进行处理。

页面 WelcomeResult.aspx 仅根据调用的参数返回一个字符串, 在 .NET 环境中其页面的 codebehind 代码为:

```
private void Page_Load(object sender, System.EventArgs e)
{
    Response.Clear();
    string TempName = Request.QueryString["Name"];
    if (TempName != null)
    { try
        { Response.Write(TempName + ", Wellcome to Ajax's world!");
        }
        catch { }
    }
    Response.End();
}
```

其运行效果如图 7-2 所示。



图 7-2 Ajax 实例运行效果

XMLHttpRequest 还可用异步方式调用网络 Web 服务, 调用远程服务器中已经编写好的方法, 实现各种功能, 仍旧包括发送 XMLHttpRequest 对象请求和获取响应信息进行数据处理两个步骤。

远程天气预报 Web 服务免费提供了包括 340 多个国内主要城市和 60 多个国外主要城市三日内的天气预报数据, 数据来源于中国气象局, 每 2.5 小时更新一次, 准确可靠。其接口为 <http://webservice.webxml.com.cn/WebServices/WeatherWebService.asmx>, 点击该链接可查看该天气预报 Web 服务所提供的接口调用介绍, 如图 7-3 所示。下例将通过该 Web 服务提供的 getWeatherbyCityName 方法来获取某城市的天气情况。



图 7-3 天气预报 Web 服务所提供的各接口介绍

【例 7.2】该例代码演示了利用 XMLHttpRequest 对象连接远程天气预报 Web 服务接口并显示城市天气的过程。

```
<html><head><title>重庆的天气预报</title> </head>
<script type="text/javascript">
var xmlhttp=null;
window.onload = getChongqing; // 浏览器加载网页完毕后执行 onload 事件, 调用
getChongqing 函数
function getChongqing() { // 获取重庆天气信息
    // 判断浏览器类型, 根据不同的浏览器, 创建 XMLHttpRequest 对象 xmlhttp
    if (window.XMLHttpRequest) {
        xmlhttp=new XMLHttpRequest();
    }
    else if (window.ActiveXObject) {
```

```
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
    else {
        xmlhttp=new ActiveXObject("Msxml2.XMLHTTP");
    }
xmlhttp.open("GET","http://www.webxml.com.cn/WebServices/WeatherWebService.asmx/getWeatherbyCityName?theCityName=重庆",true);
xmlhttp.send(null); //发送 Web 服务请求
xmlhttp.onreadystatechange=stateChange; //当 XMLHttpRequest 对象 xmlhttp 的状态变换时,就执行 stateChange 函数。
function stateChange()
{
    if(xmlhttp.readyState==4) //判断 xmlhttp 对象的返回状态
    {
        if(xmlhttp.status==200|| xmlhttp.status==0)//如果是 404,说明未找到服务文件。0 是专门为火狐和谷歌浏览器保留的,它说明本地运行成功的情况。
        {
            var data=xmlhttp.responseXML; //获取所有返回的数据
            document.getElementById("no1").firstChild.nodeValue=data.getElementsByTagName("string")[5].firstChild.nodeValue; //显示当前温度
            document.getElementById("no2").firstChild.nodeValue=data.getElementsByTagName("string")[6].firstChild.nodeValue; //显示当前天气情况
        }
    }
}
</script>
<body>
温度: <span id="no1"></span>
天气: <span id="no2" ></span>
</body>
</html></html>
```

上述代码中可将重庆换成其他城市名实现不同城市的天气预报。通过 xmlhttp 的 open 方法连接天气预报 Web 服务, xmlhttp 的 send 方法则主要是向该 Web 服务发出处理请求。stateChange 函数体部分实现对返回的 XML 文档格式的天气数据信息进行处理,这里主要是返回当前所选地区的天气信息。根据 getWeatherbyCityName 方法可知,传入参数 theCityName 的具体值后可以查询获得该城市未来三天内天气情况、现在的天气实况、天气和生活指数等信息,如图 7-4 所示,以重庆为例。

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<ArrayOfString xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://WebXml.com.cn/?">
  <string>重庆市</string>
  <string>重庆</string>
  <string>57516</string>
  <string>57516.jpg</string>
  <string>2012-10-26 11:12:28</string>
  <string>16℃/20℃</string>
  <string>10月26日 阵雨转阴</string>
  <string>无持续风向微风</string>
  <string>3.gif</string>
  <string>2.gif</string>
  <string>今日天气实况：气温：19℃；风向/风力：东风 1级；湿度：68%；空气质量：良；紫外线强度：最弱</string>
  <string>穿衣指数：建议着薄型套装或牛仔衫裤等春秋过渡装。年老体弱宜着套装、夹克衫等。 过敏指数：天气条件极易诱发过敏，有降水，易过敏人群尽量减少外出，如需外出最好穿长衣长裤，预防感冒可能引发的过敏。 运动指数：有降水，较适宜在户内开健身和休闲运动，若坚持户外运动，注意携带雨具并注意避雨防潮。 洗车指数：不宜洗车，未来24小时内有雨，如果在此期间洗车，雨水和路上的泥水可能会再次弄脏您的爱车。 晾晒指数：有降水，可能会弄湿晾晒的衣物，不太适宜晾晒。 请随时注意天气变化。 旅游指数：有阵雨，温度适宜，在细雨中游玩别有一番情调，可不要错过机会呦！但记得出门要携带雨具。 路况指数：有降水，路面潮湿，车辆易打滑，请小心驾驶。 舒适度指数：温度适宜，风力不大，您在这样的天气条件下，会感到比较清爽和舒适。 空气污染指数：气象条件有利于空气污染物稀释、扩散和清除，可在室外正常活动。 紫外线指数：微弱紫外线辐射天气，无需特别防护。若长期在户外，建议涂擦SPF在8-12之间的防晒护肤品。
  </string>
  <string>11℃/21℃</string>
  <string>10月27日 阴转阵雨</string>
  <string>无持续风向微风</string>
  <string>2.gif</string>
  <string>3.gif</string>
  <string>11℃/20℃</string>
  <string>10月28日 阵雨转阴</string>
  <string>无持续风向微风</string>
  <string>3.gif</string>
  <string>2.gif</string>
  </ArrayOfString>
  <string>重庆市简称渝，位于我国西南地区东部，长江上游。1997年以原四川省重庆、万县、涪陵三地级市区级设中央直辖市重庆，是我国面积最大、行政辖区最广、人口最多的中央直辖市。全市面积8.3万平方千米，人口3002万，有汉、回、苗、土家等民族。长江斜贯本市，形成著名的长江三峡。中亚热带湿润季风气候，冬暖夏热，夏季因地形郁闭，气候闷热，成为长江三大“火炉”之一。年降水量1000毫米以上。重庆市区坐落在长江与嘉陵江交汇处，四面环山，江水回绕，城市傍水依山，层叠而上，既以山城著称，又以山城得名。然而最具特色的，还要数山城的夜色。凭高眺望，万家灯火起伏错落，根根明灭，与两江斑斓的波光、满天闪烁的星斗交相辉映，其景奇而醉人。重庆中心城区为长江、嘉陵江环抱，山清水秀，风貌独特。重庆具有二千年悠久历史，为重庆留下了不少驰名中外的风景名胜。古老的历史遗迹和雄峻的自然奇观。全市有大的旅游景区20多个，景点达1300多处，已初步开发3000多处，其中国家级重点风景名胜区4个，国家级文物保护单位6个，市级文物保护单位1100多个，各种文物点12000多处。重庆市拥有丰富的生物资源、矿产资源、水能资源和独具特色的三峡旅游资源，具有极大的开发潜力。重庆是中国西南地区 and 长江上游的经济中心城市、重要的交通枢纽和内河口岸，拥有汽车、机械、电力、化工、轻纺等综合性工业基地。
  </string>
</ArrayOfString>
```

图 7-4 具体城市天气情况

从图 7-4 可知，调用该天气预报 Web 服务后，返回的天气信息以 XML 文档格式存在，共有 23 个 string 节点，可放在一个 string 数组中，图中每一个<string></string>中间的元素就是数组的一个元素，可根据想要显示的天气信息来确定网页的设计。在浏览器中运行例 7.2 网页代码，其实现效果如图 7-5 所示（代码中略去了相关显示的样式）。


温 度	5℃/8℃
天 气	12月21日 小雨 

图 7-5 示例运行结果（重庆实时天气情况）

7.3 传统 Web Form 和基于 Ajax 的 Web Form 应用对比

传统的 Web Form 应用允许客户端填写表单，当提交表单时就向 Web 服务器发送一个请求。服务器接收并处理传来的表单，然后返回一个生成的完整网页，但这个做法不仅耗费带宽，而且用户响应速度慢。因为表单提交前的网页和表单提交后返回的网页往往区别不大，但每次都需将生成的完整网页返回到客户端浏览器。

与此不同，基于 Ajax 的 Web Form 应用可以仅向服务器发送并取回必须的数据，并在客户端采用 JavaScript 处理来自服务器的回应。因为在服务器和浏览器之间交换的数据大量减少，服务器响应速度快，用户体验更好。

【例 7.3】该例代码演示了传统 Web 应用和基于 Ajax 的 Web Form 应用的异同，在 VS2010 环境通过添加例 7.2 的天气预报 Web 服务来实现天气查询。为了查看实现效果，在网页上放置了一幅较大图片。

传统的 Web Form 应用，每次点击获取天气按钮就都会向 Web 服务器发送一次请求，服务器接收请求处理后，会返回一个生成的完整网页，此时明显看到整个图片被较缓慢地刷新一次，导致对用户界面的响应变长。前台 aspx 代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="get weather by web
form.aspx.cs" Inherits="Shen.get_weather_by_web_form" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server"> <title>传统的 Web 应用</title> </head>
<body>
    <form id="form1" runat="server">
        <div>
            城市: <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <br /> <br />
            温度: <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
            <br /> <br />
            天气: <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
            <br /> <br />
        </div>
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text=
"GetWeather" />
    </form>
    
</body>
</html>
```

后台.cs 代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    String []xdoc; //建立保存 Web 服务返回信息的数组
    MyWeatherWebService.WeatherWebService myWeather = new MyWeatherWebser
vice.WeatherWebService(); //新建 Web 服务实例
    xdoc = myWeather.getWeatherbyCityName("重庆"); //传入参数为“重庆”，向数
组返回重庆的天气情况
    TextBox1.Text = xdoc[1]; //在文本框中显示城市
    TextBox2.Text = xdoc[5]; //在文本框中显示温度
    TextBox3.Text = xdoc[6]; //在文本框中显示天气
}
```

基于 Ajax 的 Web Form 应用，利用其核心 XMLHttpRequest 对象，在不更新整个页面的前提下维护数据，使得 Web 应用更为迅速地回应用户请求，避免了在网络上发送那些没有改变过的信息。前台 aspx 代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="get_weather_by_Ajax.aspx.cs" Inherits="Shen.get_weather_by_Ajax" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>基于 Ajax 框架的应用</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                城市: <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                <br /> <br />
                温度: <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
                <br /> <br />
                天气: <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
                <br /> <br />
                <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
                Text="GetWeather" />
            </ContentTemplate>
        </asp:UpdatePanel>
    </form>
    </p>
</body>
</html>
```

其中的 **ScriptManager** 控件, 管理支持 Ajax 的 ASP.NET 网页的客户端脚本。必须在页面上使用 **ScriptManager** 控件, 才能启用 ASP.NET 的 Ajax 功能, 默认情况下, **ScriptManager** 控件会向页面注册 **Ajax Library** 的脚本, 这将使客户端脚本能够使用类型系统扩展并支持部分页呈现, 允许单独刷新页面上的区域而无需回发和 Web 服务调用等功能。

当页包含一个或多个 **UpdatePanel** 控件时, **ScriptManager** 控件将管理浏览器中的部分页呈现, 该控件与页生命周期进行交互, 以更新位于 **UpdatePanel** 控件内的部分页, 而不需要自定义客户端脚本。使用 **UpdatePanel** 控件可生成功能丰富的、以客户端为中心的 Web 应用程序。通过使用 **UpdatePanel** 控件, 可以刷新页的选定部分, 而不是使用回发刷新整个页面。

可以通过声明方式向 **UpdatePanel** 控件添加内容, 也可以在设计器中通过使用 **ContentTemplate** 属性来添加内容。当首次呈现包含一个或多个 **UpdatePanel** 控件的页时, 将呈现 **UpdatePanel** 控件的所有内容并将这些内容发送到浏览器。在后续异步回发中, 可能会更

新各个 UpdatePanel 控件的内容。更新将与面板设置、导致回发的元素以及特定于每个面板的代码有关。

因为也是返回某个城市的温度和天气的信息，故后台代码与传统的 WebForm 应用相同，两种方式的运行结果也相同，但运行效率大不一样，如图 7-6 所示。



图 7-6 示例运行结果

7.4 Silverlight4 概述

Silverlight 是设计、开发和发布有多媒体体验与富互联网应用程序 RIA (Rich Internet Application) 的网络交互程序，是提升互联网用户体验的一项重要 Web 技术。Silverlight 技术的出现将桌面丰富的用户界面体验带到了互联网。Silverlight 整合了一系列工具、技术和服务，使创建 RIA 的工作更加轻松，不再受限于浏览器所能实现的功能，而是可以实现新的 RIA 平台所支持的各种交互行为，是一种跨浏览器、跨平台的 .NET Framework 实现，用于为 Web 生成和提供下一代媒体体验和丰富的交互式应用程序。Silverlight 统一了服务器、Web 和桌面的功能，统一了托管代码和动态语言、声明性编程和传统编程以及 Windows Presentation Foundation (WPF) 的功能。它以浏览器的外挂元件方式提供 Web 应用程序中多媒体与高互动性前台应用程序的解决方案，同时它是微软用户体验策略中的一环，也是微软视图将美术设计和程序开发人员的工作明确切分与协同合作开发应用程序的尝试之一。

7.4.1 什么是 Silverlight

Silverlight 的原名叫 WPF/E, 它是微软推出的一个跨浏览器、跨平台的互联网应用程序开发技术, 具有极其优越的矢量图形、动画和多媒体支持的能力, 内置支持丰富的网络通信功能。Silverlight4 具有如下功能:

WPF 和 XAML。Silverlight 包含 WPF 的一个子集, 大大地扩展浏览器中用于创建 UI 的元素。

JavaScript 扩展。Silverlight 对 JavaScript 进行了扩展, 提供对 WebUI 更加强大的控制能力和与 WPF 元素协同工作的能力。

跨浏览器、跨平台支持。一个 Silverlight 程序可以在大多数的浏览器上运行, 这样我们开发 Silverlight 应用的时候就不用考虑它将运行在什么平台上。

可与现有应用程序集成。Silverlight 可无缝地与现有的 JavaScript 和 ASP.Net Ajax 代码集成, 并作为已创建功能的一个重要的补充。

采用 .NET 编程模型和相关的开发工具。我们可以采用托管的 JScript 和 IronPython 来编写 Silverlight 应用程序, 也可以使用 C# 和 Visual Basic 来完成。你还可以使用 Visual Studio 这样的工具来创建 Silverlight 应用程序。

LINQ。Silverlight 支持 LINQ, 它可让你在 .NET 中通过更加直观的和强类型的对象来访问数据。

目前, Silverlight 的稳定版本 Silverlight5.0 已经在 2011 年 12 月 9 日正式发布。它在兼容以前版本功能的前提下, 也加入了如支持 GPU 加速图像压缩、自带 3D graphics、自带远程控制、加强 Visual Studio 2010 的用户界面自动化测试、文字检错系统等新功能, 极大的方便了开发者的使用。

7.4.2 Silverlight4 开发环境部署

通常来说, Silverlight4 的开发环境需要包含以下几个开发工具:

1. Visual Studio 2010 或者 Visual Web Developer 2010 Express, 在安装过程中, .Net Framework 4 会同时被安装, 另外, 也可以单独下载 .NET Framework 4 安装;

2. 安装完开发工具后, 需要下载安装 Silverlight4 Tools for Visual Studio 2010 开发包。安装完成后, 在 Visual Studio 2010 中将会自动更新添加 Silverlight4 项目开发模板, 以及 Silverlight4 SDK 和相关开发环境。以下是该开发包中包含的内容:

Silverlight4 developer runtime

Silverlight4 SDK (software development kit)

Update for Visual Studio 2010 and Visual Web Developer Express 2010 (KB982218)

Silverlight4 Tools for Visual Studio 2010

WCF RIA Services V1.0 for Silverlight4

3. 针对 Silverlight 美工人员, 需要下载 Expression Studio 4, 因为只有 Blend 4 支持 Silverlight4 项目开发, 另外 Blend 4 和 Visual Studio 2010 是无缝结合, 使用 Blend 4 可以很轻松地创建和修改自定义控件样式, 开发专业的 Silverlight 项目界面。

4. 最后推荐安装 Silverlight4 Toolkit 控件包, 该控件是微软开发的 Silverlight 控件扩展

包,其中包含数十种扩展控件,可以方便 Silverlight 项目开发,另外该项目是开源项目,开发人员可以轻松地修改控件,创建自定义控件效果。

安装完成以上工具, Silverlight4 的开发环境已经创建完毕,如果是使用 Web Platform Installer,可以选择以上相关选项进行一键安装,就可以开发 Silverlight4 项目了。

另外微软已经推出 Silverlight4 简体中文正式版脱机帮助文档,该文档包含 Silverlight4 所有的技术知识,是开发人员必备资料。

7.4.3 认识 XAML

1. 什么是 XAML

在开发 Silverlight 应用程序时,可扩展应用程序标记语言 XAML(eXtensible Application Markup Language)具有举足轻重的地位。它是一种声明性语言,可以使用声明性 XAML 标记创建可见 UI 元素,然后可以使用单独的代码隐藏文件响应事件和操作使用 XAML 声明的对象。

XAML 本质上属于一种 .NET Programming Language,属于通用语言运行时(Common Language Runtime)。与 HTML 类似,特点是用来描述使用者接口。XAML 的语法格式为:<Application.../>, Application,是必备的基本元素。XAML 可以定义 2D 和 3D 物件、旋转、动画,以及各式各样的效果。如下面的示例代码,声明了一段最简单的 XAML:

```
1 <UserControl x:Class="SilverlightApplication1.MainPage"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6     mc:Ignorable="d"
7     d:DesignHeight="300" d:DesignWidth="400">
8     <Grid x:Name="LayoutRoot" Background="White">
9         </Grid>
    </UserControl>
```

这段代码中最外层是以 UserControl 开始,然后在 UserControl 中声明了一个 Grid 元素,仅此而已,其中最重要的一点就是命名控件的声明,如上面 2、3 两行代码。第一个声明将整个 Silverlight 命名空间映射为默认命名空间,第二个声明为 XAML 映射一个单独的 XML 命名空间,通常将它映射到 x: 前缀。这两个声明之间的关系是: XAML 是一个语言定义,而 Silverlight 是将 XAML 用作语言的一个实现,特别要指出的是, Silverlight 使用了 XAML 的一个严格子集。XAML 语言指定某些语言元素,其中的每个元素都可通过针对 XAML 命名空间执行 XAML 处理器实现来进行访问。

2. XAML 的基本使用

XAML 的一些基本使用包括在 XAML 中声明对象、为元素设置属性等。在 XAML 中声明对象可以直接使用对象元素语法,使用开始标记和结束标记将对象声明为 XML 元素,如下示例代码所示,在 Grid 中声明了一个矩形元素:

```
<Grid x:Name="LayoutRoot" Background="White">
    <Rectangle></Rectangle>
</Grid>
```

在 XAML 中为元素设置属性，有多种方式可供选择：

- 使用 XML 特性语法；
- 使用属性元素语法；
- 使用内容元素语法；

此方法列表并不表示可以使用这些方法中的任何一种来设置给定的属性，在 Silverlight 中某些元素的属性设置只支持其中一种，某些属性可能支持多种方式的属性设置方式。

使用 XML 特性语法为元素设置属性非常简单，如下面的示例代码：

```
<Rectangle Width=" 200" Height=" 100" Fill="OrangeRed"></Rectangle>
```

Silverlight 中的某些元素属性支持使用属性元素语法来设置属性，即在元素的属性中再指定另外一个子元素，如下面的示例代码：

```
<Rectangle Width=" 200" Height=" 100" >
    <Rectangle.Fill>
        <SolidColorBrush Color="OrangeRed"></SolidColorBrush>
    </Rectangle.Fill>
</Rectangle>
```

某些 Silverlight 元素提供的属性允许使用 XAML 语法时忽略该属性的名称，仅通过提供所属类型的对象元素标记中的一个值来设置该属性，称之为“内容元素语法”。如 TextBox 元素的 Text 属性，可以如下示例代码所示设置 Text 属性而无须指定 Text 属性的名称：

```
<TextBlock >
    欢迎进入 Silverlight 世界
</TextBlock>
```

当然对于该属性也可以按如下形式指定 Text 属性，效果是一样的：

```
<TextBlock Text="欢迎进入 Silverlight 世界"></TextBlock>
```

3. XAML 特性

除了上面介绍的 XAML 的基本使用之外，XAML 还有如下一些重要的特性：

- XAML 是以 XML 为基础的语法扩展；
- XAML 必须是格式良好的 XML；
- XAML 中的标记对应 .NET Framework 中的类型；
- XAML 具备面向对象及继承的特性；
- XAML 区分大小写；
- XAML 中能实现的，通过隐藏代码同样可以实现；
- XAML 中也会创建元素树。

7.4.4 创建一个 Silverlight 应用

后面的所有内容都围绕“如何使用 Silverlight4 构建一个简单的 Digg 客户端应用”展开。

首先创建一个新的 Silverlight 应用，选择 Visual Studio 2010 中的文件->新建->项目->Silverlight 应用程序。如图 7-7 所示：

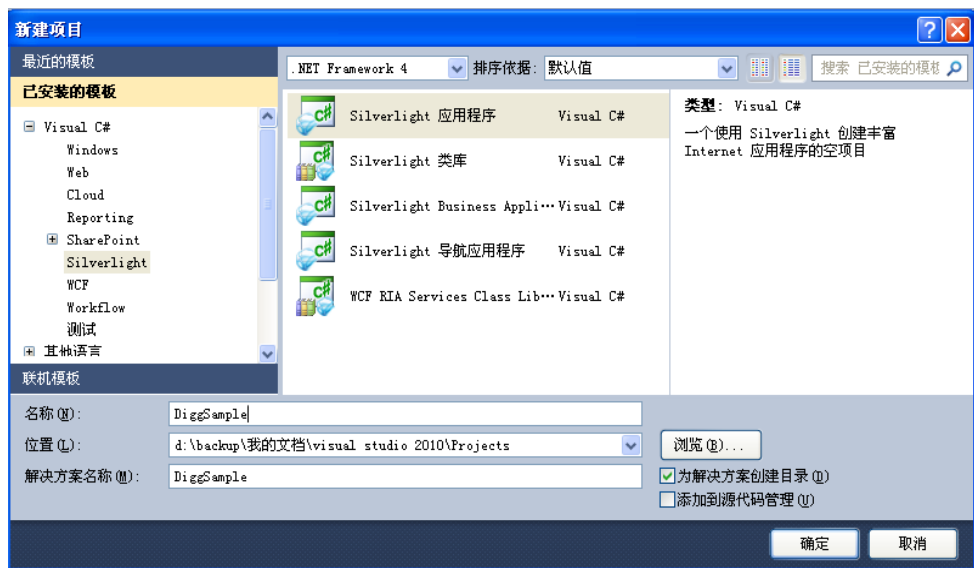


图 7-7 新建项目

我们将该项目命名为“DiggSample”。在点击确定按钮后，Visual Studio 会显示另外一个对话框，允许选择是否只创建一个 Silverlight 应用项目，或者还要加一个服务器端的 ASP.NET Web 项目到包含 Silverlight 应用的解决方案里去，如图 7-8 所示。

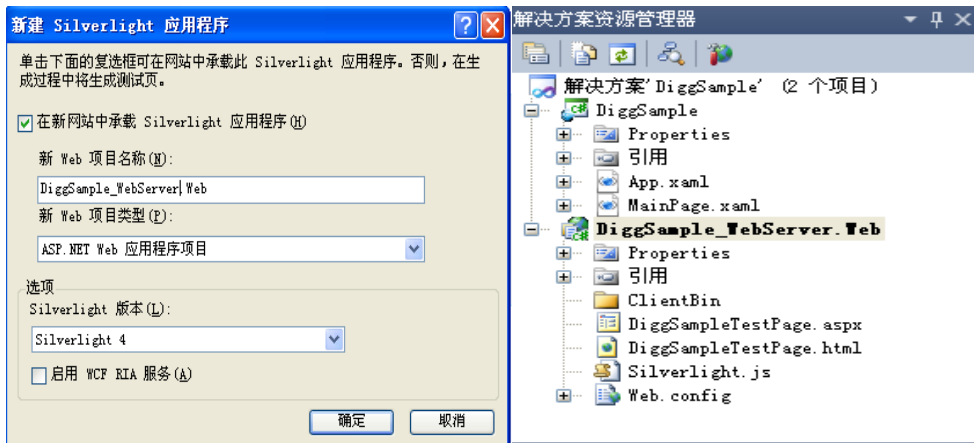


图 7-8 新建 Silverlight 应用程序

图 7-9 Silverlight 解决资源管理器

新 Web 项目命名为“DiggSample_WebServer”。在点击确定之后，Visual Studio 会为我们创建一个解决方案，里面包含一个 Silverlight 客户端应用和一个 ASP.NET Web 服务器端应用，如图 7-9 所示。Silverlight 应用可用于任何 Web 服务器（包括 Linux 上的 Apache），宿主于静态 HTML 文件或者任何服务器端生成的网页中。在这个 Digg 示例中，我们不会写任何服务器端的代码，而是将使用 Silverlight 的跨域 Networking 功能，来直接访问 Digg 服务的 API。选择创建一个 ASP.NET Web 服务器项目，主要是想获得自动的部署，并且使用它内置的 Web 服务器来做测试。

如果我们做一次编译的话, Visual Studio 会自动把编译好的 Silverlight 应用拷贝到我们的 Web 服务器项目中去, 不需要手工的步骤或配置。VS 为我们创建的默认的 Web 服务器项目包含一个 ASP.NET 网页和一个静态的 HTML 网页, 我们可以用来运行和测试其中的 Silverlight 应用。

7.4.5 理解 Silverlight 应用的内容

在默认情形下, 一个新建的 Silverlight 应用项目包含一个 MainPage.xaml 和一个 App.xaml 文件, 以及与它们相关的后台 (code behind) 类文件 (可以用 VB, C#, Ruby 或 Python 来编写)。

XAML 文件是 XML 文本文件, 可以用来用声明的方式指定 Silverlight 或 WPF 应用的用户界面。XAML 还可更广泛地用来用声明的方式代表 .NET 对象。

App.xaml 文件一般用来声明比如像画刷和样式对象这样可在整个应用中共享的资源。App.xaml 的后台 Application 类可用来处理应用级的事件, 像 Application_Startup, Application_Exit 和 Application_UnhandledException。

MainPage.xaml 文件在默认情形下是在应用激活时装载的起始的 UI 控件。在其中, 我们可以使用 UI 控件来定义用户界面, 然后在 Page 的后台代码类里处理它们的事件 (详见后文)。

在编译 DiggSample 项目时, 默认情形下, Visual Studio 会把代码和 XAML 标识编译进一个标准的 .NET 程序集文件中, 然后把它和任何静态资源 (图片或我们想要包含的静态文件) 包装进硬盘上一个叫做 “DiggSample.xap” 的文件中去, 如图 7-10 所示。

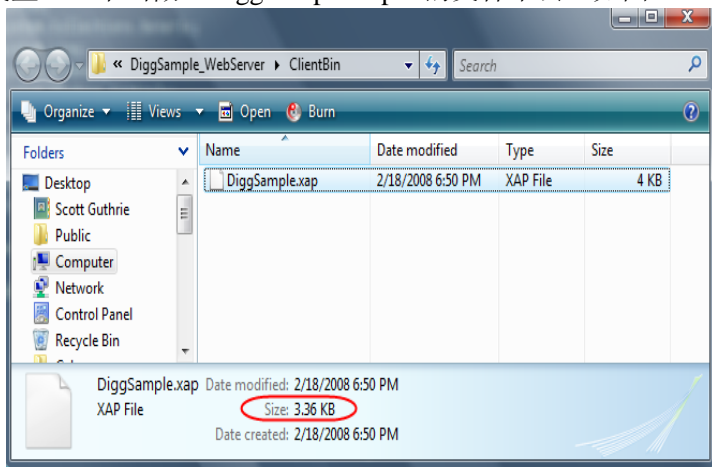


图 7-10 DiggSample.xap 文件

“.xap” 文件使用标准的 .zip 压缩算法来减小客户端下载的大小。一个 “hello world” Silverlight 应用 (用 VB 或 C# 编写的) 其大小大概为 4KB。

要宿主和运行一个 Silverlight4 应用, 可将 <object> 标签加到任何标准的 HTML 页面中 (不需要 JavaScript) 并将其指向 .xap 文件。Silverlight 就会自动下载这个 .xap 文件, 生成实例, 将其宿主于浏览器中的 HTML 网页中。它会跨浏览器 (Safari、FireFox、IE 等) 和跨平台 (Windows、Mac、Linux) 工作的。

HTML 和 ASP.NET 测试网页 (内含 <object> 标签, 其引用指向 Silverlight 应用) 是在

创建项目时自动添加的，只要点击 F5 编译，运行和测试就可以了。

7.4.6 学习如何添加控件和处理事件

目前的 Digg 应用什么都不做，在运行它时，只会调出一个空白的网页。可打开项目中的 MainPage.xaml 文件来改变它，往里面加些内容，代码如下：

```
<Grid x:Name="LayoutRoot" Background="White">
    <Button x:Name="myButton" Content="Push Me!" Width="100" Height="50">
    </Button>
</Grid>
```

当我们运行应用时，我们的按钮将会在网页的中间出现，内含“Push Me!”内容文字，如图 7-11 所示。

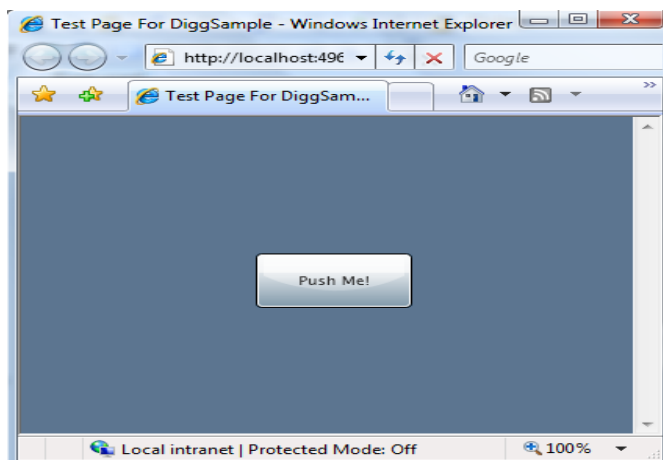


图 7-11 Button 按钮

要给按钮添加行为的话，可给它加一个“Click”事件处理函数。可在源码视图中通过输入事件的名称来做，代码如下：

```
<Grid x:Name="LayoutRoot" Background="White">
    <Button x:Name="myButton" Content="Push Me" Width="100" Height="50"
    Click="myButton_Click"></Button>
</Grid>
```

后台代码：

```
private void myButton_Click(object sender, RoutedEventArgs e)
{
    myButton.Content = "Pushed";
}
```

7.4.7 Silverlight 界面布局

Silverlight 和 WPF 都提供了非常强大和灵活的界面布局管理控件，能够让开发人员和设计人员轻松地对用户界面上的控件进行定位。该布局系统对显式指定坐标的控件支持固定的定位模型；除此之外，还支持一种更为动态的定位模型，控件和布局能随着浏览器的大小改变而自动改变其大小和方位。

在 Silverlight 和 WPF 中,开发者可以用布局面板来协调包含在其中的控件的位置和大小。Silverlight4 中内建的布局面板包括在 WPF 中最常用的 3 种: Canvas、StackPanel、Grid。

1. 使用 Canvas 绝对布局

Canvas 是从 Silverlight1.0 时代就有的一种基础布局面板,它支持对其中的控件采用绝对坐标定位。可使用附加属性对 Canvas 中的元素进行定位。通过附加属性指定控件相对于其直接父容器 Canvas 控件的上、下、左、右坐标的位置。附加属性很有用,因为它让父容器可以扩展其中包含的控件的属性集。Canvas 通过定义扩展属性 Top 和 Left,就能定义其中 Button (或其他任何 UI 元素)的 Left、Top,而不需要真正向 Button 类中添加这个属性,或修改 Button 类。

在下面的 XAML 中声明了两个 Button 控件,它们分别相对于父容器 Canvas 的左边距是 80,相对于父容器 Canvas 的上边距分别是 50 和 150。使用如下 XAML 语法即可完成(其中 Canvas.Top 和 Canvas.Left 都是附加属性的例子):

```
<Canvas Background="#FF5C7590">
    <Button Content="1" Height="50" Width="100" Canvas.Left="10" Canvas.Top=
"50"></Button>
    <Button Content="2" Height="50" Width="100" Canvas.Left="10" Canvas.Top=
"150"></Button>
</Canvas>
```

运行效果如图 7-12 所示。Canvas 适用于其中包含的 UI 元素比较固定的情形,但是如果你想向其中添加更多的控件,或者 UI 需要改变大小或能够移动,Canvas 显得不太灵活。这时,你不得不忙于手写代码来移动 Canvas 中的东西。应付这种动态的场景,更好的办法通常是使用其它带有相关功能的内建语义的布局面板,如 StackPanel 和 Grid。

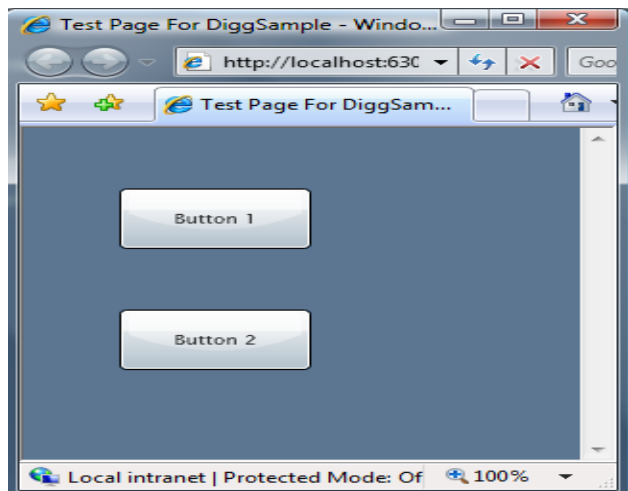


图 7-12 Canvas 面板

2. 使用 StackPanel 局部布局

StackPanel 面板布局，顾名思义就是“堆栈”布局，它是一种非常简单的布局面板，它支持用行或列的方式来定位其中的 UI 元素，可以很方便地对 UI 元素进行定位而无需设置其坐标。StackPanel 常用于布局页面上的一个很小的 UI 部分。

例如，可用下面的 XAML 标签在页面上垂直的排布 3 个按钮：

```
<StackPanel>
    <Button Width="200" Height="50" Margin="20" Content="button"></Button>
    <Button Width="200" Height="50" Margin="20" Content="button"></Button>
    <Button Width="200" Height="50" Margin="20" Content="button"></Button>
</StackPanel>
```

当然也可指定这些元素为水平排列，通过将 Orientation 属性设置为 Horizontal 来完成。如下面示例代码所示：

```
<StackPanel Orientation="Horizontal">
    <Button Width="200" Height="50" Margin="20" Content="button"></Button>
    <Button Width="200" Height="50" Margin="20" Content="button"></Button>
    <Button Width="200" Height="50" Margin="20" Content="button"></Button>
</StackPanel>
```

运行效果如下面 7-13 所示。

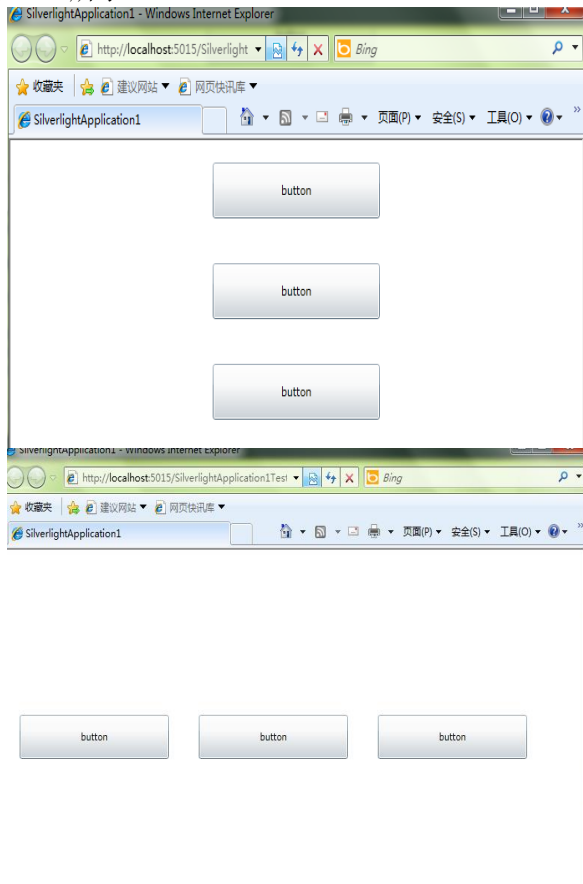


图 7-13 StackPanel 控件布局

7.4.8 使用 Grid 相对布局

Grid 控件是 Silverlight 中强大和灵活的布局面板，它支持用多行和多列的方式排布页面元素，非常类似于 HTML 里的 Table。不同的是，它不需将 UI 元素内嵌到单元格中，而是通过<Grid.RowDefinitions>和<Grid.ColumnDefinitions>属性来定义 Grid 的行和列。这两个属性要定义在<Grid>标签中，然后就可以使用 XAML 的附加属性语法指定 UI 元素属于哪一行、哪一列。

在下面这个例子中，将使用 Grid 进行一个用户登录界面的布局，为了使显示的效果明显，设置 ShowGridLines 属性为 true，在 Grid 的 Column 和 Row 的附加属性中，索引从 0 开始。如下面的代码所示：

```
<Grid x:Name="LayoutRoot" Background="#46461F" ShowGridLines="True">
    <Grid.RowDefinitions>
        <RowDefinition Height="120"/>
        <RowDefinition Height="*/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="100"/>
        <ColumnDefinition Width="*/>
    </Grid.ColumnDefinitions>
    <TextBlock Grid.Row="0" Grid.Column="0" Text="UserName:" VerticalAlign
ment="Center" Foreground="White"/>
    <TextBlock Grid.Row="1" Grid.Column="0" Text="Password:" VerticalAlign
ment="Center" Foreground="White"/>
    <TextBox Grid.Row="0" Grid.Column="1" Width="200" Height="30"
Horizontal Alignment="Left"/>
    <TextBox Grid.Row="1" Grid.Column="1" Width="200" Height="30"
Horizontal Alignment="Left"/>
</Grid>
```

运行效果如图 7-14 所示。

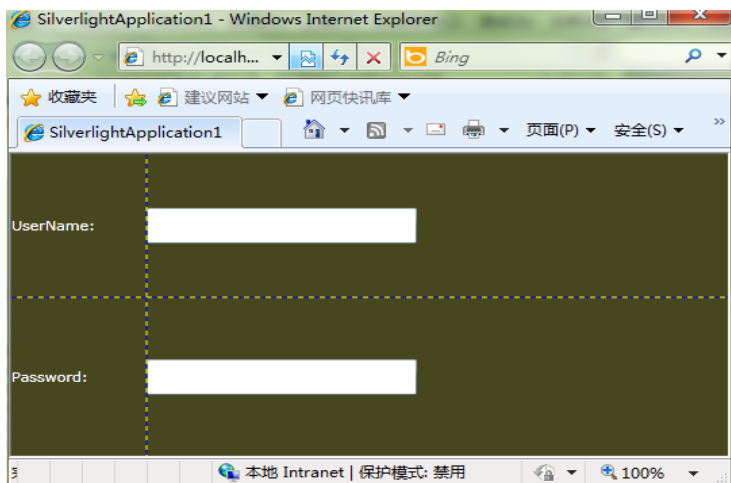


图 7-14 Grid 控件

可以看到,使用 Grid 可实现非常零碎和复杂的界面布局。除了像前面的示例那样指定具体的宽度和高度数值之外,还可设置 ColumnDefinition 和 RowDefinition 的高度和宽度为 Auto,这样将会根据置于其中的用户界面元素来自动调整 Grid 的高度和宽度,如下面的示例代码所示:

```
<Grid x:Name="LayoutRoot" Background="#46461F">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="Auto"/>
    </Grid.ColumnDefinitions>
    <TextBlock Grid.Row="0" Grid.Column="0" Text="UserName:" VerticalAlign
ment="Center" Foreground="White"/>
    <TextBlock Grid.Row="1" Grid.Column="0" Text="Password:" VerticalAlign
ment="Center" Foreground="White"/>
    <TextBox Grid.Row="0" Grid.Column="1" Width="200" Height="30"
Horizontal Alignment="Left"/>
    <TextBox Grid.Row="1" Grid.Column="1" Width="200" Height="30"
Horizontal Alignment="Left"/>
</Grid>
```

运行界面如下图 7-15 所示。

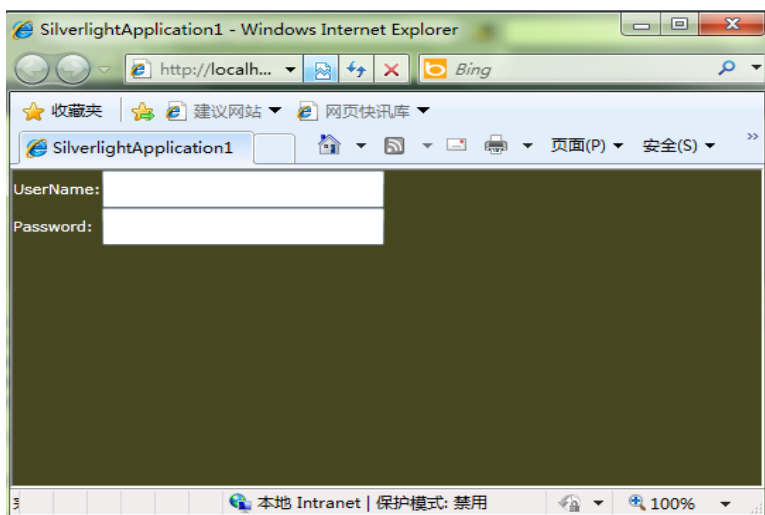


图 7-15 Grid 控件自适应边距

Grid 的 Row 和 ColumnDefinitions 还支持叫做“Proportional Sizing”（按比例缩放）的特性。用这个特性，可以让 Grid 的行列按相对比例的方式排放（可指定第二行的尺寸为第一行的 2 倍）。可以发现 Grid 提供了非常多的功能和灵活性，它是最常用的布局面板控件。

【例 7.4】用布局面板排布 Digg 页面。Digg 页面的最终界面如图 7-16 所示。

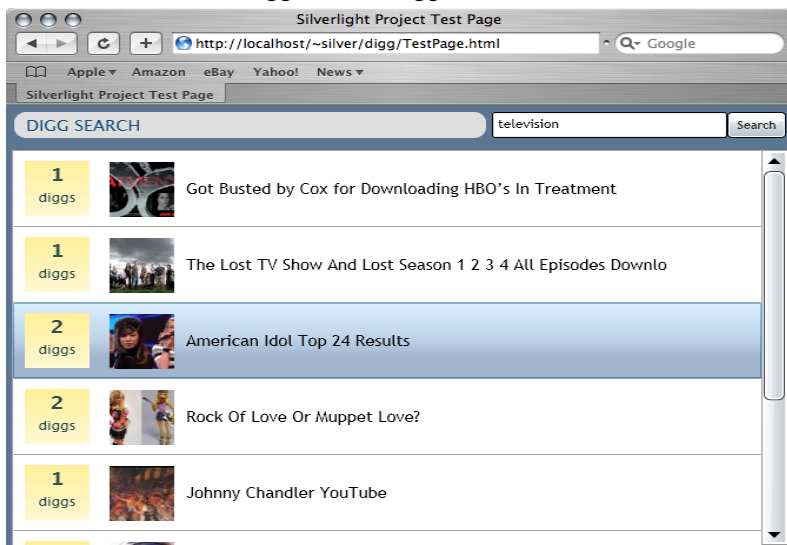


图 7-16 Digg 应用最终界面

要创建这种布局，首先添加一个其中包含两个 RowDefinition 的根级 Grid 面板。第一行的高度是 40px，而第二行则占据所有剩下的空间（Height="*"）：

```
<Grid x:Name="LayoutRoot" Background="#FF5C7590" ShowGridLines="True" Height="50">
    <Grid.RowDefinitions>
```

```

        <RowDefinition Height="40"></RowDefinition>
        <RowDefinition Height="*"></RowDefinition>
    </Grid.RowDefinitions>
</Grid>

```

运行界面如图 7-17 所示。

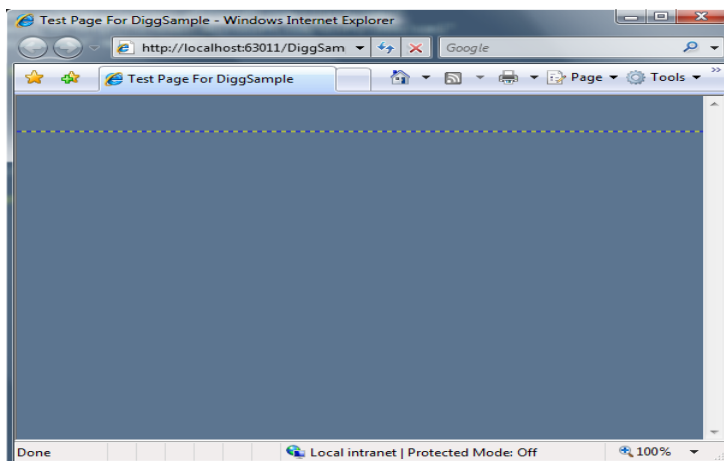


图 7-17 Grid 控件 RowDefinitions 属性

接下来，在刚才的根级 Grid 面板里，添加第二个 Grid 面板到第一行的位置，用它来排布页面顶部的行（页面头部）。在其中创建 3 列：分别容纳标题、搜索文本框和搜索按钮，代码如下：

```

<Grid x:Name="LayoutRoot" Background="#FF5C7590" ShowGridLines="True" Height="50">
    <Grid.RowDefinitions>
        <RowDefinition Height="40"></RowDefinition>
        <RowDefinition Height="*"></RowDefinition>
    </Grid.RowDefinitions>
    <Grid Grid.Row="0" Margin="7" ShowGridLines="True">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"></ColumnDefinition>
            <ColumnDefinition Width="200"></ColumnDefinition>
            <ColumnDefinition Width="40"></ColumnDefinition>
        </Grid.ColumnDefinitions>
    </Grid>
</Grid>

```

完成了这些后，就得到了 Digg 搜索页面的基本布局。接下来要做的是向其中添加控件。如果不用嵌套的 Grid，还可用一个 2 行 3 列的 Grid 来完成这个布局，配合使用 Grid 的 ColumnSpan/RowSpan 特性来合并多个列中的内容（和在 HTML table 中的做法类似）。

对头部的行，用内建的<Border>控件，可以设置 CornerRadius 属性将边框的各角改为圆角，并在其中添加一些文本来创建标题；用<WatermarkedTextBox>控件来创建第二列的搜索文本框；在第 3 列中放置一个搜索<Button>控件。然后在第二行放一些占位文字，稍

后在这里显示搜索结果。代码如下：

```
<Grid x:Name="LayoutRoot" Background="#FF5C7590" ShowGridLines="True"
Height="298" Width="400">
    <Grid.RowDefinitions >
        <RowDefinition Height="40"></RowDefinition>
        <RowDefinition Height="*"></RowDefinition>
    </Grid.RowDefinitions>
    <Grid Grid.Row="0" Margin="7" ShowGridLines="True" >
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"></ColumnDefinition>
            <ColumnDefinition Width="200"></ColumnDefinition>
            <ColumnDefinition Width="40"></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Border Grid.Column="0" CornerRadius="10" Background="#FFDEDEDE" Margin
="0,0,5,0">
            <TextBlock Text="DIGG Search" Margin="10,3,0,0" Foreground="#FF1451
7B"/>
        </Border>
        <my:WatermarkedTextBox x:Name="txtSearchTopic" FontSize="14"
Text="Topic..." Grid.Column="1"/>
        <Button Content="Search" x:Name="SearchBtn" Grid.Column="2"></Button>
    </Grid>
    <TextBlock Grid.Row="1" Margin="10" Foreground="White">
        Todo:Stories will display here...
    </TextBlock>
</Grid>
```

运行效果如图 7-18 所示。

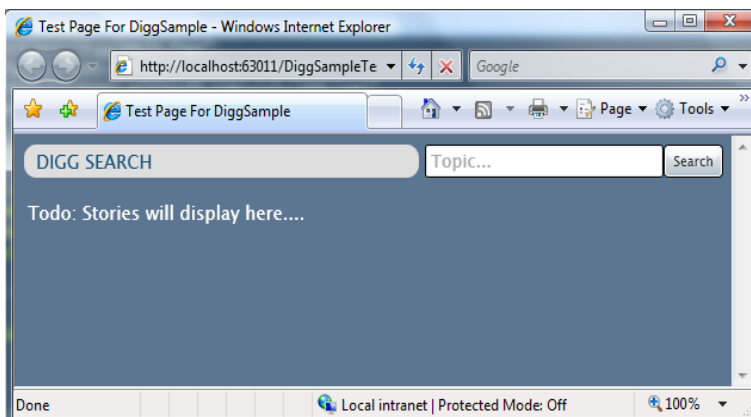


图 7-18 Digg 页面布局效果

限于篇幅，更多的 Silverlight 技术及应用可参阅相关书籍和网络资源。

思考练习题

1. 简述什么是 Ajax 技术，Ajax 的作用是什么。
2. 比较传统 Web 应用和基于 Ajax 框架的应用异同。
3. 简述什么是 Silverlight？它有什么特点。
4. 完成书上所示 Silverlight 实例，理解属性、事件、模板等概念。