CHAPTER 16

Creating Reports with SSRS

Just as we succeeded on the desktop, we will strive to succeed in services on the Web.

—Steve Ballmer

In the past three chapters of this book, we discussed creating reports with SQL code, MDX code, and Excel spreadsheets. Now we will take a look at Microsoft's premier reporting application, SQL Server Reporting Services (SSRS), also known as Reporting Services.

Reporting Services provides web-based reporting with its own web application or in combination with Microsoft SharePoint in a professional development structure. With Reporting Services, companies can have reports created by developers who understand the technology and data. Then the reports can be viewed by business analysts who have a deeper understanding of the company's business model.

This development structure also includes the ability to use source control to track changes in your reports and the ability to back up reports so that development hours are not lost in case of a disaster. These common features have been standard development tools for creating applications for more than a decade, but now the same tools can be applied to your reports.

In this chapter, we take a look at Reporting Services from the perspectives of report developers, administrators, and consumers. We show you how Reporting Services streamlines the interaction between all three of these roles. Afterward, you create a basic report so you can see how easy this is to accomplish.

Note It is odd, but Microsoft refers to SSRS as Report Server, Reporting Services, and sometimes Reporting Server to add flavor to the mix. Wherever possible, we try to use the term that is associated with the user interface we are discussing. Sometimes, however, these terms are used interchangeably within the same interface. In the end, it really does not matter if you call SSRS a server or a service, because it is both.

SSRS Architecture

SSRS is the most complex of SQL Server's business intelligence services. This is because there are many different components designed to work together to create a complete reporting solution. These various components can be spread across multiple computers to provide a high degree of scalability and performance. Figure 16-1

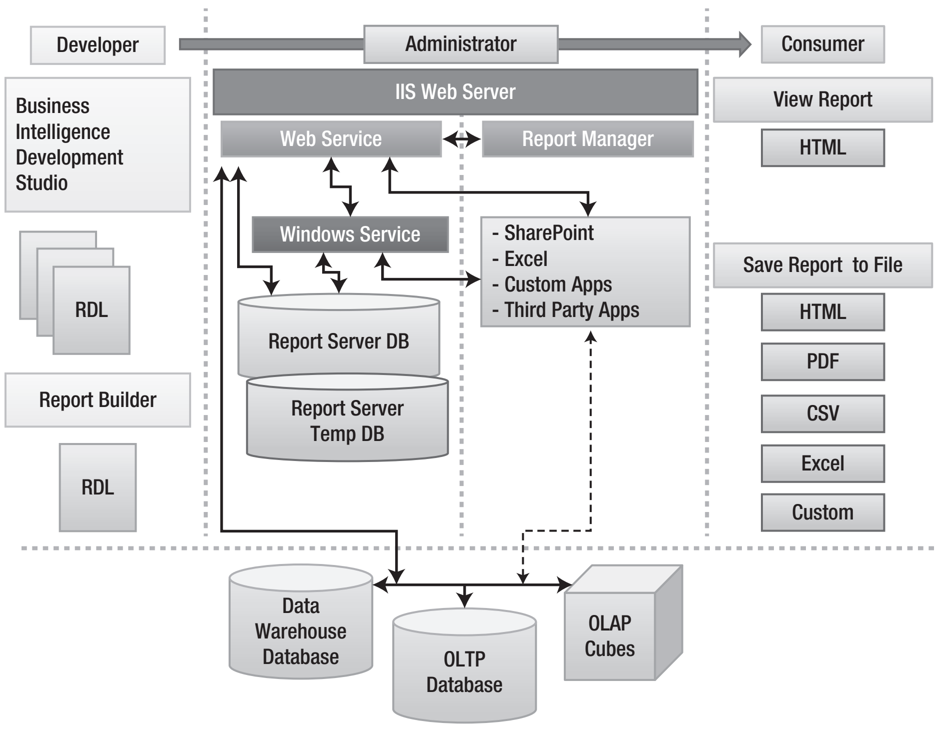


Figure 16-1. *The architecture of SSRS*

outlines the various components that make up Microsoft's Reporting Services. They are structured into three main categories:

- Developer tools
- Administration services
- Consumer-rendered reports

Developer Tools

Developers create reports using tools like Visual Studio or Microsoft's Report Builder. These development tools create XML files using the Report Definition Language (RDL) format. RDL files have been proposed as a standard for creating reports, and third-party applications can also be used to create RDL files.

After the RDL files are created, both BIDS and Report Builder can preview the report by rendering the XML code into a visual HTML output. Although it is not intended for end users to use HTML output, it does give a clear representation of what end users can expect to see when they browse the Reporting Services websites.

After the development is done, the RDL files are uploaded to the Reporting Services web service where it is then stored in a SQL Server database. When an end user requests a report using Reporting Services websites, the RDL file's code is read from the database and converted into a human-readable format. The default output is HTML, but the output format can also be in a PDF, CSV, Excel spreadsheet, or many custom formats.

Report Builder

The Report Builder application provides a simple way to create and edit RDL files. Report Builder is designed to be user-friendly, and its GUI interface is designed to have the look and feel of Microsoft Office so that developers accustomed to building reports in either Microsoft Access or Excel will feel right at home (Figure 16-2).

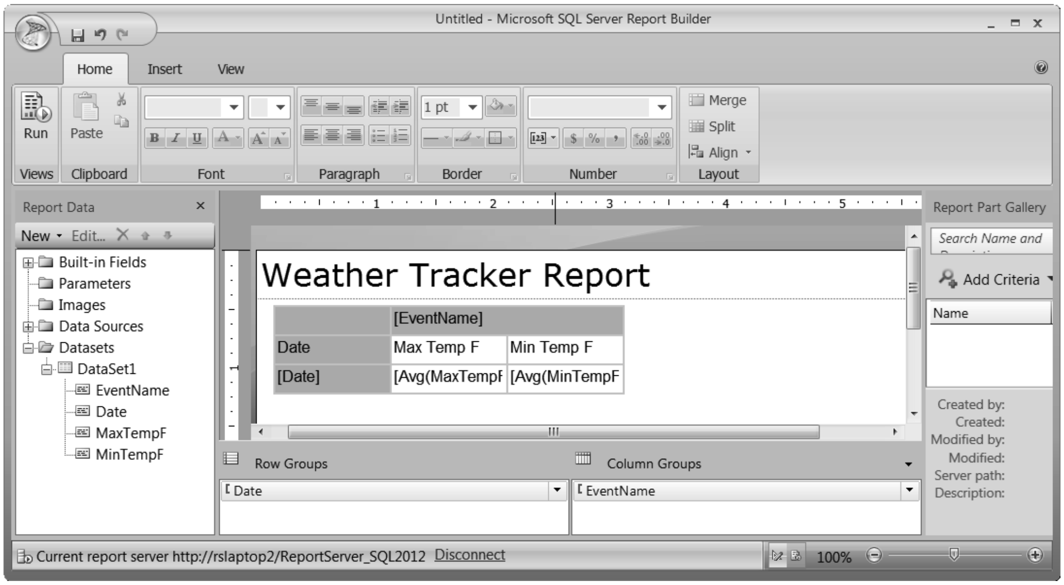


Figure 16-2. *The Report BuilderIU*

Unlike Visual Studio, Report Builder is not automatically installed when you install SQL Server 2012. It must be downloaded from the Internet and installed separately. The download and installation are both small and simple. Within a few minutes you have Report Builder installed, and you can start building new RDL files or edit existing ones.

Report Builder is designed for casual developers. It provides most, but not all, of the features that come with Visual Studio. For example, Report Builder is designed to work with only a single RDL file at a time, while Visual Studio can manage multiple files. Also, Report Builder is not a Visual Studio project type. The significance of this is that you cannot add Report Builder to your existing Visual Studio solution like we have with our SSIS and SSAS projects.

■ **Tip** Much of what we discuss in this chapter is the same information used to work with Report Builder. Therefore, as you work through this chapter, you are effectively learning both tools. For additional help, Microsoft offers several Report Builder video tutorials on the Internet.

Visual Studio

Visual Studio's Reporting Services project provides full-featured development. With it you can accomplish everything that can be done in Report Builder, plus you have the ability to manage multiple files concurrently, work directly with source control, and include the Reporting Services project in the same solution as your SSIS and SSAS projects.

Unlike Report Builder, Visual Studio is more utilitarian in design. This is partially because it provides additional functionality, but it is also because its intended audience is professional developers rather than casual developers.

The development tools are not difficult to use; they just take more effort to learn. It is similar to SSIS projects, in that you first drag and drop items from the Toolbox onto a design surface and then configure them (Figure 16-3).

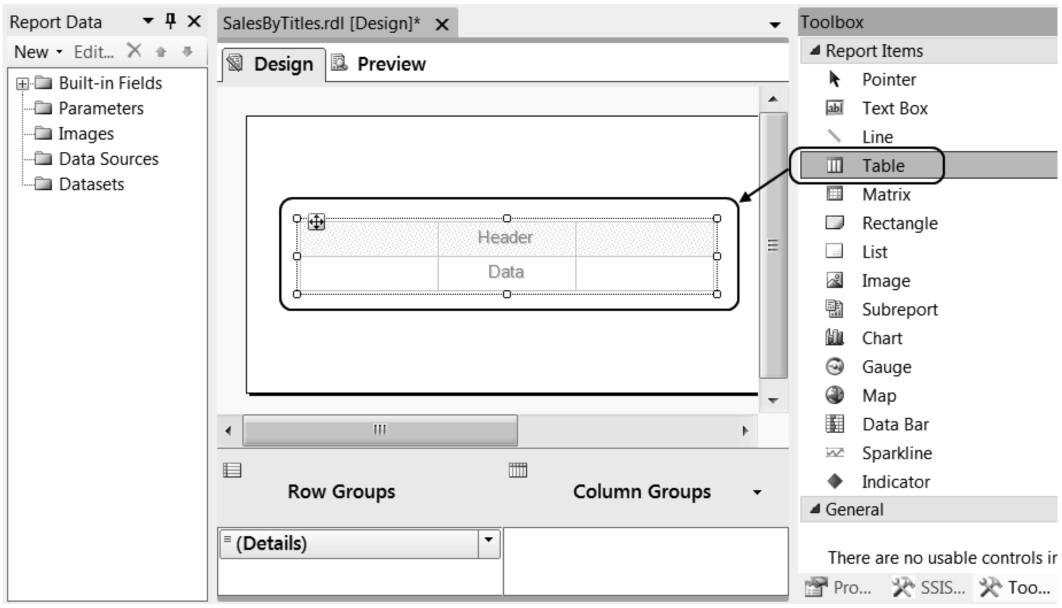


Figure 16-3. *The Visual Studio UI*

Much of this chapter, and the next, is devoted to explaining how to use these development tools.

The Administrative Services

After RDL files are developed and tested, they are uploaded to the Reporting Services web service, which in turn stores the RDL files in a Microsoft SQL Server database. When a report is browsed, it is automatically retrieved from the database by this same web service.

Access to the reports for human consumption is normally through an ASP.NET website that comes with SSRS, known as Report Manager, but the reports can also be accessed through your own custom applications. Note that applications do not access the reports directly from the Reporting Services database. Instead, they must indirectly access the reports through the SSRS web service, which serves as an abstraction layer. This adds complexity, but it also adds greater flexibility.

By using this design, you can place the web service (and its supporting Windows service) on one computer, while placing the Report Manager application or your own custom applications on a second computer. You can even span more computers by placing the Report Server databases on a separate computer. This can greatly increase your reporting performance at the cost of administrative overhead, but it is this scalability that sets Reporting Services apart from most reporting software.

The administrative services of SSRS are divided into report management features and end-user support (Figure 16-1). Report management features consist of the web service that manages the uploading and downloading of report files, report rendering, and exposing management functions or methods. The web service has an associated Windows service that it interacts with. The Windows service provides additional functionality such as scheduling components that allow reports to automatically be rendered and delivered to end users.

Although the Windows service does provide functionality similar to the web service, it is more difficult to interact with from a programming perspective. Thus, for programmers, the web service is the default way of interacting with SSRS's administrative services.

In addition to the web and Windows service, report management is handled by stored procedures within SSRS databases. These stored procedures are executed from the web and Windows service to perform all database activities.

SSRS Web Applications

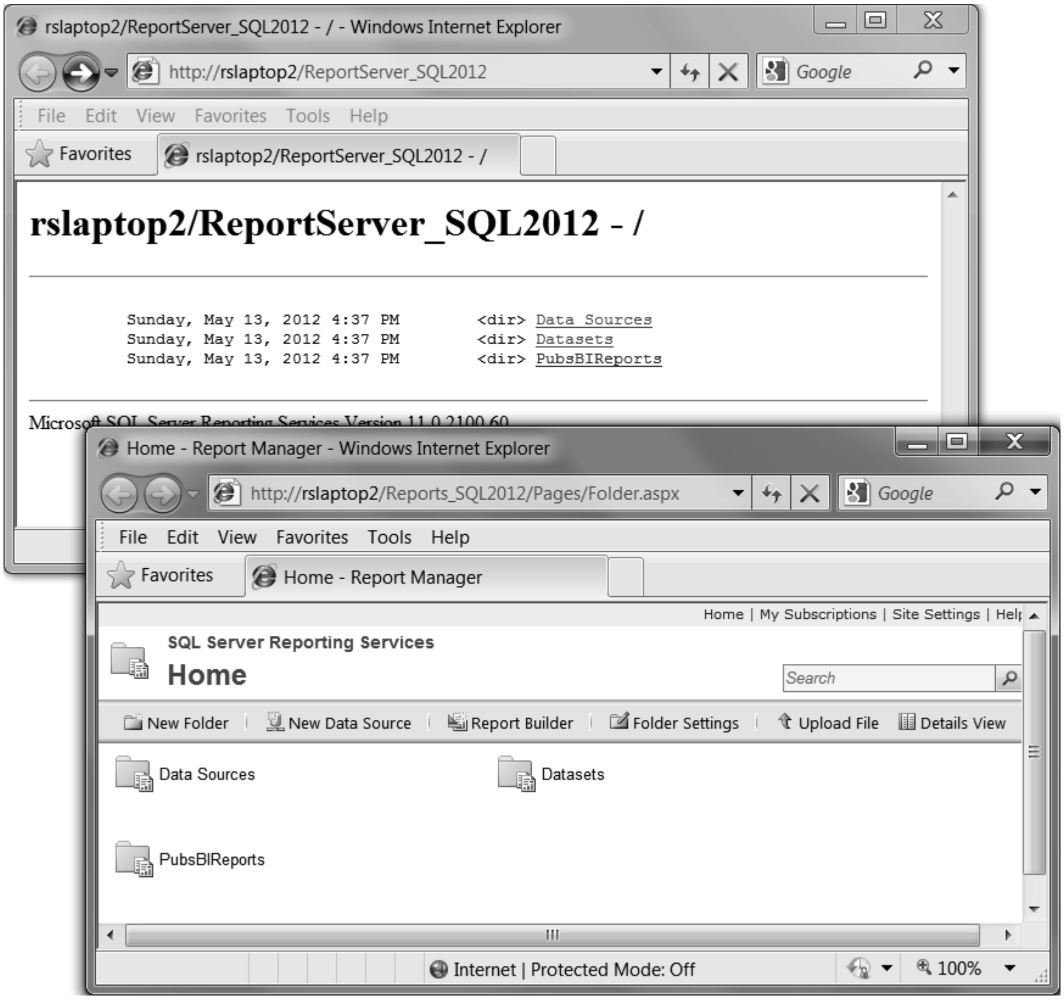
Each installation of Reporting Services includes two web applications, both of which are built using ASP.NET. The first is the web service we just mentioned, and the second is an end-user application for viewing and managing reports. This end-user application is known as Report Manager and is designed to interact with the web service.

Most companies find that the Report Manager web application works well for their needs. Because of this, these companies do not have a great need to create their own custom application. However, the web service allows other developers to create custom windowed, console, and web applications that can interact with the web service as well. This means that if you do not want to use Microsoft's ready-made web application—Report Manager—you can create your own by programming it to interact with the web service.

The SSRS web service is not designed for humans to use directly, but it can be accessed directly if you so desire. The web page that you see when you access the web service is quite basic when compared to the Report Manager. Both of these are shown for comparison in Figure 16-4. Notice that the web service contains text and hyperlinks but little else. Clicking a hyperlink navigates you to either a subfolder containing reports or launches a report for viewing. When you select a report for viewing, it will display the report in an HTML format.

The Report Manager web application provides a more user-friendly way of accessing reports, by using interactive menus and icons. It also supplies administration options for users with administrative privileges.

■ **Note** Although many features are accessible with any web browser, the Report Manager web application is designed to work with Internet Explorer, because it provides native support for Windows authentication. Expect to use Internet Explorer when performing administrative tasks using Report Manager.


rslaptop2/ReportServer_SQL2012 - / - Windows Internet Explorer
http://rslaptop2/ReportServer_SQL2012
File Edit View Favorites Tools Help
Favorites
rslaptop2/ReportServer_SQL2012 - /

rslaptop2/ReportServer_SQL2012 - /

```
Sunday, May 13, 2012 4:37 PM        <dir> Data Sources
Sunday, May 13, 2012 4:37 PM        <dir> Datasets
Sunday, May 13, 2012 4:37 PM        <dir> PubsBIReports
```

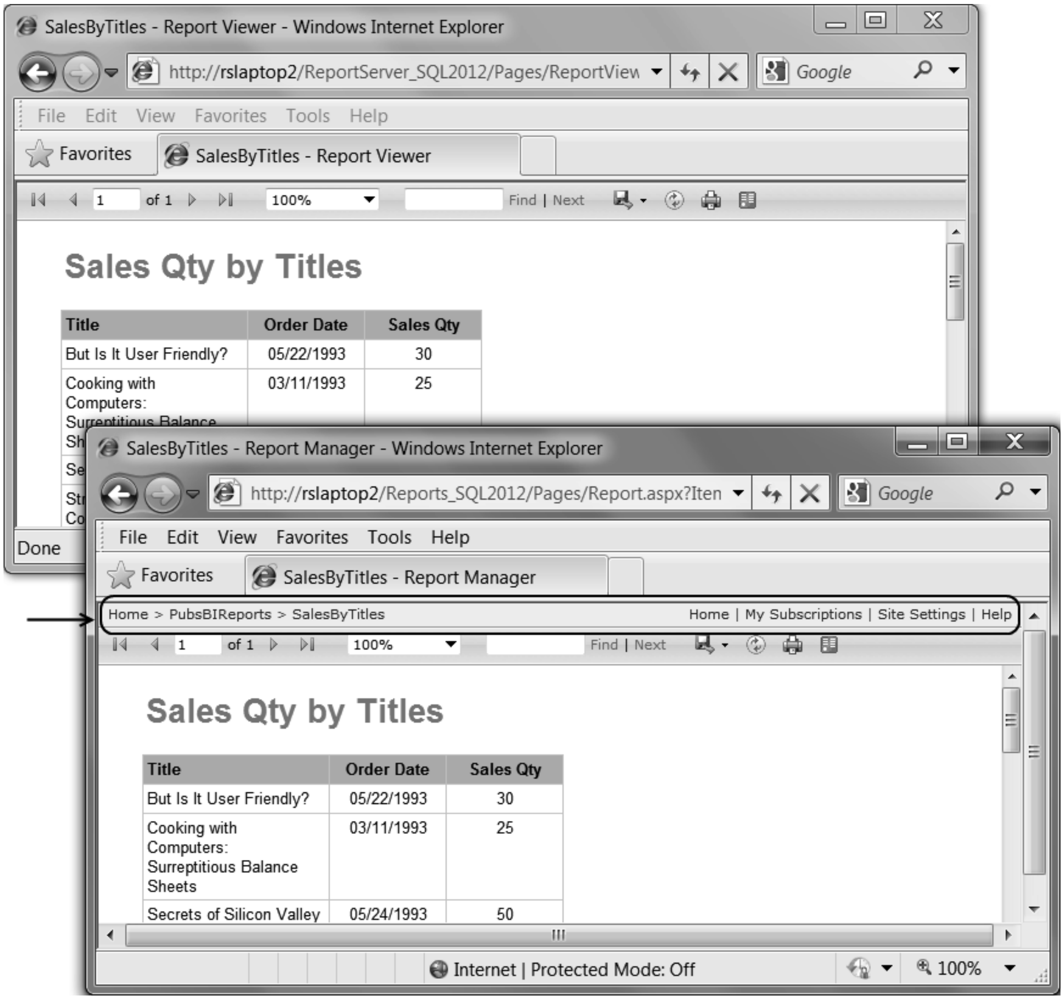
Microsoft SQL Server Reporting Services Version 11.0.2100.60

Home - Report Manager - Windows Internet Explorer
http://rslaptop2/Reports_SQL2012/Pages/Folder.aspx
File Edit View Favorites Tools Help
Favorites
Home - Report Manager
Home | My Subscriptions | Site Settings | Help
SQL Server Reporting Services
Home
Search
New Folder
New Data Source
Report Builder
Folder Settings
Upload File
Details View
Data Sources
Datasets
PubsBIReports
Internet | Protected Mode: Off
100%

Figure 16-4. *The SSRS web service and Report Manager user interfaces*

To provide users with the ability to view reports directly from the web service, Microsoft created a Report Viewer ASP.NET web page that launches when you click a report hyperlink listed on the web service's website. Whenever you launch a report using the Report Manager web application, the report displays this same ASP.NET page, but this time it is embedded within the Report Manager's web pages. For example, when you create a report action in an SSAS cube, as we did in Chapter 12, the report action must be configured to point to the web services and not the Report Manager web application.

When an end user clicks a report action, they will see an SSRS report displayed from the web service that looks similar to the one shown in the upper half of Figure 16-5, rather than the report in the lower half of Figure 16-5 (which is embedded into the Report Manager website). The difference is subtle, but if end users or other developers are not told about it, they can get confused as to why the reports appear different, depending on how they are accessed.


SalesByTitles - Report Viewer - Windows Internet Explorer
http://rslaptop2/ReportServer_SQL2012/Pages/ReportViewer
File Edit View Favorites Tools Help
Favorites
SalesByTitles - Report Viewer
1 of 1
100%
Find | Next
Done
SalesByTitles - Report Manager - Windows Internet Explorer
http://rslaptop2/Reports_SQL2012/Pages/Report.aspx?Iten
Google
SalesByTitles - Report Manager
Home > PubsBIReports > SalesByTitles
Home | My Subscriptions | Site Settings | Help
Internet | Protected Mode: Off
100%

Sales Qty by Titles

Title	Order Date	Sales Qty
But Is It User Friendly?	05/22/1993	30
Cooking with Computers: Surreptitious Balance Sheets	03/11/1993	25
Secrets of Silicon Valley	05/24/1993	50

Figure 16-5. *Report Viewer as shown from the web service and Report Manager*

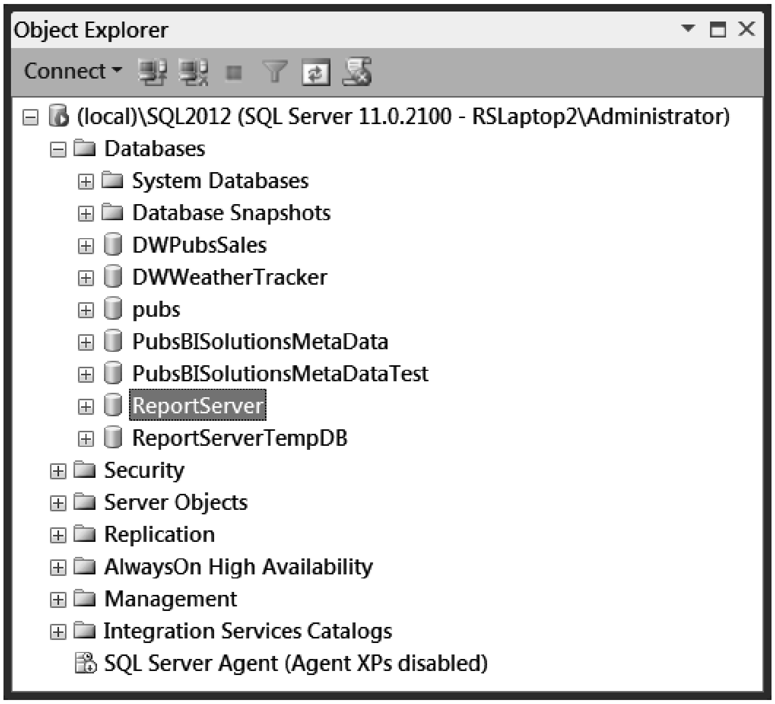
As you develop your reports, it is best to make sure you view the reports using both of these methods to ensure that what users are seeing is sufficient for their needs.

SSRS Services

Once you are done creating and testing a report, you can upload the file to the SSRS web service using Visual Studio, Report Builder, the Report Manager website, or even a SharePoint website. This variety of options is made possible because all of these applications can interact with the SSRS web service. This flexibility is one of the reasons why Microsoft designed Reporting Services to use a web service for the report processing.

In addition, the web service acts as a public interface for the SSRS Windows service, allowing Visual Studio, Report Builder, the SSRS Report Manager, SharePoint, and other software to interact with the SSRS windows service through an abstraction layer.

The web and Windows services' purpose is to process the programming instructions found in RDL files. The contents of the RDL files will be stored in a pair of SQL Server databases (Figure 16-6).



```
(local)\SQL2012 (SQL Server 11.0.2100 - RSLaptop2\Administrator)
  Databases
    System Databases
    Database Snapshots
    DWPubsSales
    DWWeatherTracker
    pubs
    PubsBISolutionsMetaData
    PubsBISolutionsMetaDataTest
    ReportServer
    ReportServerTempDB
  Security
  Server Objects
  Replication
  AlwaysOn High Availability
  Management
  Integration Services Catalogs
  SQL Server Agent (Agent XPs disabled)
```

Figure 16-6. *The Report Server databases*

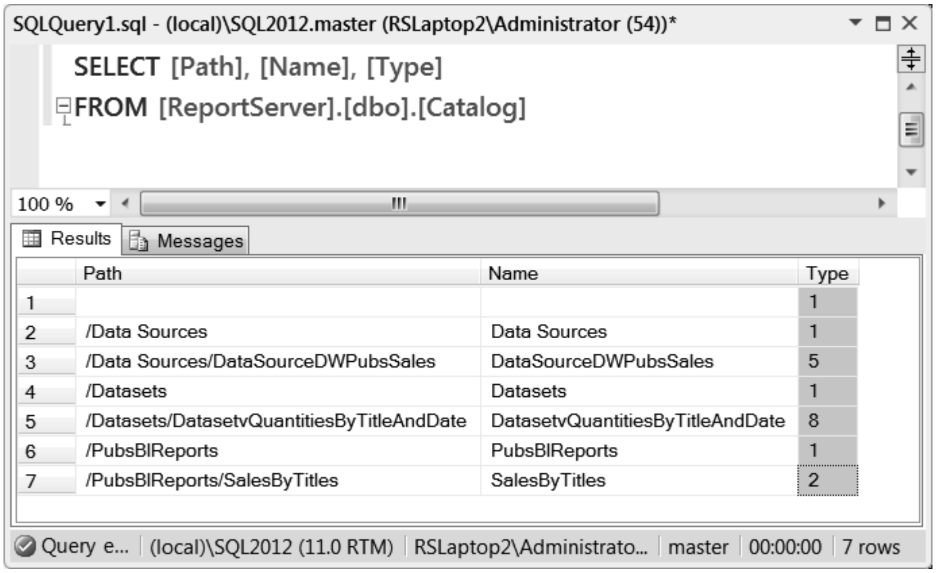
SSRS Databases

As shown in Figure 16-6, Microsoft installs two SQL Server databases with each installation of Reporting Services. The first database, normally named ReportServer, is designed to hold the RDL file code, metadata, and configurations. The second database, normally called ReportServerTempDB, is designed as a workspace for processing report requests. It also acts as a repository for cached reports.

■ **Tip** If you are using a named instance, the databases may have the instance name added to their standard names: ReportServer and ReportServerTempDB. Not to worry, though, the name variation does not change the way they work.

When a report is uploaded to the ReportServer database, the report's code is stored in a binary format, within a table named Catalog. If you select the data from the Catalog table, you will see a listing for every SSRS folder, data source, dataset, and report that has been created on the Reporting Server websites.

For example, in Figure 16-7, we have queried the Catalogs table, and seven objects are currently listed. The Path indicates the logical location that will be shown in the SSRS websites. The Name is, of course, the name we gave the objects when we created them. The Type is the type of SSRS object listed. Microsoft documentation on this table is sparse, but Table 16-1 describes the common type codes used.


SQLQuery1.sql - (local)\SQL2012.master (RSLaptop2\Administrator (54))*

```
SELECT [Path], [Name], [Type]
FROM [ReportServer].[dbo].[Catalog]
```

	Path	Name	Type
1			1
2	/Data Sources	Data Sources	1
3	/Data Sources/DataSourceDWPubsSales	DataSourceDWPubsSales	5
4	/Datasets	Datasets	1
5	/Datasets/DatasetvQuantitiesByTitleAndDate	DatasetvQuantitiesByTitleAndDate	8
6	/PubsBIReports	PubsBIReports	1
7	/PubsBIReports/SalesByTitles	SalesByTitles	2

Query e... | (local)\SQL2012 (11.0 RTM) | RSLaptop2\Administrato... | master | 00:00:00 | 7 rows

Figure 16-7. *Selected results from the Catalog table*

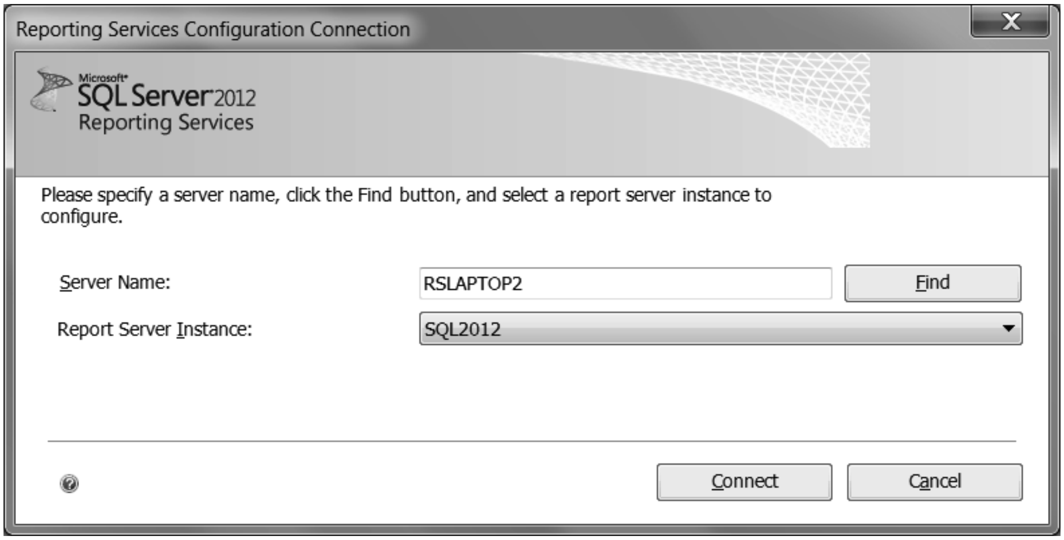
Table 16-1. *Types of Objects in the Catalog Table*

TypeId	Type Description
1	A logical folder
2	A report
5	A shared data source
8	A shared dataset

Report Server folders are used to organize your reports. And although they have no physical location on a hard drive, they do have a visual representation when using the Report Server websites. Report Server shared data sources are objects that hold connection information to be used by one or more reports. A Report Server shared dataset can also be used by one or more reports but holds query information, not connection information.

SSRS Configuration Manager

Because SSRS has so many components, it comes with its own configuration application. It can be accessed from the Windows Start menu by navigating to All Programs ➤ Microsoft SQL Server 2012 ➤ Configuration Tools ➤ Reporting Services Configuration Manager. Clicking this link launches the SSRS configuration application and presents you with a choice as to which server you would like to configure. In our examples, you connect to your own computers, but configurations can be managed on remote computers as well. Moreover, Reporting Services can be installed multiple times on a single computer using named instances. Therefore, you need to define which instance of Report Server you want to configure. Figure 16-8 shows the user interface that allows you to select both the server name and the instance name.



Reporting Services Configuration Connection

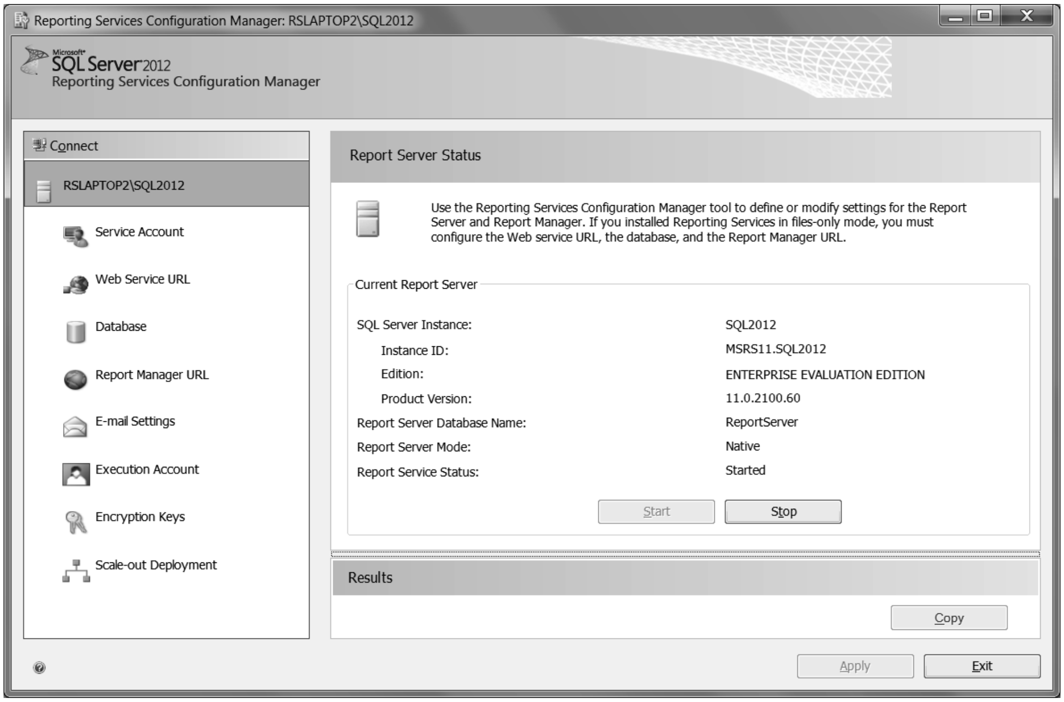
Please specify a server name, click the Find button, and select a report server instance to configure.

Server Name: RSLAPTOP2

Report Server Instance: SQL2012

Figure 16-8. *Connecting to a Report Server installation*

Once you have selected the Report Server installation, you will be presented with a user interface like the one in Figure 16-9. The user interface is designed with a treeview on the left side of the window and a detailed view of property settings on the right side. Items in the treeview are referred to as *pages*. Here you can define properties for the SSRS web applications and databases.



Reporting Services Configuration Manager: RSLAPTOP2\SQL2012

Report Server Status

Use the Reporting Services Configuration Manager tool to define or modify settings for the Report Server and Report Manager. If you installed Reporting Services in files-only mode, you must configure the Web service URL, the database, and the Report Manager URL.

Current Report Server

SQL Server Instance:	SQL2012
Instance ID:	MSRS11.SQL2012
Edition:	ENTERPRISE EVALUATION EDITION
Product Version:	11.0.2100.60
Report Server Database Name:	ReportServer
Report Server Mode:	Native
Report Service Status:	Started

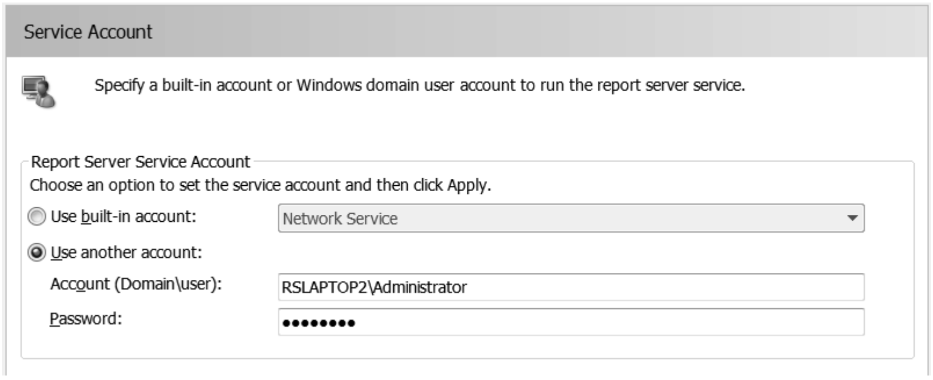
Figure 16-9. *The Report Server Configuration Manager user interface*

■ **Tip** As we have mentioned, Randal's computer, which is used for the screen shots in this book, uses a named instance of SQL Server called SQL2012. Consequently, the instance name used in Figure 16-8 and in Figure 16-9 matches his configuration. Most readers will use the SSRS default instance, but of course your configuration is dependent on the choices you made while installing SQL Server 2012.

The first page of the Reporting Services Configuration Manager is a general overview of the Report Server status. From here you can stop or start the Report Server and identify some of the startup parameters, such as whether it is running in native mode or configured to work with SharePoint. Native mode indicates that is using its own miniature web server to support the web service and the web application.

■ **Note** For earlier versions of Reporting Services, these websites were hosted on Microsoft's Internet Information Service (IIS) installation, meaning that you would have to first install IIS before you could install Reporting Services. All of the versions since 2005 provide their own web server support. Therefore, you do not have to install IIS before you can install Reporting Services. SharePoint, however, still requires a full IIS installation. Thus, when your SSRS server is configured to use SharePoint, it is utilizing the features associated with IIS and not its own native web server.

The Service Account page of the Reporting Services Configuration Manager allows you to define which account will be used when running the SSRS Windows service (Figure 16-10). The account's purpose is to interact with the SSRS databases, so if the account you choose does not have access to these databases, you will not be able to start the SSRS service. Ideally, you provide a Windows account that has limited permissions. But in a test environment, such as for the exercises in this book, an administrator account will work. It should be noted that it is not sufficient for the account to just be a Windows administrator; make sure that whatever account you use also has access to SQL Server and the SSRS databases.


Service Account

Specify a built-in account or Windows domain user account to run the report server service.

Report Server Service Account

Choose an option to set the service account and then click Apply.

Use built-in account: Network Service

[✓ Use another account:]

Account (Domain\user): RSLAPTOP2\Administrator

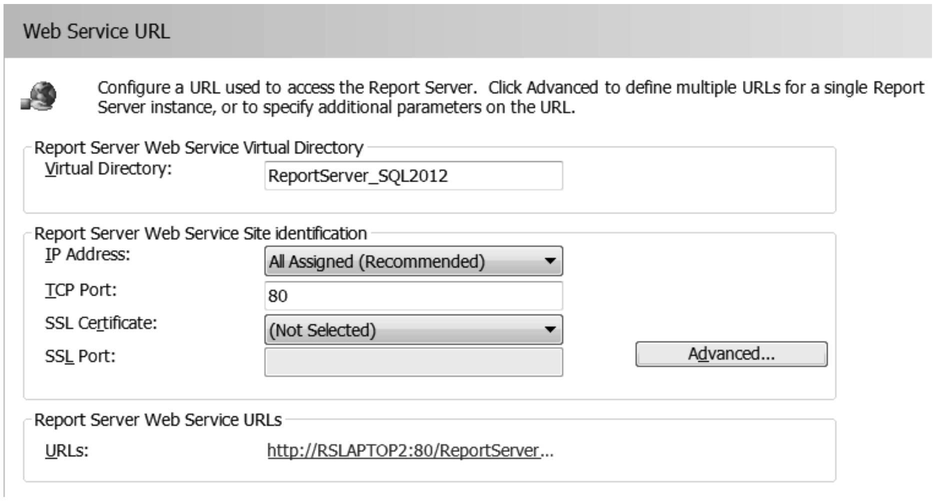
Password: ••••••••

Figure 16-10. *The service account page of the Configuration Manager*

The account that you choose will also be used for more than just connecting to SQL Server databases. Depending on how you set up SSRS, you can also interact with mail servers, network shares, and domain controllers. This means that if you are planning to use features that would involve those servers, you must also expect to configure the Windows account to have access to these items.

■ **Tip** We use the Windows administrator account to avoid permission issues, but of course this is not recommended practice for production, because it represents a security risk. In this book, we have been using an administrator account that provides access to SQL. In production, it is a very common issue to be unable to gain access to the Windows account. Microsoft's website provides a considerable amount of advice to aid you in configuring the Reporting Services service account.

The web service URL page is helpful in identifying whether Reporting Services is working correctly (Figure 16-11). A hyperlink on this page launches your browser and tries to navigate to the Reporting Services web service. At first, the web service will be unresponsive. It takes a few minutes to come alive, so assume that first request may take a minute or so before the web page will open successfully. This will happen every time you reboot your computer.


Web Service URL

Configure a URL used to access the Report Server. Click Advanced to define multiple URLs for a single Report Server instance, or to specify additional parameters on the URL.

Report Server Web Service Virtual Directory
Virtual Directory: ReportServer_SQL2012

Report Server Web Service Site identification
IP Address: All Assigned (Recommended)
TCP Port: 80
SSL Certificate: (Not Selected)
SSL Port:

Report Server Web Service URLs
URLs: http://RSLAPTOP2:80/ReportServer...

Figure 16-11. *The web Service URL page of the Configuration Manager*

The Report Manager website URL can also be used to identify whether Reporting Services is working correctly. If you have previously opened the web service, the Report Manager web application should start up rather quickly. If you go directly to the Report Manager application without accessing the web service first, it will take a few minutes for it to open, because, as we just stated, it must start the web service application before it can display its content. This is perfectly normal and will not affect your end user's system.

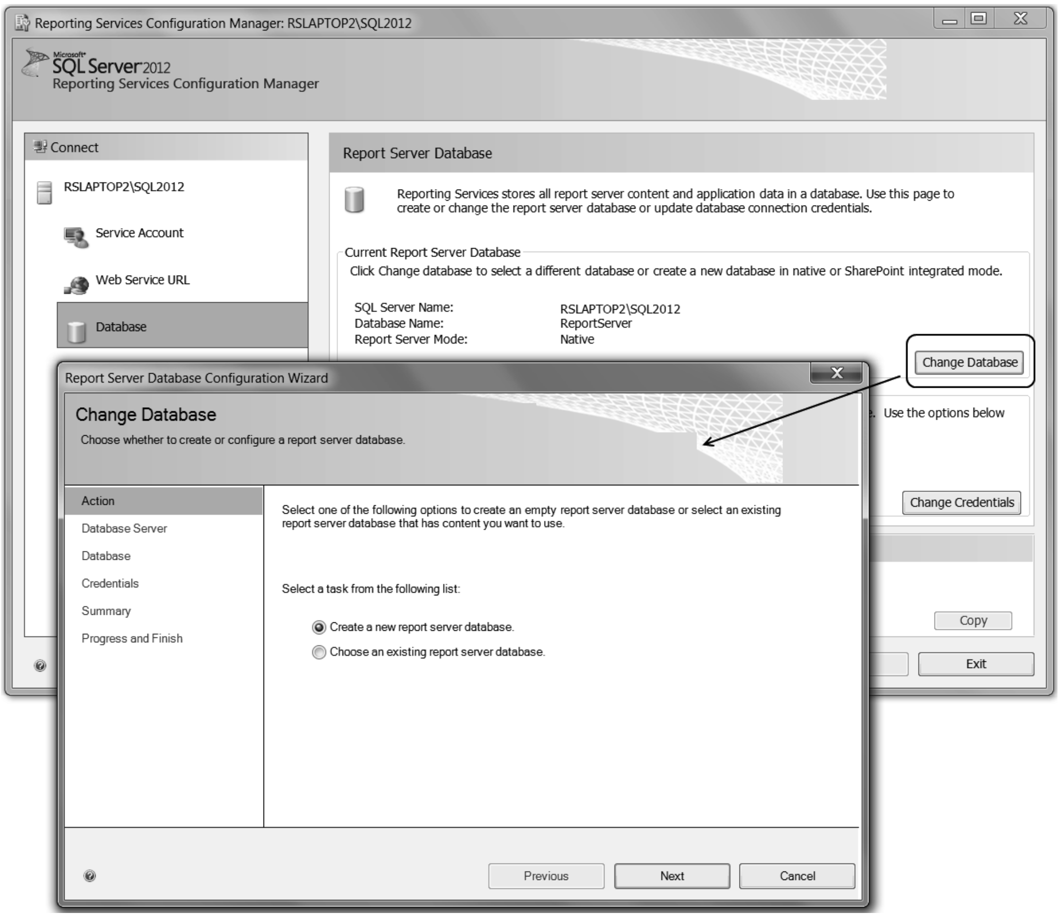
When you first install Reporting Services, it will not include any reports. So, when the websites are accessed, the web pages will appear with a header at the top and a mostly blank area beneath that. We discuss how to change that next.

WHEN THINGS GO WRONG

Even if you cannot get the web and Windows services running correctly, you can still create and test your SSRS reports for the rest of the book. The problems come when you try to follow along with some of the administration tasks outlined at the end of the chapter. If you have not managed to get the Reporting Services running correctly by then, you will only be able to read about how the administration tasks are performed, which won't be as much fun, although it will still be informative. We prefer that you follow along throughout the chapter, however, so here are some tips for troubleshooting SSRS installations.

The Databases

Normally, the SSRS databases are created during the installation of SQL Server, but occasionally they are not. You need the database if you want to perform SSRS administration. You can use SQL Server Management Studio to verify that the Report Server databases are installed. If you do not see them there, you can assume you found the problem. To fix this issue, use the database page of the Reporting Server Configuration Manager to manually add the databases. Microsoft provides a wizard that easily walks you through the process, which is accessed by clicking the Change Database button shown in Figure 16-12.


Reporting Services Configuration Manager: RSLAPTOP2\SQL2012

Report Server Database

Reporting Services stores all report server content and application data in a database. Use this page to create or change the report server database or update database connection credentials.

Current Report Server Database

Click Change database to select a different database or create a new database in native or SharePoint integrated mode.

SQL Server Name: RSLAPTOP2\SQL2012
Database Name: ReportServer
Report Server Mode: Native

Report Server Database Configuration Wizard

Change Database

Choose whether to create or configure a report server database.

Select one of the following options to create an empty report server database or select an existing report server database that has content you want to use.

Select a task from the following list:

[✓ Create a new report server database.]
Choose an existing report server database.

Figure 16-12. *Installing the Report Server databases*

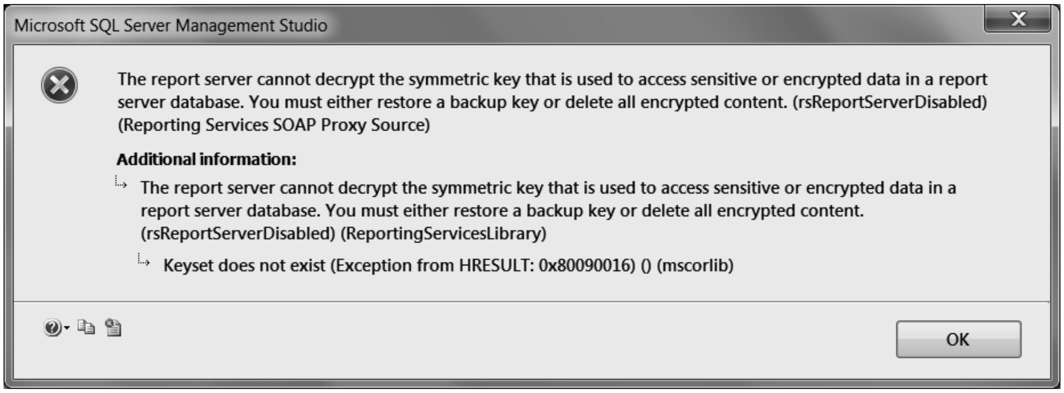
The Windows Service

Once you know that the databases exist, you need to prove that SSRS can connect to them. Do this by launching one or both of the Report Server websites. If you have successfully configured the service account (Figure 16-10), you will be able to connect to the web pages, and they will look similar to Figure 16-4. If the service account is not configured correctly, you will see an error message.

If you get an error, your first step is to verify that the account you are using for the service can connect to SQL Server. You can test this by logging into SQL Server as that particular user from SQL Server Management Studio and then prove that you can select data from the Catalog table in the ReportServer database. If you cannot, configure the service to use an account that does have access.

Encrypted Content

Another common error that may occur is a message stating that your encrypted content is not configured correctly. This can be quite confusing, because on a first install you do not expect to have any encrypted content. This message shows up when you try to use SQL Server Management Studio to interact with SSRS directly. For example, attempting to expand the Jobs node will give the error message shown in Figure 16-13.



The report server cannot decrypt the symmetric key that is used to access sensitive or encrypted data in a report server database. You must either restore a backup key or delete all encrypted content. (rsReportServerDisabled) (Reporting Services SOAP Proxy Source)

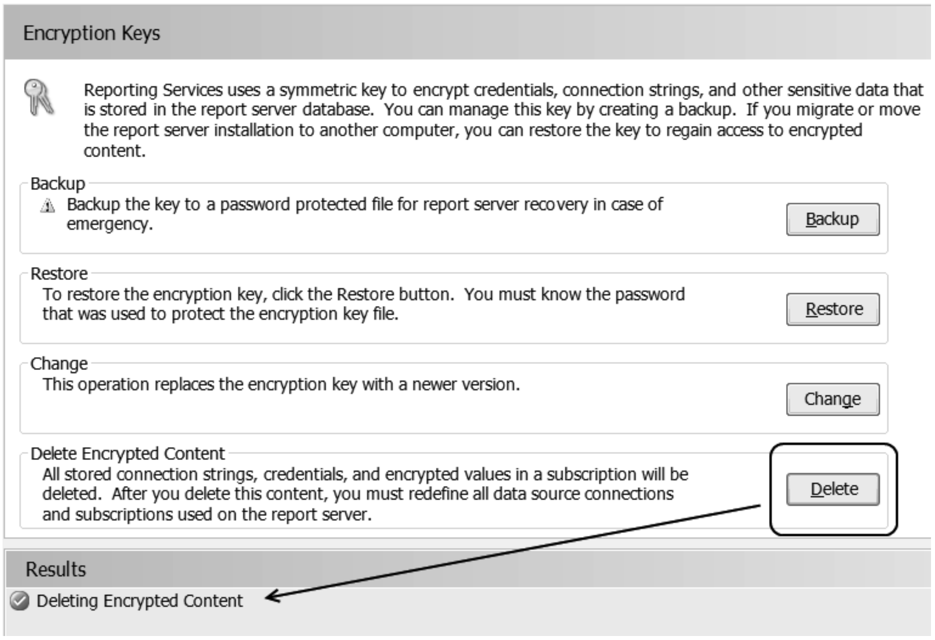
Additional information:

The report server cannot decrypt the symmetric key that is used to access sensitive or encrypted data in a report server database. You must either restore a backup key or delete all encrypted content. (rsReportServerDisabled) (ReportingServicesLibrary)

Keyset does not exist (Exception from HRESULT: 0x80090016) () (mscorlib)

Figure 16-13. *An error message when accessing SSRS from Management Studio*

The way around this is rather odd. First, use the Reporting Server Configuration Manager to access the Encryptions Keys page. Then, use the Delete Encrypted Content option to reset the Encryption Key process by clicking the Delete button (Figure 16-14).



Reporting Services uses a symmetric key to encrypt credentials, connection strings, and other sensitive data that is stored in the report server database. You can manage this key by creating a backup. If you migrate or move the report server installation to another computer, you can restore the key to regain access to encrypted content.

Backup
Backup the key to a password protected file for report server recovery in case of emergency.

Restore
To restore the encryption key, click the Restore button. You must know the password that was used to protect the encryption key file.

Change
This operation replaces the encryption key with a newer version.

Delete Encrypted Content
All stored connection strings, credentials, and encrypted values in a subscription will be deleted. After you delete this content, you must redefine all data source connections and subscriptions used on the report server.

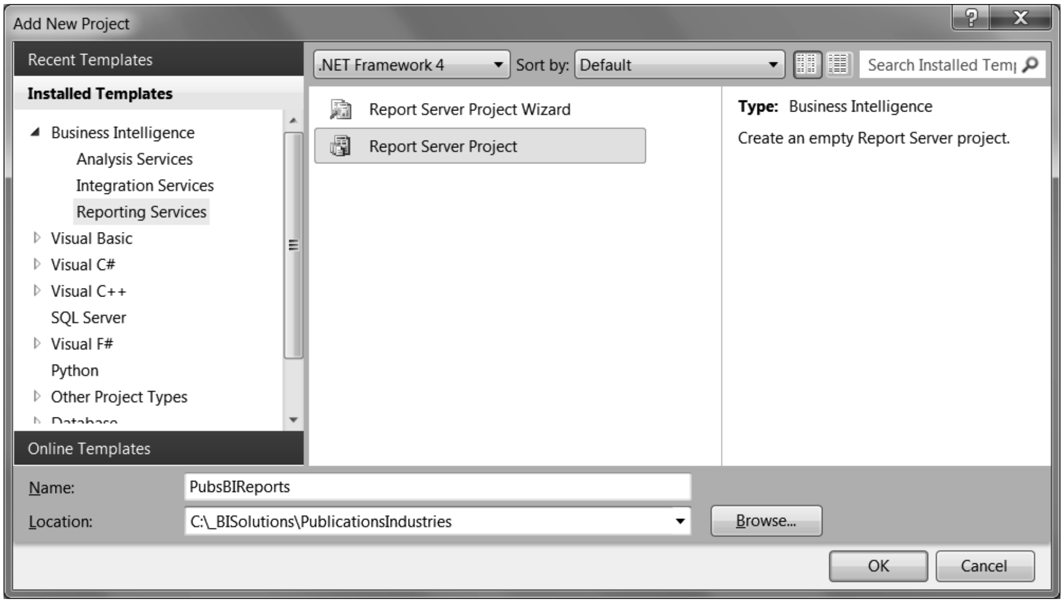
Figure 16-14. *An error message when accessing SSRS from Management Studio*

You are warned that this is a bad idea, but don't worry. It is safe to delete the encrypted content after you first install SSRS, because there is no encrypted content!

These three items represent the most common issues you are likely to encounter. If this is not enough to help you connect, we suggest searching the Web for *Reporting Services Configuration Manager* for more information. We have also included a video detailing an SSRS setup on the author's website at `www.NorthWestTech.org/SSRSDemos/SSRSSetupTips.aspx.`

Creating SSRS Objects

Most developers use Visual Studio to create their SSRS reports. To create reports with Visual Studio, you need to create a Visual Studio project. To add the project to the solution we have been working on throughout this book, start by opening the Publication Industries' Business Intelligence Solution and adding a new Report Server project to it, as you see in Figure 16-15.



Type: Business Intelligence
Create an empty Report Server project.
Name: PubsBIReports
Location: C:_BISolutions\PublicationsIndustries

Figure 16-15. *Adding a Reporting Services project to our existing BI solution*

Each SSRS project is a collection of XML files that define different SSRS objects. These objects include data sources, datasets, and reports. Visual Studio organizes these files into virtual folders within the Solution Explorer window. Let's take a closer look at each of these objects.

Data Sources

Once the project has been created, you can see it in Solution Explorer. Next, create one or more data sources. Data sources provide the connection information for your reports. This connection information can be stored either in the report RDL file or in a separate report data source (RDS) file. When the connection information is stored in the separate RDS file, it is referred to as a *shared data source*. Shared data sources can be referred to by many reports, and using shared data sources is considered a best practice. This is because it allows you to update a single data source and automatically affects all of the reports that refer to it.

To create a shared data source, right-click the folder called Shared Data Sources and select the Add New Data Source option from the context menu (Figure 16-16). The Shared Data Source Properties dialog window will appear.

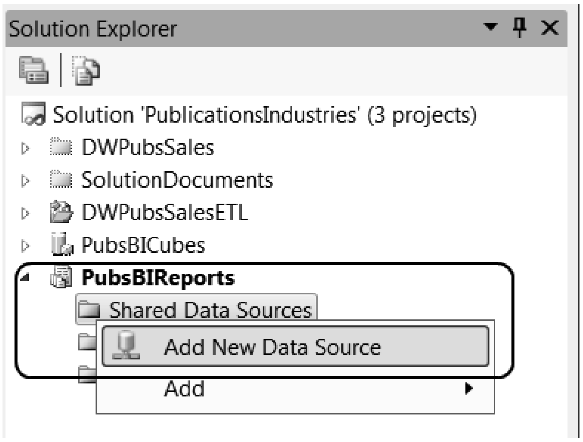
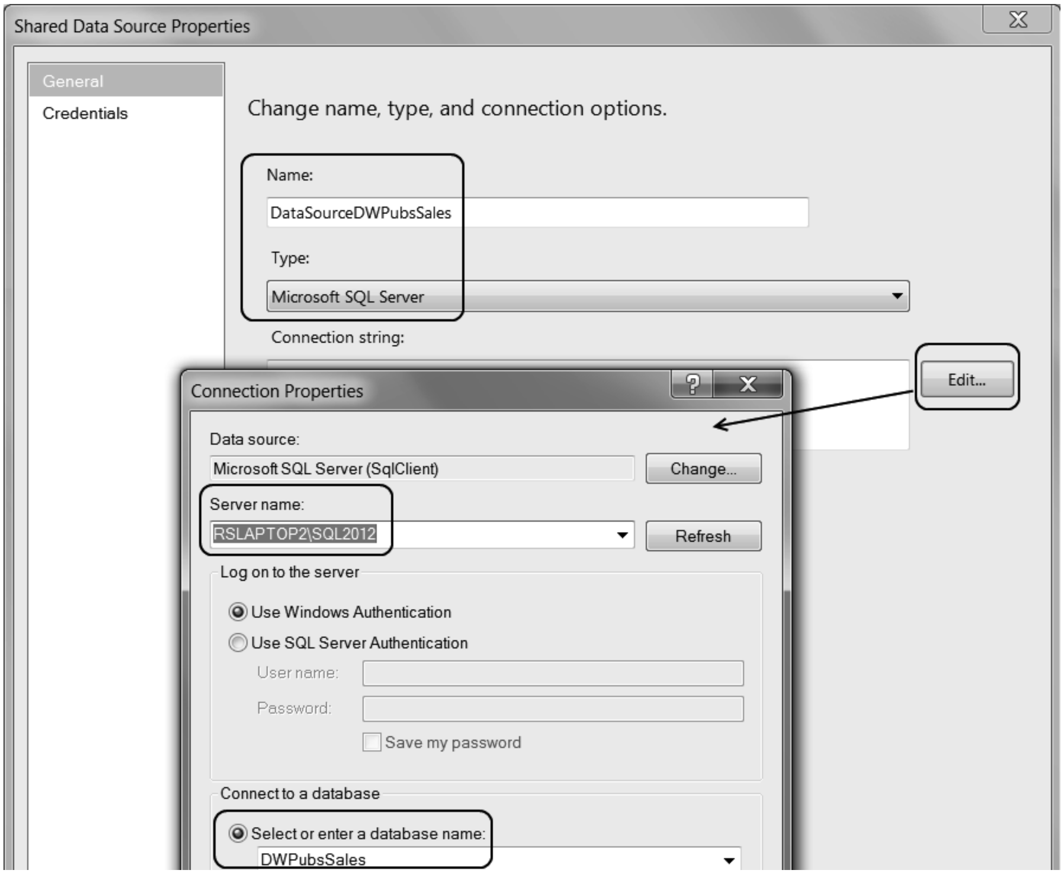


Figure 16-16. *Creating a shared data source*

In the Shared Data Source Properties dialog window, you enter a name for your new data source, select the type of connection to create, and provide a connection string (Figure 16-17).


Shared Data Source Properties

General
Credentials

Change name, type, and connection options.

Name: DataSourceDWPubsSales

Type: Microsoft SQL Server

Connection string:

Edit...

Connection Properties

Data source: Microsoft SQL Server (SqlClient)

Change...

Server name: RSLAPTOP2\SQL2012

Refresh

Log on to the server

(●) Use Windows Authentication

( ) Use SQL Server Authentication

User name:

Password:

[ ] Save my password

Connect to a database

(●) Select or enter a database name: DWPubsSales

Figure 16-17. Configuring a shared data source

The name is configured using the Name textbox and should represent the object that you are connecting to. For example, in Figure 16-17, we have named our connection **DataSourceDWPubsSales**.

The connection type is chosen using the dropdown box. In Figure 16-17 we have selected a Microsoft SQL Server connection type. The dropdown box also contains connections for Analysis Server, Oracle, OLE DB, XML, and many others. We use SQL Server and Analysis Server for our demonstrations.

To create a connection string, type in the connection string or click the Edit button to access the Connection Properties dialog window. Choose the server name and the database name in the Connection Properties dialog window, just as we have done in many previous chapters. Click OK to create the connection string.

After you have identified the name, type, and connection string, you close the Shared Data Source Properties dialog window by clicking OK, and the new shared connection shows up in Solution Explorer (Figure 16-18). You may notice that the extension on the new data source file is RDS, which stands for Report Data Source, and the connection information is stored in an XML format. Listing 16-1 shows an example of the contents of an RDS file.

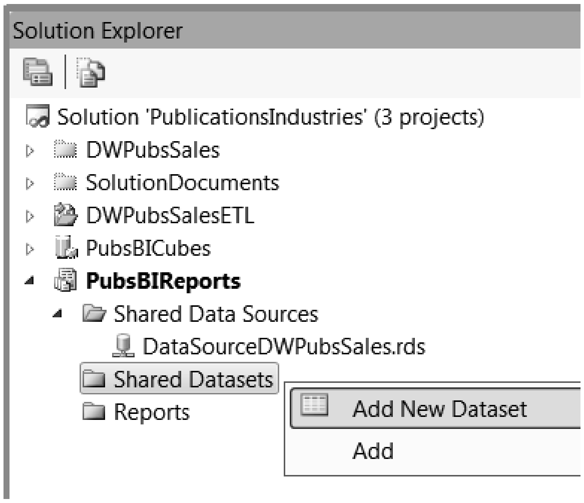


Figure 16-18. *Creating a new shared dataset*

Listing 16-1. XML Code in a Typical RDS Data Source File

```
<?xml version="1.0" encoding="utf-8"?>
<RptDataSource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" Name="DWPubsSalesDataSource">
  <ConnectionProperties>
    <Extension>SQL</Extension>
    <ConnectString>Data Source=RSLAPTOP2\SQL2012;Initial Catalog=DWPubsSales</ConnectString>
    <IntegratedSecurity>true</IntegratedSecurity>
  </ConnectionProperties>
  <DataSourceID>e54393b4-108f-4798-ad55-353a30c1e6e0</DataSourceID>
</RptDataSource>
```

Data source code can also be embedded within a reports' RDL file, but doing so means that the data source is available for only that individual report and cannot be used by other reports on the SSRS websites.

Datasets

After you create your data sources, you then make one or more datasets. Datasets contain query string used to generate report data. The query string language is dependent on which type of connection is used. For example, if you are connecting to a SQL Server database, use the SQL language. If you are connecting to an Analysis Server database, use a language such as MDX.

Datasets can be either embedded within a report's RDL file or created as a separate file with the extension of RSD. These Report Server dataset (RSD) files are simply XML files that contain your query sting as well as a cross-reference to a data source that is used to execute the query.

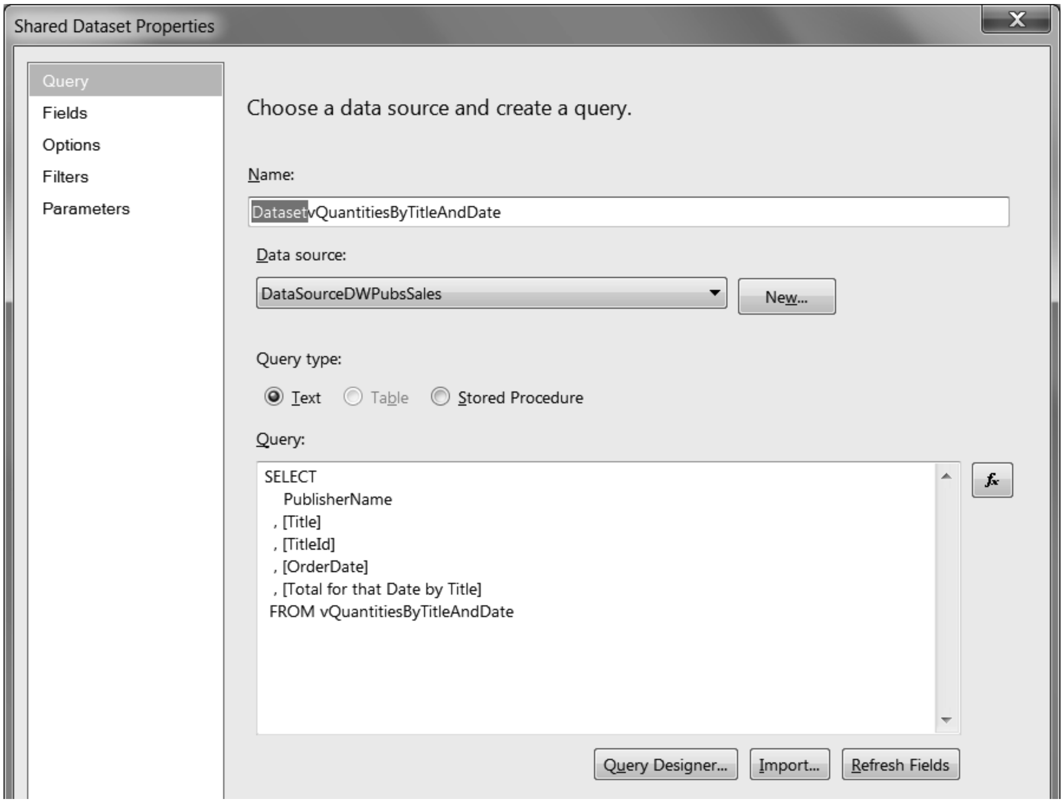
Listing 16-2 shows an example of an RSD file that contains a SQL statement. Notice that it also contains a reference to the data source from Listing 16-1. Any report using this dataset automatically looks up the data source and use it to execute the SQL statement and retrieves the report data.

Listing 16-2. XML Code in a Typical RSD Dataset File

```
<?xml version="1.0" encoding="utf-8"?>
<SharedDataSet xmlns:rd="http://schemas.microsoft.com/SQLServer/reporting/reportdesigner"
xmlns="http://schemas.microsoft.com/sqlserver/reporting/2010/01/shareddatasetdefinition">
  <DataSet Name="">
    <Query>
      <DataSourceReference>DWPubsSalesDataSource</DataSourceReference>
      <CommandText>SELECT
    PublisherName
  , [Title]
  , [TitleId]
  , [OrderDate]
  , [Total for that Date by Title]
 FROM vQuantitiesByTitleAndDate
</CommandText>
    </Query>
    <Fields>
      <Field Name="PublisherName">
        <DataField>PublisherName</DataField>
        <rd:TypeName>System.String</rd:TypeName>
      </Field>
      <Field Name="Title">
        <DataField>Title</DataField>
        <rd:TypeName>System.String</rd:TypeName>
      </Field>
      <Field Name="TitleId">
        <DataField>TitleId</DataField>
        <rd:TypeName>System.String</rd:TypeName>
      </Field>
      <Field Name="OrderDate">
        <DataField>OrderDate</DataField>
        <rd:TypeName>System.String</rd:TypeName>
      </Field>
      <Field Name="Total_for_that_Date_by_Title">
        <DataField>Total for that Date by Title</DataField>
        <rd:TypeName>System.Int32</rd:TypeName>
      </Field>
    </Fields>
  </DataSet>
</SharedDataSet>
```

To create a shared dataset, right-click the Shared Datasets folder in Solution Explorer and choose Add New Dataset from the context menu, as shown in Figure 16-18.

A Shared Dataset Properties dialog window appears where you can identify the name, data source, query type, and query for the dataset, as shown in Figure 16-19.



Choose a data source and create a query.

Name: DatasetvQuantitiesByTitleAndDate

Data source: DataSourceDWPubsSales

Query type: [✓ Text] Table Stored Procedure

Query:

```
SELECT
   PublisherName
 , [Title]
 , [TitleId]
 , [OrderDate]
 , [Total for that Date by Title]
 FROM vQuantitiesByTitleAndDate
```

Figure 16-19. *Configuring a shared dataset*

The name is configured by filling in the Name textbox. As always, the name you choose should describe its purpose. We often choose to use the table, view, or stored procedure name that is being accessed. In Figure 16-19 we have configured the dataset name to be the view name, DatasetvQuantitiesByTitleAndDate.

The “Data source” configuration uses a dropdown box that allows you to select a previously created shared data source, or you can make a new one if necessary. Using the New button launches the same dialog window shown in Figure 16-19.

The “Query type” configuration uses radio buttons (Figure16-19). There are only three options to choose from:

- Text
- Table
- Stored Procedure

The options will be either available or grayed out, depending on the type of data source. In the case of a Microsoft SQL Server data source, your options will be Text and Stored Procedure. But when using an Excel spreadsheet, the Table (representing a worksheet) option will appear, and Stored Procedure will be grayed out.

The “Query configuration” uses a large textbox where you can either type in the query code or click the Query Designer button to access the Query Designer dialog window (Figure 16-19).

Reports

In an SSRS project, the main task a developer performs is creating new reports. As you may have guessed, you create reports by right-clicking the Reports folder in Solution Explorer and choosing Add New Report.

But wait—there is a trick to this one! The Add New Report option launches the Report Wizard. Now, there is nothing wrong with the Report Wizard per se, but more experienced developers will want to create a new report from scratch without using a wizard.

■ **Note** If you have launched the Report Wizard by mistake, cancel the wizard and try again. Luckily, Visual Studio does not penalize you for starting a wizard and canceling it. We say luckily, because this inadvertent launching is likely to happen quite often. Doh!

To create a new report without using the Report Wizard, select the Add New Item option, as shown in Figure 16-20.

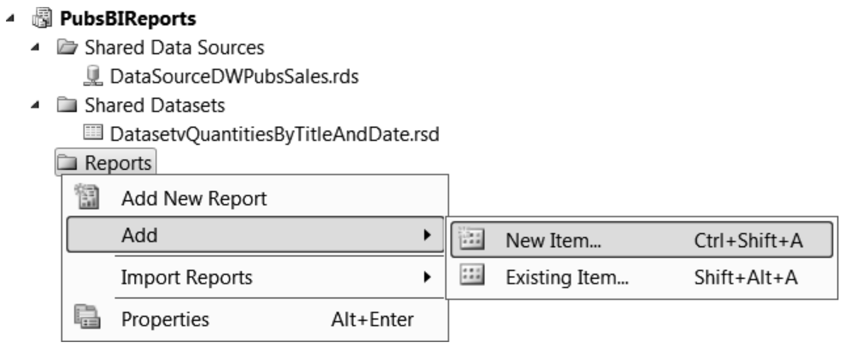
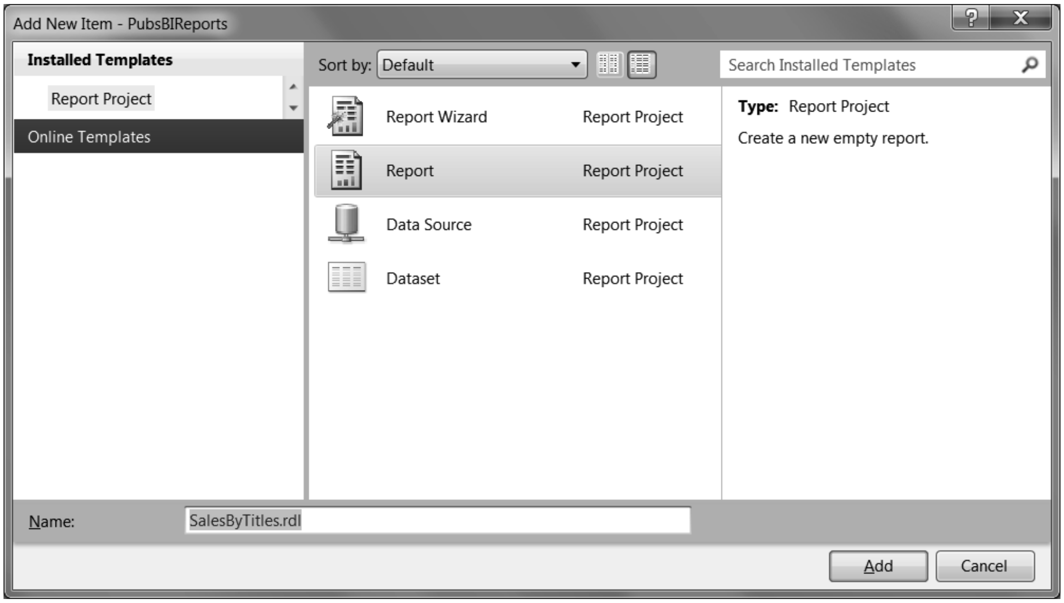


Figure 16-20. *Creating a new report without the Report Wizard*

After selecting the New Item option in the context menu, the Add New Item dialog window appears. Notice that, once again, the Report Wizard is present and available. In fact, it is the default selection! We are creating a report without using the wizard; therefore, we need to select the Report icon and provide a name in the Name textbox (Figure 16-21). Click Add to close the dialog window, and a new RDL file will be created in Solution Explorer.



Type: Report Project
Create a new empty report.

Name: SalesByTitles.rdl

Figure 16-21. *Selecting the Report icon and providing a name*

Mysteriously, at this point it is hard to say exactly what you will see, because your Visual Studio setup determines which windows are displayed. The two windows that are used most often are the Report Data and Toolbox windows. These windows are essential to configuring your reports, but they may or may not appear in Visual Studio when your new RDL file opens. Not to worry! You can force both of these windows to display by accessing the View menu in Visual Studio and selecting their corresponding menu items.

■ **Important** The Report Data menu item should be located at the very bottom of the View menu; however, it may not show up in the menu if you are not focused on an RDL file in Solution Explorer. If, for some reason, it does not show, make sure that you have the RDL file open, as shown in Figure 16-22. Then go back to the View menu and it should appear.

In Figure 16-22 we have placed the Report Data window on the left side of Visual Studio and the Toolbox on the right, but they can be dragged and dropped to various positions within Visual Studio.

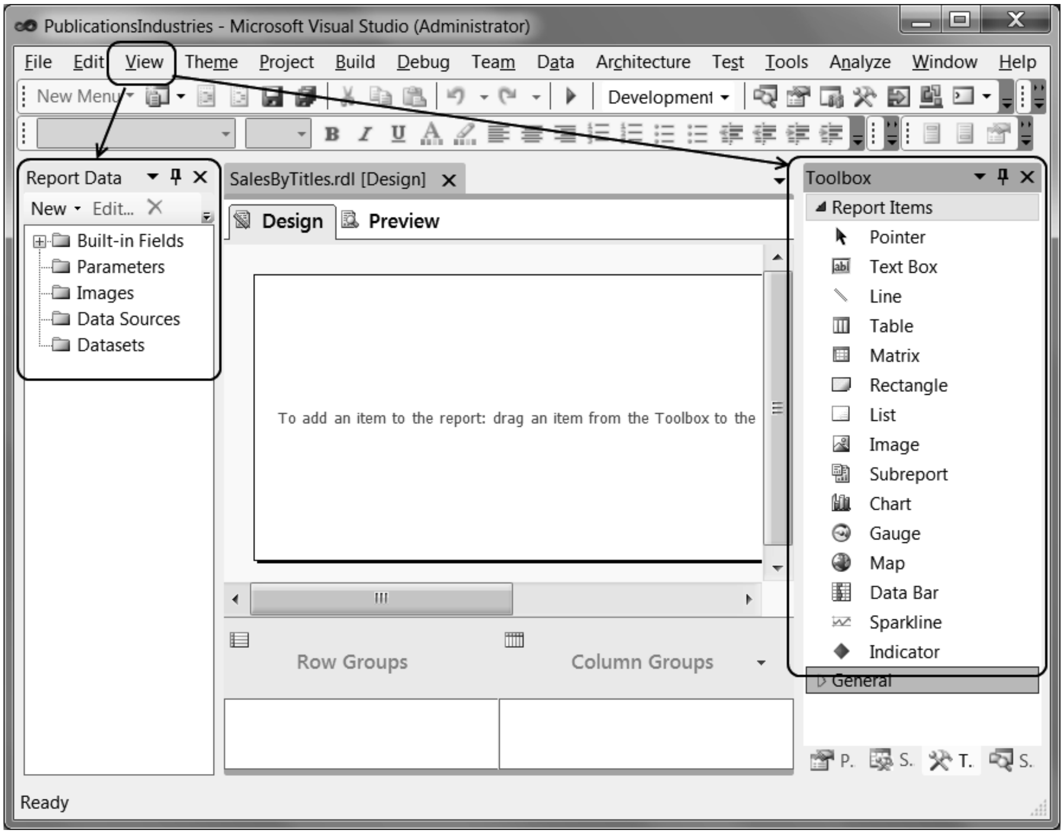

PublicationsIndustries - Microsoft Visual Studio (Administrator)
File Edit View Theme Project Build Debug Team Data Architecture Test Tools Analyze Window Help
Report Data
New Edit...
Built-in Fields
Parameters
Images
Data Sources
Datasets
SalesByTitles.rdl [Design]
Design
Preview
To add an item to the report: drag an item from the Toolbox to the
Row Groups
Column Groups
Toolbox
Report Items
Pointer
Text Box
Line
Table
Matrix
Rectangle
List
Image
Subreport
Chart
Gauge
Map
Data Bar
Sparkline
Indicator
General
Ready

Figure 16-22. *Preparing to configure the report using the Report Data and Toolbox windows*

Once you create your first report and set up your Report Data and Toolbox windows, we recommend you configure the report in this order:

1. Configure the report data.
2. Add report items from the Toolbox to the report's design surface.
3. Configure the various report items on the design surface.
4. Preview your report design in Visual Studio.
5. Deploy the report to the SSRS web service (optional).

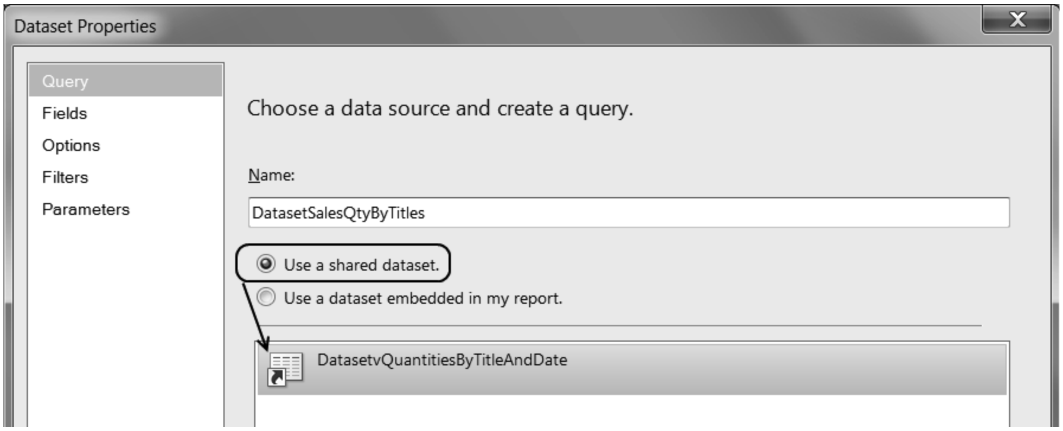
Configuring Report Data

You must identify the data that you are going to use in your report. To do so, you need two things. First, you need some way to connect to the data. Second, you need some kind of query to identify which data you want from the source.

To connect to the data you must create a local data source within the RDL file. This is necessary even if you have previously created a Shared Data Source earlier. Looking at Figure 16-22, you can see that the Report Data window has a Data Sources folder. From here, you can add a data source to the report by right-clicking this folder and selecting Add Data Source from the context menu, and the dialog window will appear. This dialog window works the same as the one used to create a Shared Data Source (Figure16-17).

■ **Tip** There is one important difference between the local and shared data source configuration dialogs; you can link a local data source to a shared data source using the *Use shared data source reference* radio button. When this option is selected, the data source connection string is not stored within the RDL file. Instead, the RDL file contains a reference to a shared data source and its connection string.

Once you have at least one Data Souce created, move on to creating Data Sets. You can add a dataset to the report by right-clicking the Datasets folder (Figure 16-22), and selecting Add Dataset from the context menu. As we discussed earlier, a dataset is a saved query that identifies the data that you want to use in your report. Clicking this context menu will display a dialog window like the one shown in Figure 16-23.



Choose a data source and create a query.

Name:
DatasetSalesQtyByTitles

(●) Use a shared dataset.
( ) Use a dataset embedded in my report.

DatasetvQuantitiesByTitleAndDate

Figure 16-23. *Referencing a shared dataset*

Like data sources, there are both local and shared datasets. In order for it to display data, each report must have a local dataset, but this dataset can either contain the select query itself or a reference to a Shared Dataset. When you compare the Shared Dataset dialog window shown in Figure 16-19 to the local Dataset window in Figure 16-23, notice the two radio buttons that allow you to either use a shared dataset or dataset embedded within the report. If you opt to use an embedded dataset, the SQL code will be saved within the RDL file. If you choose a shared dataset, the RDL file will cross-reference an RSD (Listing 16-2).

Both options provide you with data for your reports; however, using a shared dataset has the advantage of being reusable between multiple reports. This is also true of a shared data source. As a bonus, if you ever reconfigure these shared object, all changes are propagated to any reports that reference them. As you might suspect, using both shared data sources and shared dataset objects is considered a best practice.

In Figure 16-23, we are electing to use the shared dataset we defined previously. Now our report file will cross-reference the Shared Dataset file, which in turn cross-references the Shared Data Source file. Therefore, we only need to identify a shared dataset, and the shared data source will be cross-referenced implicitly.

Be sure to name the dataset appropriately. In Figure 16-23 we named it **DatasetSalesQtyByTitles**. Once the dataset has been created, the Report Data window will show the dataset by name and all the associated data fields beneath it. These data fields represent the columns or column aliases associated with the query results.

Adding Report Items

Once you have configured your report data, you can add report items by dragging and dropping them from the Toolbox window to the design surface. After you place items on the design surface, you can click each item to display sizing handles, as shown in Figure 16-24. The sizing handles allow you to adjust the size of the report item objects and are represented as little white or black boxes.

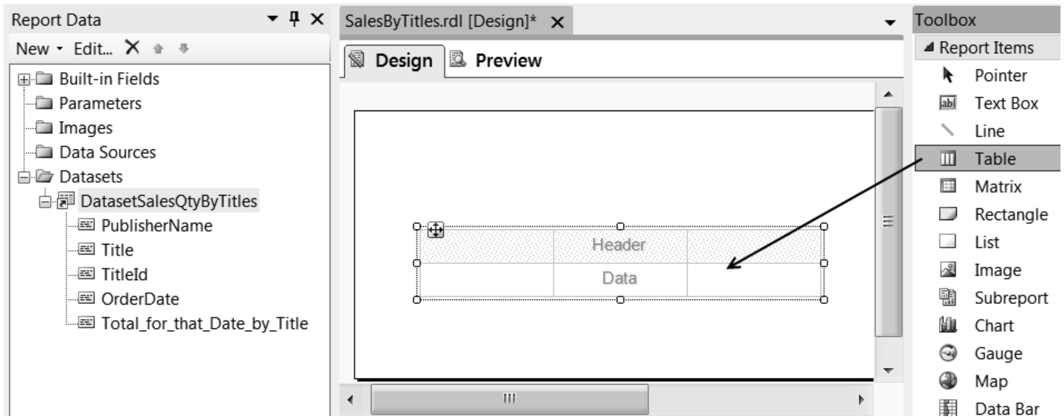


Figure 16-24. Adding a table to the report

In some cases, the sizing handles will be obscured by other design components. For example, if you click a Table report item, you first see the column and row design component shown in Figure 16-25 and not the sizing handles shown in Figure 16-24. By clicking the upper-left corner of the column and row design component, you are able to access the sizing handles of the Table report item.

■ **Tip** Like any GUI interface, it is a good idea to take a few moments to get used to it. Be patient with yourself and explore the environment for a while. You will soon find that the design tools are not nearly as bewildering as they first appear.

Configuring Report Items

Arguably the most time-consuming process of creating reports is the configuration process. During this phase you map data to the report items and change various properties until the report looks as intended.

To map data to a report item, click a data field in the Report Data window and drag it to the report item on the design surface, as indicated in Figure 16-25. You can also click a report item and choose a data field using the *Smart Tag* icon that magically appears. These smart tags are becoming more common in Microsoft's development tool and usually launch a context menu (Figure 16-25).

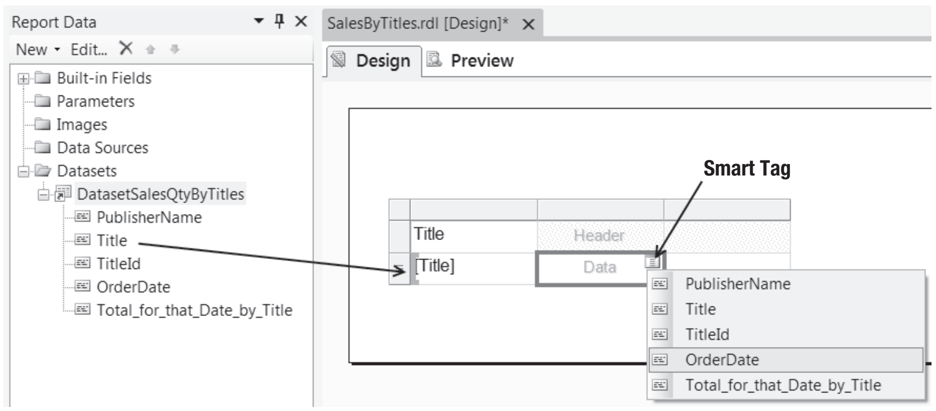
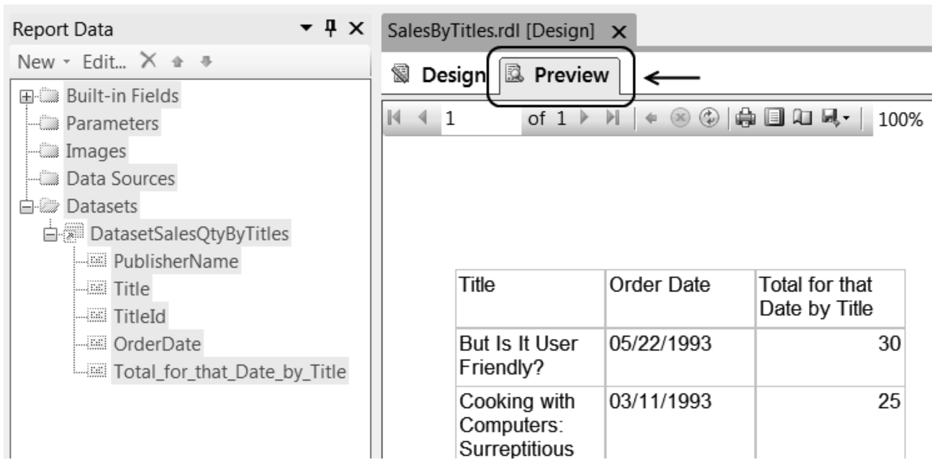


Figure 16-25. *Adding data to the table*

Previewing Reports

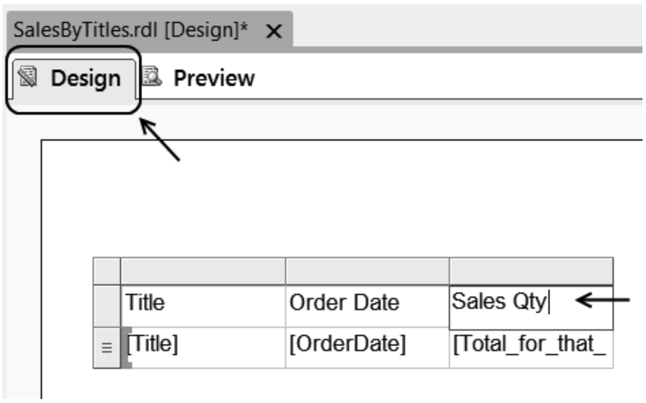
After you have made some changes to the report, it is important to preview your work to get an idea of how the report is coming along. When you click the Preview tab, located at the top of the Report Designer surface, a representation of the report will be displayed in HTML format. Figure 16-26 shows an example of the report we have configured so far, after having added three data fields, Title, OrderDate, and Total_for_that_Date_by_Title, to a table report item.



Title	Order Date	Total for that Date by Title
But Is It User Friendly?	05/22/1993	30
Cooking with Computers: Surreptitious	03/11/1993	25

Figure 16-26. *Previewing the report*

As you can see in Figure 16-25, tables are divided into data and header rows, but the table looks different on the Design and Preview tabs. In our example, the SSRS table report item displays one header row and multiple data rows on the Preview tab (Figure 16-26), but only one header and data row on the Design tab (Figure 16-27). Expect to navigate between them repeatedly as you make configuration changes.

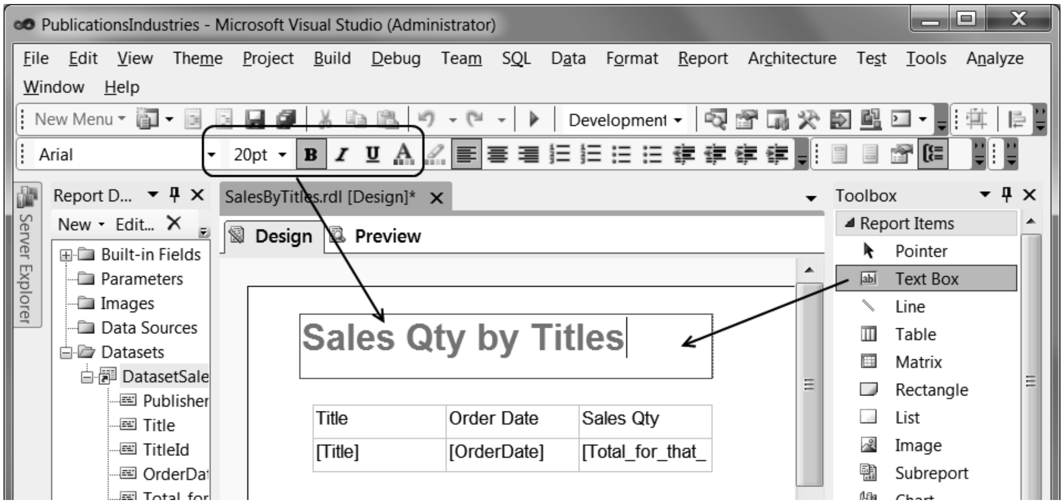


Title	Order Date	Sales Qty
[Title]	[OrderDate]	[Total_for_that_

Figure 16-27. *Changing the column titles*

As you preview your report, you may notice things that you would like to change. For instance, you may want to change the header text. You can do this by clicking the text once to select it, pausing for a second, and then clicking the text once more. A cursor appears that allows you to delete and retype the text (Figure 16-27).

You can add several other items to the report, such as a Text Box report item to display the report header (Figure 16-28). You can also format the text within the Text Box by changing the font colors with the tools on the toolbar (circled in Figure 16-28).

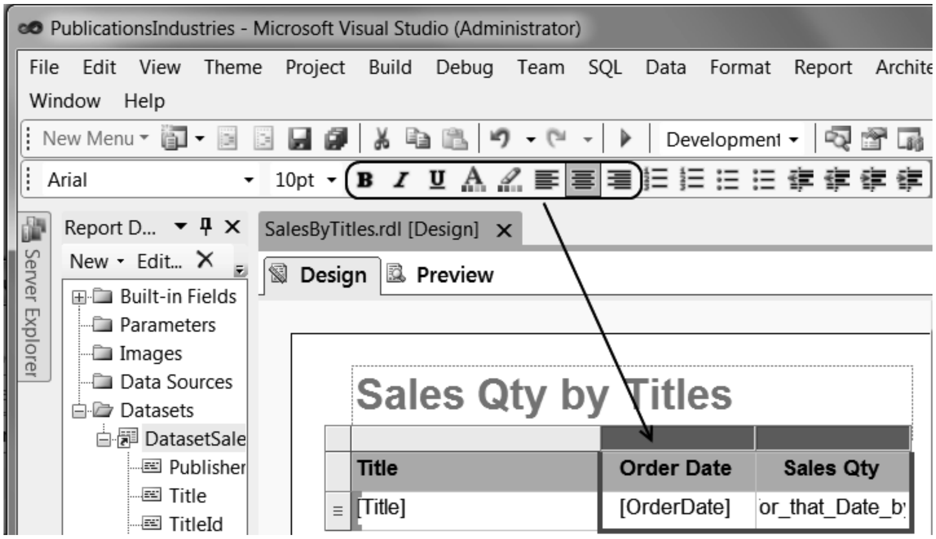


Sales Qty by Titles

Title	Order Date	Sales Qty
[Title]	[OrderDate]	[Total_for_that_

Figure 16-28. *Formatting report textboxes*

Tables can be formatted as well. For example, you can highlight the entire header row by clicking the gray Row button on the left side of the Table report item and then using the Format toolbar just as you would with a text box (Figure 16-29).

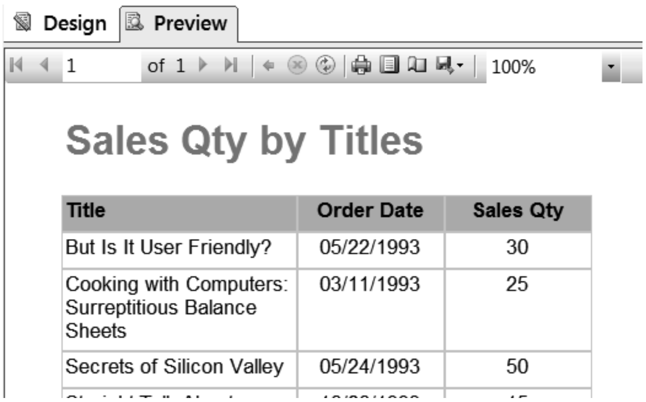


Title	Order Date	Sales Qty
[Title]	[OrderDate]	or_that_Date_b

Figure 16-29. *Formatting table report items*

After making a few more format changes, you can preview the changes and continue configuring the report until you are satisfied with the result.

You can see in Figure 16-30 that our report isn't a work of art, but it is sufficient for our needs. It is time to give it a try in the following exercise.



Sales Qty by Titles

Title	Order Date	Sales Qty
But Is It User Friendly?	05/22/1993	30
Cooking with Computers: Surreptitious Balance Sheets	03/11/1993	25
Secrets of Silicon Valley	05/24/1993	50

Figure 16-30. *Viewing your changes*

EXERCISE 16-1. CREATING A REPORT

In this exercise, you add an SSRS project to your existing Publications Industries solution. You then create a shared data source, a shared dataset, and a basic report.

1. Open Visual Studio by clicking the Windows Start button and selecting All Programs ➤ Microsoft SQL Server 2012 ➤ Data Tools.

Important: You are practicing administrator-level tasks in this book; therefore, you need administrator-level privileges. The easiest way to achieve this is to remember to always right-click a menu item, select Run as Administrator, and then answer Yes to access administrator-level privileges. In Windows 7 and Vista, logging in with an administrator account is not enough. For more information, search the Web for the keywords *Windows 7 True Administrator and User Access Control.*

2. Open the Publications Industries solution you have been working on throughout the book using the File ➤ Open Project/Solution menu item.
3. Locate and select the SLN file `C:\_BISolutions\PublicationsIndustries\PublicationsIndustries.sln`.
4. Click the Open button at the bottom of the dialog window to open the solution.

Add a Project to the Current Solution

1. Add a new project to the solution by selecting File ➤ Add ➤ New Project from the main menu at the top of Visual Studio. An Add New Project dialog window appears (Figure 16-15).
2. Select the Business Intelligence ➤ Reporting Services template category on the left of the dialog window. Then select the Reporting Server Project option from the center of the dialog window (Figure 16-15).
3. Name the Project PubsBIReports at the bottom of the dialog window and click OK to add the new project to your current solution (Figure 16-15).
4. Use the View ➤ Solution Explorer menu item to display the Solution Explorer window.

Create a Shared Data Source

We need a way to connect to the data warehouse to access our report data. Let's create a shared data source.

1. Add a new shared data source by right-clicking the Shared Data Sources folder and selecting the Add New Data Source option in the context menu (Figure 16-16).
2. When the Shared Data Source Properties dialog window appears (Figure 16-17), type **DataSourceDWPubsSales** in the Name textbox.
3. In the Type dropdown box, select Microsoft SQL Server (Figure 16-17).
4. Click the Edit button next to the Connection String textbox and type in the name of your SQL Server, as shown in Figure 16-17.
5. Using the dropdown box beneath the "Select or enter a database name" radio button, select the DWPubsSales database (Figure 16-17).
6. Close both dialog windows by clicking the OK buttons to complete the creation of the shared data source.

Create a Shared Dataset

Along with our shared data source, we want a shared dataset as well. Let's create one now.

1. Right-click the Shared Datasets folder in Solution Explorer and choose the Add New Dataset option from the context menu (Figure 16-18).
2. When the Shared Properties dialog window appears, type **DatasetvQuantitiesByTitleAndDate** into the Name textbox (Figure 16-19).
3. Use the Data Source dropdown box to select the DataSourceDWPubsSales Data Source (Figure 16-19).
4. Type the SQL code found in Listing 16-3 in the Query textbox.

   Listing 16-3. SQL Code for the DatasetvQuantitiesByTitleAndDateDataset

   ```
   SELECT
     PublisherName
    , [Title]
    , [TitleId]
    , [OrderDate]
    , [Total for that Date by Title]
    FROM vQuantitiesByTitleAndDate
   ```

5. Close the dialog window by clicking OK to complete the creation of the shared dataset.

Important: The view that the SQL code uses was created in Chapter 13, but if you do not have it already in your database, you can open the SQL script file `C:\_BookFiles\Chapter16Files\ViewAndStoredProcedureForChapter16.sql` and execute its code to create it.

Create a Basic Report

Next, we need to add a report to the project and configure it to display the report data in a tabular format.

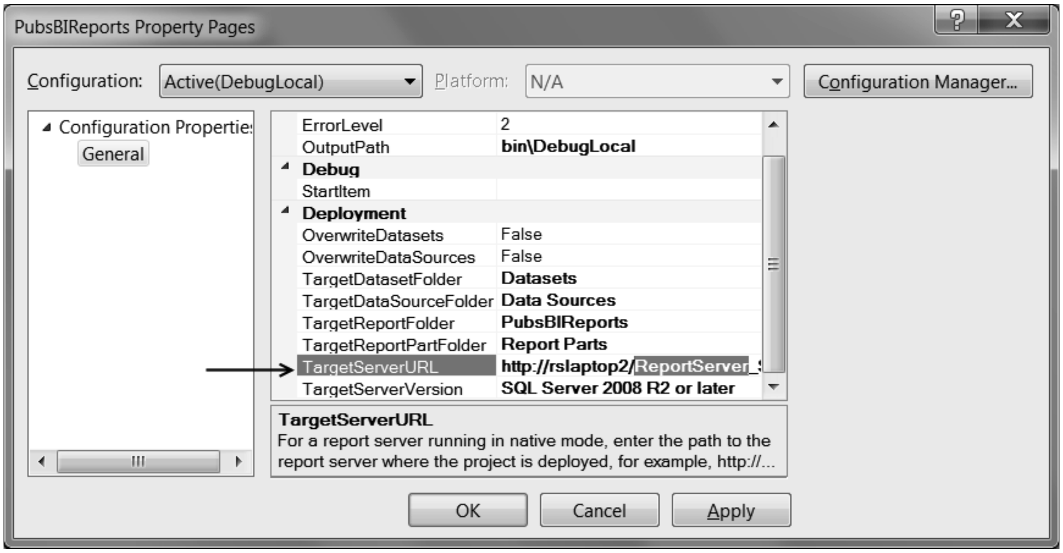
1. Add a new report to the project by right-clicking the Reports folder in Solution Explorer and selecting New Item from the context menu (Figure 16-20).
2. When the Add New Item dialog window appears, select the Report icon, as shown in Figure 16-21.
3. Configure the Name textbox with the filename of `SalesByTitles.rdl`. Then click the Add button to add a new report to the project.
4. Once the Report Designer window appears, use the View menu to display both the Report Data and Toolbox windows (Figure 16-22).
5. In the Report Data window, right-click the Datasets folder and select Add Dataset from the context menu.
6. When the dataset Properties window appears, name the dataset **DatasetSalesQtyByTitles***,* select the "Use a shared dataset" radio button, and select DatasetvQuantitiesByTitleAndDate, as shown in Figure 16-23. Finally, click OK to close the dialog window.

7. Add a table to your report by clicking the Table icon in the Toolbox and dragging it to the Report Designer surface (Figure 16-24).
8. In the new table, click the first column of the Data row. When the Smart Tag icon appears, click it to display the context menu. When the context menu appears, select the Title data field, and it will be added to the report table (Figure 16-25).
9. Repeat this process to add the OrderDate data field to the second column of the data row and to add the Total_for_That_Date_by_Title data field to the third column.
10. Use the Preview tab and review your current report design. It should look similar to the one shown in Figure 16-26.
11. Use the Design tab to change the names of the columns to Title*,* Order Date*,* and Sales Qty, as shown in Figure 16-27.
12. Add a Text Box to the report and type in **Sales Qty by Titles**, as shown in Figure 16-28.
13. Use the Format toolbar to format both the table and textbox to your liking.
14. Once you have completed your formatting, take a look at the report using the Preview tab. Continue adjusting it until you are satisfied with the look of your report.

In this exercise, you created a new SSRS project and added it to a current solution. You then added a data source, a data set, and a simple report to the project. Now we can deploy the report to the Reporting Services web service so that users can access the reports either through the SSRS web service or more appropriately through the SSRS web application.

Deploying the Report

To deploy the report to the SSRS web service, you need to configure the Visual Studio project appropriately. This can be done by accessing the property pages shown in Figure 16-31. To access this property sheet, right-click the Project icon in Solution Explorer and select Properties from the context menu.



ErrorLevel	2
OutputPath	bin\DebugLocal
Debug	
StartItem	
Deployment	
OverwriteDatasets	False
OverwriteDataSources	False
TargetDatasetFolder	Datasets
TargetDataSourceFolder	Data Sources
TargetReportFolder	PubsBIReports
TargetReportPartFolder	Report Parts
TargetServerURL	http://rslaptop2/ReportServer_S
TargetServerVersion	SQL Server 2008 R2 or later

TargetServerURL
For a report server running in native mode, enter the path to the report server where the project is deployed, for example, http://...

Figure 16-31. *Configuring the deployment options*

The most important setting in this list of settings is the Target Server URL. This configuration tells Visual Studio which SSRS web service to connect to. Because a company can have many different SSRS web services available (and it is assumed that developers will not necessarily have their own on their individual desktops), the property is left blank by default.

To add the appropriate URL to this textbox, type in the following (with no spaces):

- `HTTP:` including the colon, to indicate that you want to use the HTTP protocol
- `//<ServerName>` to indicate which web server you want to connect to
- `/<virtual directory>` to indicate which virtual directory on that web server you want to use

Of course, `<ServerName>` and `<virtual directory>` are placeholders for the actual names that you need to use. Here is an example of how the configuration is set up on Randal's laptop: `http://rslaptop2/ReportServer_SQL2012`.

■ **Tip** The properties' default settings should be sufficient for your needs at first. As you become more familiar with SSRS and how the web application is organized, you may want to come back and change things, such as folder names where the report objects are placed.

To deploy an individual report, right-click the report file in Solution Explorer and select Deploy from the context menu, as shown in Figure 16-32. This asks Visual Studio to send the individual RDL file to the web service. When this happens, the web service will shred the XML code found in the RDL file and place the corresponding programming instructions into the SQL Server SSRS databases.

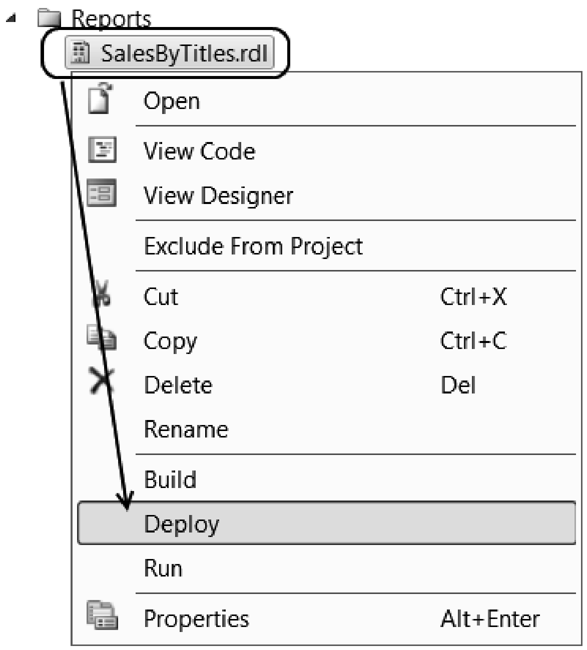
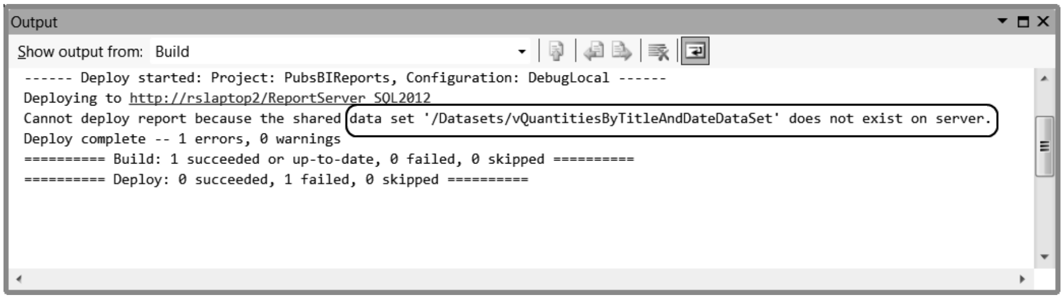


Figure 16-32. *Deploying the report*

When the report is deployed, SSRS will review the code in your RDL file and display an error report if required supporting objects are missing. For example, if we deploy the report we created in this chapter without first deploying the shared data source and shared dataset, SSRS will notice this discrepancy and display the error shown in Figure 16-33.



```
------ Deploy started: Project: PubsBIReports, Configuration: DebugLocal ------
Deploying to http://rslaptop2/ReportServer_SQL2012
Cannot deploy report because the shared data set '/Datasets/vQuantitiesByTitleAndDateDataSet' does not exist on server.
Deploy complete -- 1 errors, 0 warnings
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
========== Deploy: 0 succeeded, 1 failed, 0 skipped ==========
```

Figure 16-33. *An error in deploying the report*

The way to resolve this error is to deploy all the supporting objects before you try to deploy the reports. One simple way to do this is to use the Deploy option at the project level, which will attempt to deploy all the objects within the project simultaneously (Figure 16-34).

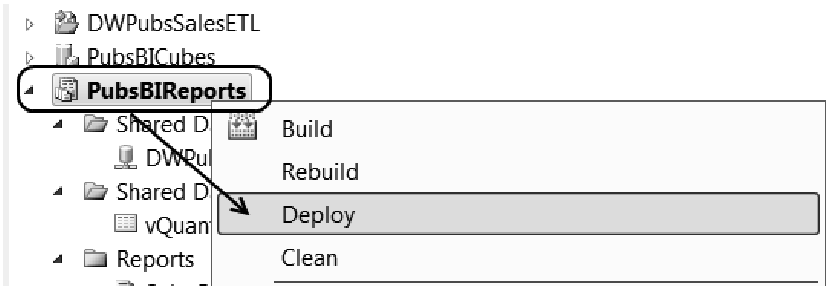
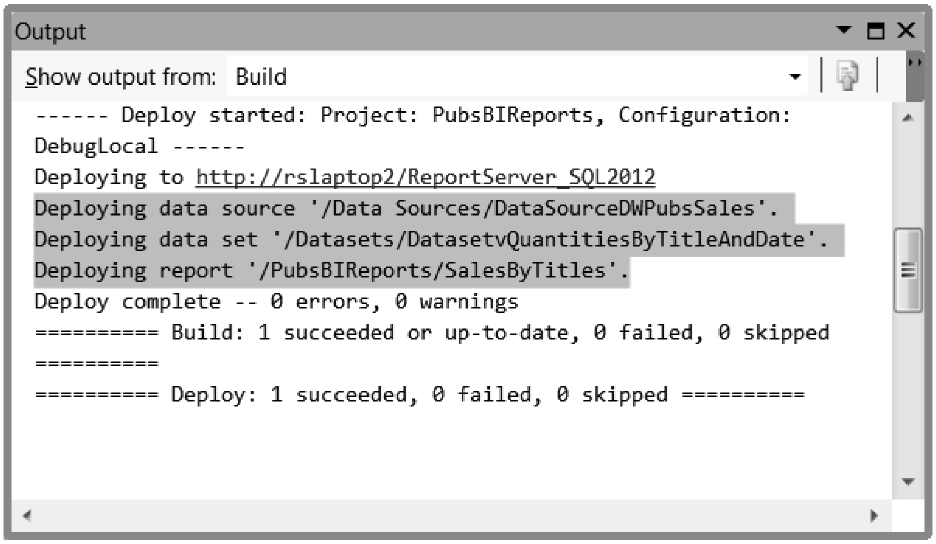


Figure 16-34. *Deploying the project*

After you deploy one or more files successfully to the SSRS web service, Visual Studio will indicate that the deployment succeeded in its Output window, as shown in Figure 16-35.

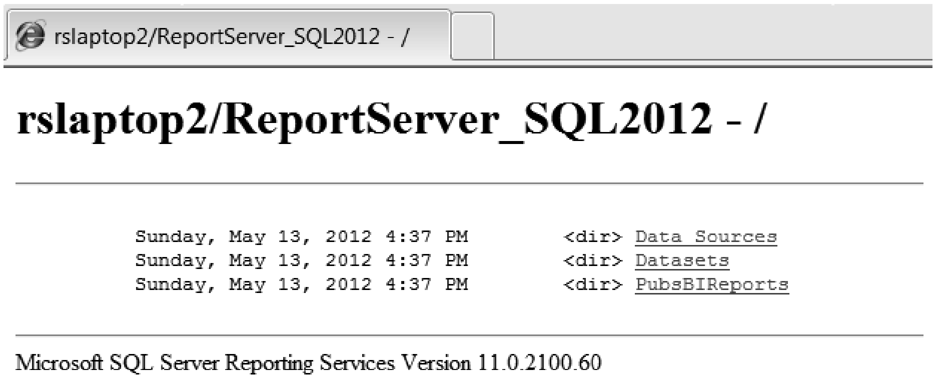


```
------ Deploy started: Project: PubsBIReports, Configuration:
DebugLocal ------
Deploying to http://rslaptop2/ReportServer_SQL2012
Deploying data source '/Data Sources/DataSourceDWPubsSales'.
Deploying data set '/Datasets/DatasetvQuantitiesByTitleAndDate'.
Deploying report '/PubsBIReports/SalesByTitles'.
Deploy complete -- 0 errors, 0 warnings
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped
==========
========== Deploy: 1 succeeded, 0 failed, 0 skipped ==========
```

Figure 16-35. *Deployment success*

If you navigate to the report's SSRS web service, you can see links to three new folders (as shown in Figure 16-36):

- Data Sources
- Datasets
- PubsBIReports (our Visual Studio project name)



rslaptop2/ReportServer_SQL2012 - /

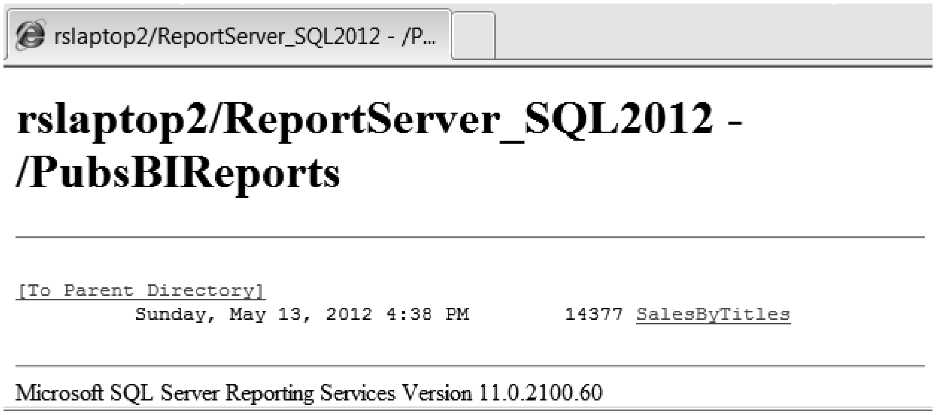
rslaptop2/ReportServer_SQL2012 - /

```
        Sunday, May 13, 2012 4:37 PM        <dir> Data Sources
        Sunday, May 13, 2012 4:37 PM        <dir> Datasets
        Sunday, May 13, 2012 4:37 PM        <dir> PubsBIReports
```

Microsoft SQL Server Reporting Services Version 11.0.2100.60

Figure 16-36. *Report folders on the ReportServer web service*

Clicking one of these links such as the PubsBIReports directory will display the objects within the subfolder. For example, in Figure 16-37, we have one report deployed within this folder called SalesByTitles.



rslaptop2/ReportServer_SQL2012 - /P...

rslaptop2/ReportServer_SQL2012 - /PubsBIReports

```
[To Parent Directory]
        Sunday, May 13, 2012 4:38 PM        14377 SalesByTitles
```

Microsoft SQL Server Reporting Services Version 11.0.2100.60

Figure 16-37. *The SalesByTitles report is accessed from the PubBIReports folder.*

Managing the Report

Most users will not use the web service to access their reports. Instead, they will use the web application known as Report Manager. By navigating to the Report Manager website, you see the same folders displayed in a graphical user interface that is much less utilitarian than that of the web service (Figure 16-38). The user interface contains links and icons to the folders and a toolbar with standard functions such as configuring folders, creating data sources, or uploading RDL files.

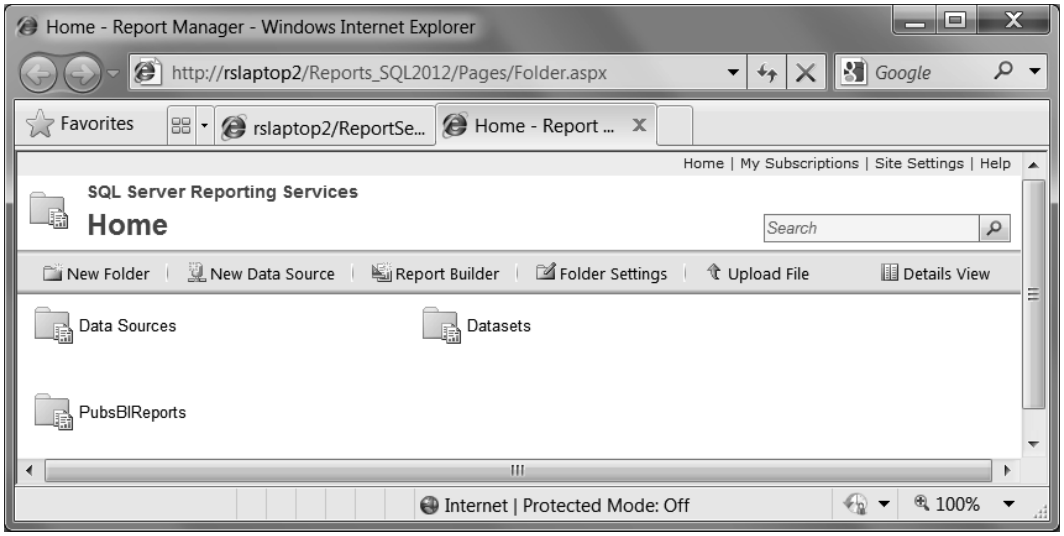

Home - Report Manager - Windows Internet Explorer
http://rslaptop2/Reports_SQL2012/Pages/Folder.aspx
Google
Favorites
rslaptop2/ReportSe...
Home - Report ...
Home | My Subscriptions | Site Settings | Help
SQL Server Reporting Services
Home
Search
New Folder
New Data Source
Report Builder
Folder Settings
Upload File
Details View
Data Sources
Datasets
PubsBIReports
Internet | Protected Mode: Off
100%

Figure 16-38. *Report folders on the Report Manager web application*

By clicking an individual folder, such as the PubsBIReports folder, you will see that the report is represented with an icon as well (Figure 16-39).

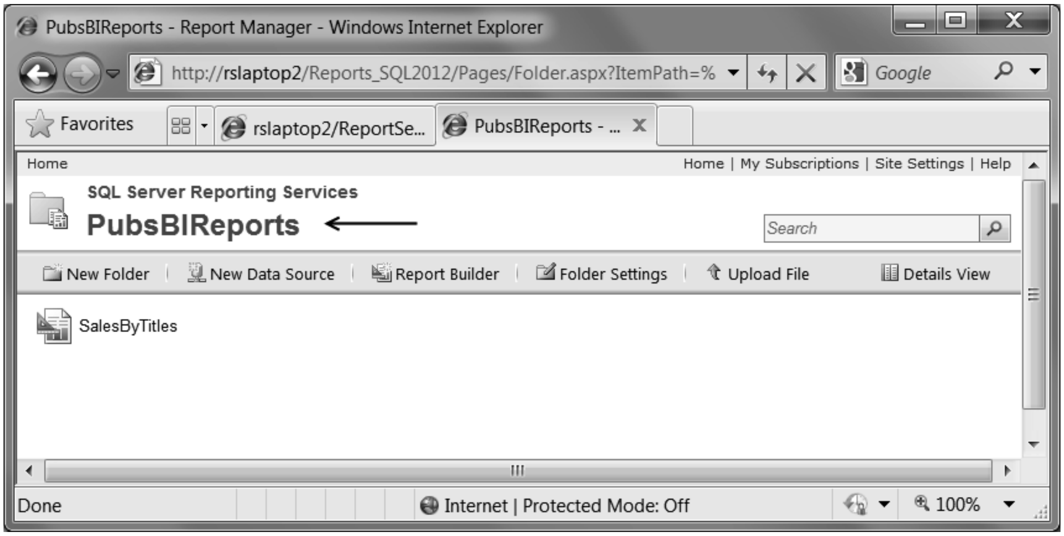
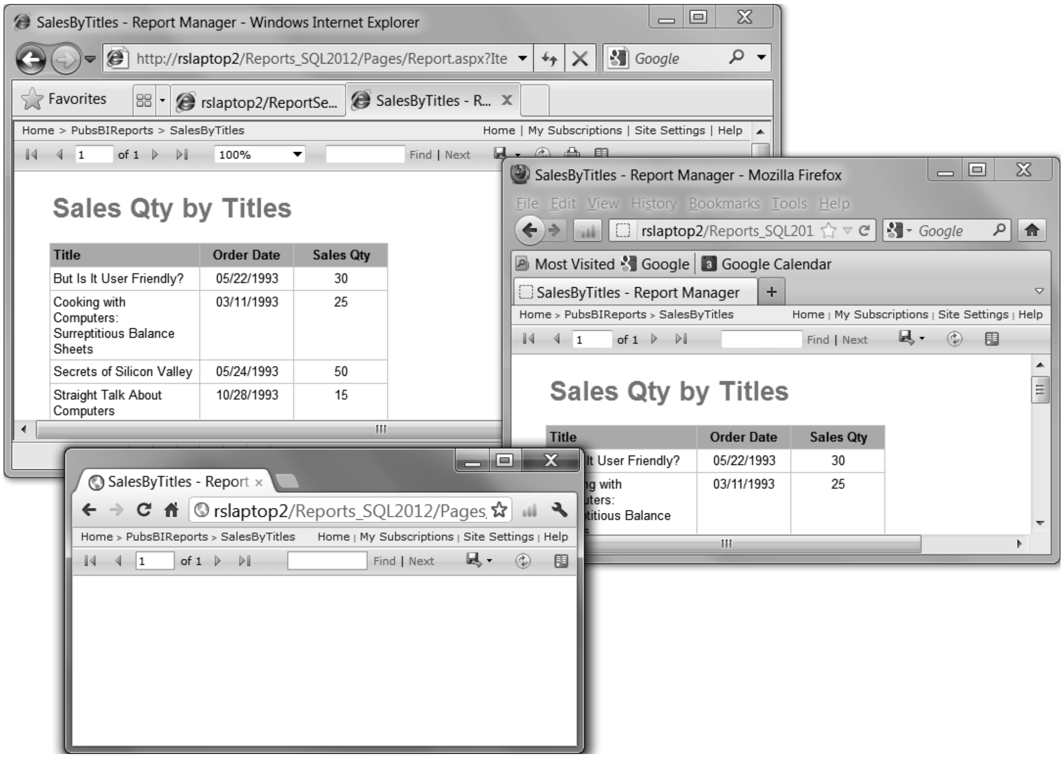

PubsBIReports - Report Manager - Windows Internet Explorer
http://rslaptop2/Reports_SQL2012/Pages/Folder.aspx?ItemPath=%
Google
Favorites
rslaptop2/ReportSe...
PubsBIReports - ...
Home
Home | My Subscriptions | Site Settings | Help
SQL Server Reporting Services
PubsBIReports
Search
New Folder
New Data Source
Report Builder
Folder Settings
Upload File
Details View
SalesByTitles
Done
Internet | Protected Mode: Off
100%

Figure 16-39. *Reports can be accessed in the Report Manager folders*

Clicking the Report icon will bring up the Report Viewer nested within the SSRS web service. Remember that it is a good idea to test the look of your reports in different browsers since they can display the report in different ways. Notice how in Figure 16-40 this report displays correctly when viewed in Internet Explorer or Firefox, but not at all in Chrome.


SalesByTitles - Report Manager - Windows Internet Explorer
http://rslaptop2/Reports_SQL2012/Pages/Report.aspx?Ite
Favorites
rslaptop2/ReportSe...
SalesByTitles - R...
Home > PubsBIReports > SalesByTitles
Home | My Subscriptions | Site Settings | Help
1 of 1
100%
Find | Next
SalesByTitles - Report Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
rslaptop2/Reports_SQL201
Most Visited
Google
Google Calendar
SalesByTitles - Report Manager
SalesByTitles - Report
rslaptop2/Reports_SQL2012/Pages

Sales Qty by Titles

Title	Order Date	Sales Qty
But Is It User Friendly?	05/22/1993	30
Cooking with Computers: Surreptitious Balance Sheets	03/11/1993	25
Secrets of Silicon Valley	05/24/1993	50
Straight Talk About Computers	10/28/1993	15

Sales Qty by Titles

Title	Order Date	Sales Qty
lt User Friendly?	05/22/1993	30
ng with uters: titious Balance	03/11/1993	25

Figure 16-40. *Testing the report on the web application*

Moving On

SSRS is a fantastic tool for creating reports. It provides multiple ways to develop, view, and store reports. It also provides flexibility in the dispersal of the different components of SSRS across one or more machines and provides an administrator with opportunities to tune the reporting performance. You now have an idea of how the different components work together to provide a complete reporting solution.

We recommend you try your hand at creating your own reports for the Northwind database in the following "Learn by Doing" exercise.

LEARN BY DOING

In this "Learn by Doing" exercise, you create an SSRS project similar to the one defined in this chapter using the Northwind database. We have included an outline of the steps you performed in this chapter and an example of how this can be completed in two Word documents. These documents are found in the folder `C:\_BISolutionsBookFiles\_LearnByDoing\Chapter16Files`. Please see the `ReadMe.doc` file for detailed instructions.

What's Next?

This far, all of the reports you have made in this book have been relatively basic. In Chapter 2, we introduced creating a report using SSRS's wizard. In this chapter, we showed you how to create a report using the Toolbox and how to design it yourself. In the next chapter, we discuss using various Toolbox tools to create professional-looking reports.

If you are interested in more information about the administration of SSRS, we recommend the book *Microsoft SQL Server 2012 Reporting Services, 4th edition* by Brian Larson (McGraw-Hill Osborne).