

# SBI305: Programming SQL Server 2008 R2 Reporting Services

Paul Litwin

Fred Hutchinson Cancer Research Center /  
Deep Training

[paul@deeptesting.com](mailto:paul@deeptesting.com)

[twitter.com/plitwin](https://twitter.com/plitwin)

SQL Server  
CONNECTIONS

# Paul Litwin

- **Developer**

- Focus: ASP.NET, ASP, C#, SQL Server, Reporting Services
- MCSD
- Microsoft MVP
- Programmer Manager with Fred Hutchinson Cancer Research Center (Seattle)

- **Co-Founder and Senior Trainer**

- Deep Training
  - [www.deeptraining.com](http://www.deeptraining.com)

- **Conference Chair/Speaker**

- Chair, Microsoft ASP.NET Connections
- Member INETA Speakers Bureau

- **Author of over a dozen books & courses, including...**

- *AppDev SQL Server 2005 & 2008 Reporting Services Courses*
- *ASP.NET for Developers*
- *Access Cookbook*
- *Access 2002 Desktop/Enterprise Dev Handbook*

# Updated Slides & Samples Download

- You can download them from:
  - [www.deeptraining.com/litwin](http://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 – (see extra slides & samples)
  - Report Definition Language (RDL)
  - Reporting Services Extensions

# SSRS 2008/ 2008 R2 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
- **Not many changes regarding programming**

- **SSRS 2008 R2**

- Report Manager revamped
- Support for Geospatial data
- Shared data sets
- **New VS 2010 ReportViewer controls**
- **New ReportService2010 web service endpoint**
- More data visualizations (sparklines, data bars, indicators)
- Improvements to global collections and variables
- Many other changes

# 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 the report in that format, bypassing the normal rendering in the HTML 4.0 format
- The basic syntax is

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

# Render Formats

- HTML4.0
- MHTML
- PDF
- IMAGE
- EXCEL
- WORD
- CSV
- XML

# URL Access – Running Reports from your App

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

- **Examples**

Project	ASP.NET Platform	Example Notes
URLAccess	Web Form	RunSimpleReport, RunParameterizedReport
URLAccessMVC	MVC	Shows navigating to report using controller method and jQuery

# 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...
```

# FYI: Rendering a Report to PDF with 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

# ASP.NET Report Viewer Controls

Visual Studio Version	.NET Version	Remote Reports SSRS Version	Local Report Featureset	Web Ctrl AJAX Support?
VS 2005	2.0	SSRS 2005	SSRS 2005	
VS 2008 SP2	3.5	SSRS 2008	SSRS 2005	
VS 2010	4.0	SSRS 2008 R2	SSRS 2008 R2	X

# Web Viewer Controls

- ASP.NET Web Form control
- Windows Form control
- What about Silverlight?
  - No control, but...
  - <http://sushantp.wordpress.com/2009/11/25/silverlight-reporting-support-for-ssrs-reports-problem-and-possible-solutions/>
  - <http://www.perpetuumsoft.com/>

# Remote vs. Local Reports

- Each of the Report Viewer controls has the ability to work in two different modes:
  - **Remote (server) mode:** the Report Viewer control uses a deployed SSRS report with the **.rdl** extension
  - **Local (client) mode:** the Report Viewer control uses a local report created in Visual Studio with the **.rdlc** extension

# Using the Web Report Viewer Control with Remote Reports

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

# Steps to using the Web Report Viewer Control with Remote 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 browser

# 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 Report Viewer Control with Local Reports

- In Local Report Processing mode, report viewer is hooked up to local report
  - Local reports are created in Visual Studio
  - Use **.rdlc** extension
  - By default, data is supplied by ObjectDataSource control connected to typed DataSet
  - But datasource can be an instance of DataTable, a IEnumerable value (e.g., DataView or Array), or a IDataSource.

# 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

# Creating a Client Report

- **Basic steps**
  1. Create new **Report** in an ASP.NET app
  2. If the Data Sources window does not show, click on report surface and select **Data>Show Data Sources**
  3. Click on **Add New Data Source** link and use wizard to create dataset
  4. Drag fields from Website Data Sources window to report
  5. Create Web page
  6. Drag the Report Viewer control onto page
  7. At the ReportViewer Tasks panel, select report
  8. Save and preview page
- **Example: WebClientViewer.aspx, rptProducts.rdlc**

# Report Viewer Client-side Programming

- **Microsoft.Reporting.WebFormsClient.ReportViewer** members
  - exportReport()
  - find()
  - findNext()
  - invokePrintDialog()
  - recalculateLayout()
  - refreshReport()
  - documentMapCollapsed
  - isLoading
  - promptAreaClosed
  - reportAreaContentType
  - reportAreaScrollPosition
  - zoomLevel

# Report Viewer Programming Examples

- ChooseReport.aspx – demonstrates switching reports using server-side code
- AjaxViewer – demonstrates use of client-side code to alter zoom level of report

# 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

# SSRS Web Service Endpoints

SSRS Version	Native WS Endpoint	SharePoint WS Endpoint	Execution Endpoint
SSRS 2000	ReportService	n/a	ReportService
SSRS 2005	ReportService2005	ReportService2006	ReportExecution2005
SSRS 2008	ReportService2005	ReportService2006	ReportExecution2005
SSRS 2008 R2	ReportService2010	ReportService2010	ReportExecution2005

# SSRS ReportService2010

- What's changed?
  - This endpoint now works with both native and SharePoint integrated modes
  - Many enumerations have been dropped and been replaced with strings
    - `if (itm.Type == ItemTypeEnum.Report)` becomes `if (itm.TypeName == "Report")`
  - Some class names/methods have changed
    - `rs.CreateLinkedReport()` becomes `rs.CreateLinkedItem()`

# SSRS Web Service

- Two web services in SSRS 2010
  - <http://server/ReportServer/ReportService2010.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

# Web Service Examples

Project	Example	Example Notes
WebServiceClient	CallReportWebService	Simple example fills drop-down list with report names
	CreateLinkedReports	Illustrates creating linked reports from parameterized report
	CreateReportSubscription	Creates report subscription and schedules it for execution. Also lists all subscriptions for a report.

# Call Report Service to Schedule Report

## CreateReportSubscriptionCS.aspx

- Interesting parts of examples
  - Tedious process to create schedule and then subscription
  - 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  
*(Optional Topic)*

# 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 (.NET 3.5) 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...)
  - a. C:\Program Files\Microsoft SQL Server\MSRS10\_50.MSSQLSERVER\Reporting Services\ReportServer\bin
  - b. C:\Program Files (x86)\Microsoft Visual Studio 9.0\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@deeptraining.com](mailto:paul@deeptraining.com)
  - [twitter.com/plitwin](https://twitter.com/plitwin)
- Download updated slides & samples from
  - [www.deeptraining.com/litwin](http://www.deeptraining.com/litwin)

# Your Feedback is Important

Please fill out a session evaluation form  
drop it off at the conference registration  
desk.

Thank you!