Hot Cell Sample - Excel Web form

See What is Hot Cell Technology? for more information.

Introduction

HotCell Technology allows you to use client-side Excel workbooks to communicate with servers to perform tasks such as database updates. This demo downloads a workbook of employee information populated by ExcelTemplate. From within Excel, edit employee address information and update the server-side database.

Code

Workbook Generator Code

```
 //-------------------------------------------------------------
//--- SoftArtisans OfficeWriter HotCell Basic Form Update Sample
//---
//--- HotCells allow you to submit data from your Excel worksheet
//--- directly back to the database on the server.  This allows
//--- you to use Excel as a rich client web form.
//---
//--- In addition to this servlet code,
//--- look at the VBA code in the Excel template workbook.
//---
//--- This page displays the web form and generates the
//--- Employee Data workbook with ExcelTemplate
//---
//--- (c) 2009 SoftArtisans, Inc.
//--- Support: http://support.softartisans.com
//--- Sales:   sales@softartisans.com
//-------------------------------------------------------------

using System;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using SoftArtisans.OfficeWriter.ExcelWriter;

namespace SoftArtisans.OfficeWriter.HotCell.Samples
{
    /// <summary>
    /// Summary description for GetEmployeeDataSheet.
    /// </summary>
    public class GetEmployeeDataSheet : System.Web.UI.Page
    {
        private int employeeId;
        private string connString =
System.Configuration.ConfigurationManager.AppSettings["connString"];
        protected System.Web.UI.HtmlControls.HtmlSelect drpEmployee;
        protected System.Web.UI.HtmlControls.HtmlInputButton btnDataMarkers;

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

```
            this.employeeId = Convert.ToInt32(drpEmployee.Value);
            GenerateEmployeeDataSheet();
        }

        /// <summary>
        /// Get the employee data sheet HotCell workbook from which database updates
can be made
        /// </summary>
        private void GenerateEmployeeDataSheet()
        {

            /* Use ExcelTemplate to populate the employee data sheet HotCell template
            * and stream it to the client
            */

            DataTable EmployeeDt = GetEmployeeData();

            /* Create an instance of ExcelTemplate */
            ExcelTemplate xlt = new ExcelTemplate();

            /* Set PreserveStrings = true to make Excel treat
             * numeric strings as text
             */
            xlt.PreserveStrings = true;

            /* Open the template workbook */
            string templatePath = Page.MapPath("./BasicFormUpdateTemplate.xls");;
            xlt.Open(templatePath);

            /* Bind the Employee DataTable to the template
             * %%=Emp.*
             */
            DataBindingProperties bindingProperties =
xlt.CreateDataBindingProperties();
            xlt.BindData(EmployeeDt, "Emp", bindingProperties);

            /* Set the postURL in a hidden cell in the workbook
             * This is the URL to which the HotCell workbook should post
             * database updates.  Use this technique to be sure that the
             * HotCell workbook always posts back to the correct server
             */
            string fullUrl = Page.Request.Url.ToString();
            string postUrl =
                fullUrl.Substring(0, fullUrl.LastIndexOf("/")) +
"/DatabaseUpdate.asmx/UpdateDatabase";

            xlt.BindCellData(postUrl, "HotCellPostUrl", bindingProperties);


            /* Process the template to populate real data
             * into the data markers
             */
            xlt.Process();

            /* Stream the populated HotCell template to the client
             * response: Current instance of HttpServletResponse to stream the book
             * "EmployeeForm.xls": File name for the SaveAs dialog
             * false: do not open in browser plug in, open the workbook in Excel
             */
```

```
            xlt.Save(Page.Response, "EmployeeForm.xls", false);

        }

        /// <summary>
        /// Query the database for employee information
        /// </summary>
        /// <returns>DataTable of employee info for the selected employee ID</returns>
        private DataTable GetEmployeeData()
        {
            /* SQL to retrieve details for the selected employee */
            string sql = "SELECT HumanResources.Employee.EmployeeID, "
                + "Person.Contact.FirstName, Person.Contact.LastName, "
                + "Person.Contact.Title, Person.Address.AddressLine1 As Address, "
                + "Person.Address.City, Person.StateProvince.StateProvinceCode As
State, "
                + "Person.Address.PostalCode, Person.Contact.Phone FROM
HumanResources.Employee "
                + "INNER JOIN Person.Contact ON
Person.Contact.ContactID=HumanResources.Employee.ContactID "
                + "INNER JOIN HumanResources.EmployeeAddress "
                + "ON
HumanResources.EmployeeAddress.EmployeeID=HumanResources.Employee.EmployeeID "
                + "INNER JOIN Person.Address ON
Person.Address.AddressID=HumanResources.EmployeeAddress.AddressID "
                + "INNER JOIN Person.StateProvince ON
Person.StateProvince.StateProvinceID=Person.Address.StateProvinceID "
                + "WHERE HumanResources.Employee.EmployeeID = @employeeId";

            DataTable dt = new DataTable();
            using(SqlConnection conn = new SqlConnection(this.connString))
            {
                SqlCommand cmd = new SqlCommand(sql, conn);
                cmd.Parameters.AddWithValue("@employeeId", this.employeeId);
                SqlDataAdapter adpt = new SqlDataAdapter(cmd);
                adpt.Fill(dt);
            }
            return dt;
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            // Put user code to initialize the page here
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
```

```
        private void InitializeComponent()
        {
            this.btnDataMarkers.ServerClick += new
System.EventHandler(this.btnDataMarkers_ServerClick);
            this.Load += new System.EventHandler(this.Page_Load);

        }
```

```
        #endregion
    }
}
```

Database Update Webservice Code

```
 //------------------------------------------------------------
//--- SoftArtisans OfficeWriter HotCell Basic Form Update Sample
//---
//--- HotCells allow you to submit data from your Excel worksheet
//--- directly back to the database on the server.  This allows
//--- you to use Excel as a rich client web form.
//---
//--- In addition to this servlet code,
//--- look at the VBA code in the Excel template workbook.
//---
//--- This webservice is called by the HotCell workbook to update
//--- employee information in the database
//---
//--- (c) 2009 SoftArtisans, Inc.
//--- Support: http://support.softartisans.com
//--- Sales:   sales@softartisans.com
//------------------------------------------------------------

using System;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Web.Services;

namespace SoftArtisans.OfficeWriter.HotCell.Samples
{
    /// <summary>
    /// Updates a record in the Address table for a specific employee
    /// </summary>
    public class DatabaseUpdate : System.Web.Services.WebService
    {
        private string connString =
System.Configuration.ConfigurationManager.AppSettings["connString"];

        public DatabaseUpdate()
        {
            //CODEGEN: This call is required by the ASP.NET Web Services Designer
            InitializeComponent();
        }

        /// <summary>
        /// Update Address information for an employee.
        /// </summary>
        /// <param name="employeeId">ID of the employee to update</param>
        /// <param name="address">Updated address</param>
        /// <param name="city">Updated city</param>
        /// <param name="stateCode">Updated 2 character state code</param>
        /// <param name="postalCode">Updated zip</param>
        /// <param name="phone">Updated phone number</param>
```

```
        [WebMethod]
        public void UpdateDatabase(int employeeId, string address, string city, string
stateCode, string postalCode, string phone)
        {

            /*Validate state value against database*/
            string strInvalidState="";
            /*This SELECT query will verify that state code exists in database*/
            string strValidState = "SELECT StateProvinceID FROM Person.StateProvince "
+
                "WHERE StateProvinceCode = @stateCode";
            /*Create SqlConnection and SqlCommand to handle select query */
            SqlConnection ConnState = new SqlConnection(connString);
            SqlCommand IsStateCmd = new SqlCommand(strValidState, ConnState);
            IsStateCmd.Parameters.AddWithValue("@stateCode", stateCode);

            /* Open the connection and execute the query */
            ConnState.Open();
            SqlDataReader dr =
IsStateCmd.ExecuteReader(CommandBehavior.CloseConnection);

            /* No returned rows indicates that entered state does not exist in
database and invalid */
            if(!dr.HasRows)
               strInvalidState="invalid state entered";

            dr.Close();
            ConnState.Close();

            if(strInvalidState!="")
            {
                strInvalidState=strInvalidState;
                throw new Exception(strInvalidState);
            }


            /* Create two UPDATE queries to update the two tables in the database with
the changed values */
            string UpdateSQL = "UPDATE Person.Address SET AddressLine1 = @address,
City = @city, " +
                "StateProvinceId = (SELECT StateProvinceID FROM Person.StateProvince
WHERE StateProvinceCode = @stateCode), " +
                "PostalCode = @postalCode " +
                "WHERE AddressID = (SELECT AddressID FROM
HumanResources.EmployeeAddress WHERE EmployeeID = @employeeId)";
            string UpdatePhoneSQL = "UPDATE Person.Contact SET Phone = @phone " +
                "WHERE ContactID = (SELECT ContactID FROM HumanResources.Employee
WHERE EmployeeID = @employeeId)";

            using(SqlConnection Conn = new SqlConnection(connString))
            {
                /* Create an SqlCommand to handle the update query */
                SqlCommand UpdateCmd = new SqlCommand(UpdateSQL, Conn);

                /* The Phone field is in a different table, so we need to send a
second query */
                SqlCommand PhoneUpdateCmd = new SqlCommand(UpdatePhoneSQL, Conn);
```

```
                /* Add Parameters for every updateable value in the query */
                UpdateCmd.Parameters.AddWithValue("@address", address);
                UpdateCmd.Parameters.AddWithValue("@city", city);
                UpdateCmd.Parameters.AddWithValue("@stateCode", stateCode);
                UpdateCmd.Parameters.AddWithValue("@postalCode", postalCode);
                UpdateCmd.Parameters.AddWithValue("@employeeId", employeeId);

                PhoneUpdateCmd.Parameters.AddWithValue("@phone", phone);
                PhoneUpdateCmd.Parameters.AddWithValue("@employeeId", employeeId);

                /* Open the connection and execute the query */
                Conn.Open();
                int rowsAffected = UpdateCmd.ExecuteNonQuery();
                PhoneUpdateCmd.ExecuteNonQuery();
                Conn.Close();

                if(rowsAffected < 1)
                    throw new Exception("Rows updated was: " +
rowsAffected.ToString());

                return;
            }
        }

        #region Component Designer generated code

        //Required by the Web Services Designer
        private IContainer components = null;

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if(disposing && components != null)
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
```

```
        #endregion
    }
}
```

Downloads

Template: BasicFormUpdateTemplate.xls

Output: SampleOutput.xls