Hot Cell Sample - Excel Drill Down

See What is Hot Cell Technology? for more information.

Introduction

This sample demonstrates an application of HotCell Technology that actively downloads reports from a web server based on selections made on a client-side Excel spreadsheet.

This is a "Drill Down" report that allows you to drill through several layers of relational data. The application starts on a list of customers from the AdventureWorks database. Select a customer to "drill down" and see a list of all orders made for that customer. Select an order to see the order details for that particular order.

Each time you select a customer or order to drill down into, HotCell VBA code contacts a Servlet on the server and downloads the data you have requested. With this technique, you only download the data you need.

In addition to the Servlet code below, look at the VBA code in the Excel template workbooks.

Code

Servlet Code

```
 //-----------------------------------------------------------
//--- SoftArtisans OfficeWriter HotCell DrillDown Sample
//---
//--- This sample demonstrates an application of HotCell
//--- Technology that actively downloads reports from a web
//--- server based on selections made on a client-side Excel
//--- spreadsheet.
//---
//--- This is a "Drill Down" report that allows you to drill
//--- through several layers of relational data.  The application
//--- starts on a list of customers.  Select a customer to "drill down"
//--- and see a list of all orders made for that customer.  Select an
//--- order to see the order details for that particular order.
//---
//--- In addition to this servlet code,
//--- look at the VBA code in the Excel template workbooks.
//---
//--- (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 HotCellDrillDown.
    /// </summary>
    public class HotCellDrillDown : System.Web.UI.Page
    {
```

```
        private string connString =
            System.Configuration.ConfigurationManager.AppSettings["connString"];
        protected System.Web.UI.HtmlControls.HtmlInputButton btnDataMarkers;

        /// <summary>
        /// If the button is clicked, this page was requested from a WebForm
        /// so generate the customer list workbook
        /// </summary>
        private void btnDataMarkers_ServerClick(object sender, System.EventArgs e)
        {
            GetCustomers();
        }

        /// <summary>
        /// If Page_Load runs, check the "action" param to see if the
        /// request came from the HotCell workbooks.  Return the requested
        /// workbook with ExcelTemplate.
        /// </summary>
        protected void Page_Load(object sender, EventArgs e)
        {

            /* When the page is loaded, check to see
             * what kind of action to take
             */
            if(Page.Request.Form["action"] == "GetOrderDetails")
                GetOrderDetails(Page.Request.Form["id"]);
            else if(Page.Request.Form["action"] == "GetOrders")
                GetOrders(Page.Request.Form["id"]);

        }

        /// <summary>
        /// Generate a workbook with ExcelTemplate
        /// </summary>
        /// <param name="TemplatePath">Path to XLS template file</param>
        /// <param name="Dt">Data to import into the workbook</param>
        /// <param name="DataName">Data source name</param>
        /// <param name="OutputName">Name for browser's saveas dialog</param>
        private void GenerateSpreadsheet(string TemplatePath, DataTable Dt, string
DataName, string OutputName)
        {
            try
            {
                /* Create an instance of ExcelTemplate */
                ExcelTemplate xlt = new ExcelTemplate();

                /* Set PreserveStrings = true to force Excel
                 * to treat numeric values as strings
                 */
                xlt.PreserveStrings = true;

                /* Open the template workbook */
                xlt.Open(TemplatePath);

                /* Bind the template to the DataTable data source */
                DataBindingProperties bindingProperties =
xlt.CreateDataBindingProperties();
                xlt.BindData(Dt, DataName, 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 server on which
             * it was generated
             */
            string PostUrl = Request.Url.ToString();
            xlt.BindCellData(PostUrl, "HotCellPostUrl", bindingProperties);

            /* Process the template to populate it with values
             * from the data source
             */
            xlt.Process();

            /* Stream the template to the client */
            xlt.Save(Page.Response, OutputName, false);

            return;

        }
        catch(Exception ex)
        {
            /* Catch exceptions and write details */
            Page.Response.Clear();
            Page.Response.StatusCode = 500;
            Page.Response.Write("GenerateSpreadsheet failed. " + ex.ToString());
        }
    }

    /// <summary>
    /// Get Orders list workbook
    /// </summary>
    /// <param name="CustomerID">Customer ID</param>
    private void GetOrders(string CustomerID)
    {
        /* Get a DataTable of Orders for the selected customer */
        DataTable OrdersDt = GetOrdersDataTable(CustomerID);

        /* Call GenerateSpreadsheet to have ExcelTemplate
         * produce the Orders report
         */
        GenerateSpreadsheet(Page.MapPath("./OrdersDrillDownTemplate.xls"),
            OrdersDt,
            "Orders",
            "Orders.xls");
    }

    /// <summary>
    /// Get customer list worksheet
    /// </summary>
    private void GetCustomers()
    {
        /* Get a DataTable of Customer data */
        DataTable CustomersDt = GetCustomersDataTable();

        GenerateSpreadsheet(Page.MapPath("./CustomersDrillDownTemplate.xls"),
            CustomersDt,
```

```
            "Customers",
            "Customers.xls");
    }

    /// <summary>
    /// Get the Order Details workbook
    /// </summary>
    /// <param name="OrderId">Order ID</param>
    private void GetOrderDetails(string OrderId)
    {
        /* Get a DataTable of Order Details info */
        DataTable OrderDetailsDt = GetOrderDetailsDataTable(OrderId);

        GenerateSpreadsheet(Page.MapPath("./OrderDetailsDrillDownTemplate.xls"),
            OrderDetailsDt,
            "Details",
            "OrderDetails.xls");
    }

    /// <summary>
    /// Get OrderDetail data from the database
    /// </summary>
    /// <param name="OrderId">Order whose detail rows should be fetched</param>
    /// <returns>Order detail data</returns>
    private DataTable GetOrderDetailsDataTable(string OrderId)
    {
        string DetailsSQL = "SELECT Sales.SalesOrderDetail.SalesOrderID, " +
            "Production.Product.Name, Sales.SalesOrderDetail.UnitPrice, " +
            "Sales.SalesOrderDetail.OrderQty FROM Sales.SalesOrderDetail " +
            "JOIN Production.Product " +
            "ON Sales.SalesOrderDetail.ProductID=Production.Product.ProductID " +
            "WHERE Sales.SalesOrderDetail.SalesOrderID = @OrderID";
        SqlCommand Cmd = new SqlCommand(DetailsSQL);
        Cmd.Parameters.Add("@OrderID", OrderId);
        return GetDataTableFromCommand(Cmd);
    }

    /// <summary>
    /// Get Orders for the specified customer
    /// </summary>
    /// <param name="CustomerID">ID of the customer</param>
    /// <returns>Orders for the selected customer</returns>
    private DataTable GetOrdersDataTable(string CustomerID)
    {
        string OrdersSQL = "SELECT SalesOrderID, CustomerID, SalesPersonID, " +
            "OrderDate, SubTotal, TaxAmt, Freight, TotalDue " +
            "FROM Sales.SalesOrderHeader WHERE CustomerID = @CustomerID";
        SqlCommand Cmd = new SqlCommand(OrdersSQL);
        Cmd.Parameters.Add("@CustomerID", CustomerID);
        return GetDataTableFromCommand(Cmd);
    }

    /// <summary>
    /// Get Customer list
    /// </summary>
    /// <returns>Customer List</returns>
    private DataTable GetCustomersDataTable()
    {
        string CustomersSQL = "SELECT TOP 50 Sales.Customer.CustomerID, " +
```

```
                "Sales.Store.Name AS ContactName, Person.Address.AddressLine1, " +
                "Person.Address.City, Person.StateProvince.StateProvinceID AS Region,
" +
                "Person.Address.PostalCode, Person.CountryRegion.Name AS Country " +
                "FROM Person.Address INNER JOIN Sales.CustomerAddress " +
                "ON Person.Address.AddressID = Sales.CustomerAddress.AddressID " +
                "INNER JOIN Sales.Customer " +
                "ON Sales.Customer.CustomerID = Sales.CustomerAddress.CustomerID " +
                "INNER JOIN Person.StateProvince " +
                "ON Person.StateProvince.StateProvinceID =
Person.Address.StateProvinceID " +
                "INNER JOIN Person.CountryRegion " +
                "ON Person.CountryRegion.CountryRegionCode =
Person.StateProvince.CountryRegionCode " +
                "INNER JOIN Sales.Store ON Sales.Customer.CustomerID =
Sales.Store.CustomerID " +
                "WHERE (Person.CountryRegion.CountryRegionCode = 'US') " +
                "ORDER BY Sales.Customer.CustomerID";
            SqlCommand Cmd = new SqlCommand(CustomersSQL);
            return GetDataTableFromCommand(Cmd);
        }

        /// <summary>
        /// Execute a SqlCommand and get a DataTable
        /// </summary>
        /// <param name="Cmd">Command to execute</param>
        /// <returns>DataTable with the results of the command query</returns>
        private DataTable GetDataTableFromCommand(SqlCommand Cmd)
        {
            DataTable dt = new DataTable();

            //--- Create and initialize an SqlConnection
            using(SqlConnection Conn = new SqlConnection(this.connString))
            {
                //--- Link the SqlConnection to the SqlCommand
                Cmd.Connection = Conn;

                //--- Populate the DataTable
                SqlDataAdapter Adpt = new SqlDataAdapter(Cmd);
                Adpt.Fill(dt);
            }

            return dt;
        }

        #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


    }

}
```

Downloads

Templates: OrdersTemplate.xls, CustomersTemplate.xls, OrderDetailsTemplate.xls

Output: Customers.xls