

Programming SQL Server Reporting Services

Paul Litwin
Deep Training &
Fred Hutchinson Cancer Research Center

paul@deeptesting.com



Paul Litwin

- **Developer**
 - Focus: ASP.NET, ASP, VB, C#, SQL Server, ...
 - MCSD
 - Microsoft MVP
 - Programming Manger with Fred Hutchinson Cancer Research Center (Seattle)
- **Co-Founder and Senior Trainer**
 - Deep Training
 - www.deeptraining.com
- **Conference Chair/Speaker**
 - Chair, Microsoft ASP.NET Connections
 - Member INETA Speakers Bureau
- **Author**
 - Author/co-author of a dozen books, including...
 - *ASP.NET for Developers*
 - *Access Cookbook, 2nd edition*
 - *Access 2002 Desktop/Enterprise Dev Handbook*
 - *AppDev SQL Server 2005 Reporting Services Course*

Updated Slides & Samples Download

- You can download them from:
 - www.deeptraining.com/litwin

The Many Reporting Services APIs

- SQL Server Reporting Services provides a number of different application programming interfaces or APIs to programmatically manipulate, control, and extend it
 - URL Access
 - Report Viewer controls
 - Web Service
 - Custom Assemblies
 - Report Definition Language (RDL)
 - Reporting Services Extensions

SSRS 2008 Changes

- **SSRS 2008**
 - Completely rearchitected
 - Hosted outside of IIS
 - Word rendering extension, Excel, CSV improved
 - New tablix report item (combines table + matrix)
 - Data visualizations
 - Supports much larger reports
 - ...
- **How does programmability change?**
 - SSRS 2008 web service namespaces do not change
 - They still are ReportService2005 and ReportExecution2005
 - Some additional methods but changes are minor

Programming Reporting Services

- URL Access
- Report Viewer Control
- Calling Reporting Services Web Service
- Creating Custom Assemblies

Programming Reporting Services

URL Access

Using URL Access to Execute a Report

- The basic syntax for using URL access is

```
http://hostname/ReportServer?  
folder/report
```

Rendering a Report to an Output Format

- Using additional parameters on the URL query string, you can tell Reporting Services to directly render report in that format, bypassing the normal rendering to HTML 4.0 format
- This makes your reports work in ALL modern browsers (not just IE) !
- The basic syntax is

```
http://hostname/ReportServer?/  
folder/report&rs:Command=Render&  
rs:Format=render_format
```

Possible Render Formats

- SQL 2005
 - HTML4.0
 - HTML3.2
 - MHTML
 - IMAGE
 - EXCEL
 - CSV
 - PDF
 - XML
- New for SQL 2008
 - HTMLLOWC
 - WORD

URL Access – Running Reports from your App

- Example: RunSimpleReport.aspx

```
Dim strUrl As String =  
    lblReportServer.Text &  
    "?" & lblReportFolder.Text & "/" &  
    "rptEmployeeSales" &  
    "&rs:Command=Render&rs:Format=" &  
    lblFormat.Text
```

```
Response.Redirect(strUrl)
```

Executing a Report with Parameters

- If the report has parameters, you can take one of the following two approaches
 - Don't include the parameter values in URL; Reporting Services prompts user for parameters (works in IE only!)
 - Prompt the user for the parameters and then pass parameters along to Reporting Services as part of URL using the syntax like

```
http://hostname/ReportServer?/  
folder/report&parameter1=value1  
&parameter2=value2...
```

Parameterized Example

- Example: `RunParameterizedReport.aspx`

FYI: Rendering a Report to PDF w/ Custom Margins

- Many of the render formats have additional device information settings that are specific to the rendering format
- For example, you can set the margins of the rendered PDF document using the following settings
 - rs:MarginTop
 - rs:MarginBottom
 - rs:MarginLeft
 - rs:MarginRight
- You can change page orientation with
 - rs:PageHeight
 - rs:PageWidth

Programming Reporting Services

Report Viewer Control

Using the Report Viewer Controls

- SQL Server 2005/2008 Reporting Services comes with two Report Viewer controls:
 - ASP.NET Web Forms Report Viewer
 - Windows Forms Report Viewer

Server vs. Client Reports

- Each of the Report Viewer controls has the ability to work in two different modes:
 - **Server mode**: the Report Viewer control uses a Server report with the **.rdl** extension.
 - **Client mode**: the Report Viewer control uses a Client report with the **.rdlc** extension.

Using the Web Report Viewer Control with Server Reports

- The Web version of the Report Viewer control is a server-side ASP.NET control you can use on your ASP.NET 2.0/3.5 Web pages

Steps to using the Web Report Viewer Control with Server Reports

1. Drag the Report Viewer control to .aspx page
2. Visual Studio displays the ReportViewer Tasks panel. Select <Server Report> from the Choose Report dropdown
3. When you select <Server Report>, Visual Studio changes the look of the ReportViewer Tasks panel
4. Set the Report Server Url to point to the path of the Reporting Services server in the format <http://hostname/ReportServer>
5. Set the Report Path to the complete path to the report
6. Save the report and open it with Internet Explorer

Using the Windows Report Viewer Control with Server Reports

- In addition to Web Report Viewer control, Microsoft ships a Windows Report Viewer control with Reporting Services
- The Windows version of the control differs in a few places but the differences are minor
- One difference is that you have to explicitly call the RefreshReport() method of the Report Viewer control to tell it to render a report

Using the Report Viewer Control with Client Reports

- Visual Studio 2005/2008 includes a type of file for Windows & Web applications
 - the Report file
- This file can be used to create client reports that closely resemble reports you can create in a Reporting Services project
- Client reports use **.rdlc** extension

Using the Web Report Viewer Control with Client Reports

- Before you can use Report Viewer control to host a client report, you must create the report
- Differences between client and server reports
 - Client report designer lacks any tabs
 - Data for a client report lives outside of the report file elsewhere in the project inside of a typed DataSet
 - The client report requires a Report Viewer control in order to be rendered
 - A client report functions independently of SQL Server Reporting Services and does not require Reporting Services

Tip

You can convert server (.rdl) reports to and from client (.rdlc) reports. For details on how, open the Visual Studio Help Search window and search for "Converting RDL and RDLC Files".

Creating a Client Report

- **Basic steps**
 1. Create new **Report** in an ASP.NET app
 2. Click on **Add New Data Source** link and use wizard to create dataset
 3. Drag fields from Website Data Sources window to report
 4. Create Web page
 5. Drag the Report Viewer control onto page
 6. At the ReportViewer Tasks panel, select report
 7. Save and preview page
- **Example: WebClientViewer.aspx, rptProducts.rdlc**

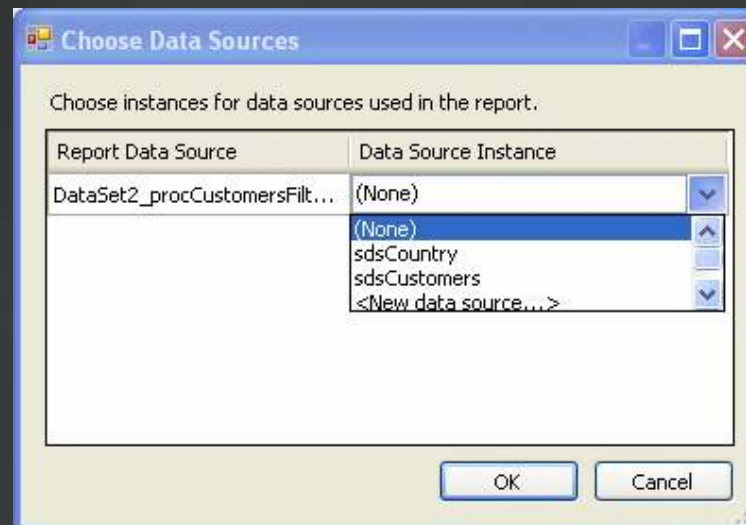
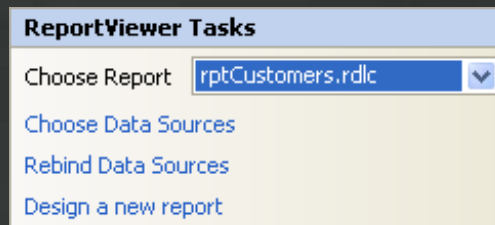
Example: Sending Filtered Dataset to Client Report (Great for Export to Excel)

- Example: FilteredReport.aspx
- Steps
 1. Start with GridView you are happy with
 2. Create client report that uses same underlying data source and fields
 3. Add ReportViewer control to GridView page; set Visible property to false

Example: Sending Filtered Dataset to Client Report (Great for Export to Excel)

- **Steps**

4. Use ReportViewer Tasks panel to select report and set data source to same data source as GridView



Example: Sending Filtered Dataset to Client Report (Great for Export to Excel)

- **Steps**

5. Add button and code to display / refresh report; for example

```
protected void cmdReport_Click(object sender, EventArgs e)
{
    Button cmd = (Button)sender;

    if (cmd.Text == "Send Rows to Report")
    {
        gvCustomers.Visible = false;
        rvCustomers.Visible = true;
        rvCustomers.LocalReport.Refresh();
        cmd.Text = "Return to Grid";
    }
    ...
}
```

Report Viewer Control **Caveat**

- The ASP.NET Report Viewer control may not work in some versions of browsers!
 - Works in IE
 - Recent versions of FireFox and Safari (at least on Windows) do seem to work

Programming Reporting Services

Reporting Services Web Service

Why Use the Web Service?

- Some of the more common tasks that you can accomplish using the Web Service endpoints
 - Get a list of reports on a Reporting Services server
 - Get a list of accounts authorized to run a report
 - Get a list of rendering methods supported by Reporting Services
 - Create a subscription to a report
 - Manage the authorization of users

Why Not Use the Web Service API?

- While it is powerful and handy at times, it is not always appropriate to use Web Service API to interact with Reporting Services
- For example, you can execute reports in Reporting Services using URL access
 - This is much easier to use than the report execution Web Service endpoint

SRS Web Service

- Two web services in SSRS 2005/2008
 - <http://server/ReportServer/ReportService2005.asmx>
 - <http://server/ReportServer/ReportExecution2005.asmx>
- Using web service from VS .NET Web project
 1. Set Web Reference to web service
 2. Add using/Imports statement
 3. Instantiate Web Service
 4. Set web service Credentials property
 - a) pass-thru login credentials, or
 - b) fixed account
 5. Call appropriate method of service

Call Web Service for List of Reports & Rendering Extensions

CallReportServiceCS.aspx and CallReportServiceVB.aspx

```
ReportingService rs = new ReportingService();
rs.Credentials = System.Net.CredentialCache.DefaultCredentials;
CatalogItem[] itmReports = rs.ListChildren(strFolder, false);

foreach(CatalogItem itm in itmReports)
{
    if (itm.Type == ItemTypeEnum.Report) {
        HyperLink hlk = new HyperLink();
        hlk.NavigateUrl = strReportsUrl + strFolder + "/" + itm.Name;
        if (itm.Description != null && itm.Description.Length>0) {
            hlk.Text = itm.Description; }
        else {
            hlk.Text = itm.Name;}

        lblLinks.Controls.Add(hlk);
        lblLinks.Controls.Add(new LiteralControl("<br>"));
    }
}
```

Call Report Service to Schedule Report

ReportSubscriptionCS.aspx

- Interesting parts of examples
 - Uses a shared schedule to create schedule
 - Much easier than constructing XML for single-use schedule
 - Shared schedule deleted after creation; but this is okay
 - Requires rights to be able to create shared schedule
 - Can use passed-through credentials or fixed user account
 - Encrypt appsettings section if using fixed account
 - See blog post
<http://aspadvice.com/blogs/plitwin/archive/2005/11/15/13830.aspx> for how to
 - Can be used to create one-time subscription and email report to users outside of network!

Call Rpt Service to Create Linked Reports

CreateLinkedReportsCS.aspx

- Linked Reports are great way to do row-level security
- Tedious to create using Report Manager, especially for large number of reports
- This example cranks out a bunch of linked reports with one click

Programming Reporting Services

Custom Assemblies

Custom Assemblies

- Why not just use Code window?
 - Reusability
 - Desire to code the solution in C# (or the *real* VB.NET)
 - Ability to do things that you can't do within the code window (e.g., database access, etc.)
- Possible Challenges
 - Reference needed
 - Security issues

Steps to Follow When Developing Custom Assembly (1 of 2)

1. Create assembly
2. Test assembly using ASP.NET, WinForm, or Console app
3. Reference assembly from report using References tab of Report Properties dialog
 - a) Call static methods using this syntax
`=namespace.class.method()`
 - b) Call instance methods using this syntax
`=Code.instance_name.method()`

`=AgeLib.AgeCalculator.Age(Fields!BirthDate.Value)`

Steps to Follow When Developing Custom Assembly (2 of 2)

5. Copy assembly to two folders on report server (your path may differ slightly)
 - a. C:\Program Files\Microsoft SQL Server\MSSQL.3\Reporting Services\ReportServer\bin
 - b. C:\Program Files\Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies
6. Run report in preview mode (if you need to change source code you will need to exit and restart VS.NET)
7. If Assembly uses protected resources (e.g., database, file system, you will have to deal with code access security
 - For more information, search help for “Using Custom Assemblies with Reports”.

Custom Assembly Example

- **AgeLib**
 - Class library project contains a single class file **AgeCalculator.cs** which contains **AgeCalculator** class and two overloads of static **Age** method
- **AgeCalculatorTest**
 - Console application used to test **AgeCalculator** class
- **Reports**
 - Reporting Services project contains **rptEmployeeAge** report and Northwind shared data source which calls **Age** method

Thank You!

- Please complete evaluation forms
- Contact: paul@deeptesting.com
- Download updated slides & samples from
 - www.deeptesting.com/litwin