Microsoft Dynamics AX 2012 R3 Reporting Cookbook

Dynamics AX 2012 is a modern ERP solution from Microsoft that is targeted at mid- and high-level organizations. Reports play a very central role in an ERP system, being accessible to customers, vendors, top management, and the shop floor. Every ERP implementation includes reporting changes as companies try to refit the reports to their processes.

Microsoft Dynamics AX 2012 R3 Reporting Cookbook focuses mainly on the reporting aspects of Microsoft Dynamics AX 2012. As a Dynamics AX developer, reporting skills are a must-have and this book intends to make report development easier and simpler.

You will gain an understanding of all the various types of report formats such as charts, matrices, tabular in precision, and auto design. This book also offers strategies for new reports as well as for migrating your reports from the old reporting framework and development.

Who this book is written for

Microsoft Dynamics AX 2012 R3 Reporting Cookbook is recommended for Dynamics AX developers and .NET-based SSRS developers looking to familiarize themselves with the new AX reporting framework.

What you will learn from this book

- Create and deploy reports in AX 2012 R3
- Understand the SSRS report programming flow
- Modify the reporting framework
- Create RDP-based reports
- Design tabular matrices and chart-based reports
- Debug reports, the reporting framework, and business logic
- Migrate reports from the legacy reporting framework
- Get to grips with unit testing and troubleshooting SSRS reports

Over 90 recipes to help you resolve your new SSRS Reporting woes in Dynamics AX 2012 R3

Microsoft Dynamics AX 2012 R3 Reporting Cookbook

Deepak Agarwal                      Chhavi Aggarwal
Kamalakannan Elangovan

In this package, you will find:

- The authors biography
- A preview chapter from the book, Chapter 1 'Understanding and Creating Simple SSRS Reports'
- A synopsis of the book’s content
- More information on Microsoft Dynamics AX 2012 R3 Reporting Cookbook

About the Authors

Deepak Agarwal (Microsoft MVP) is a Microsoft Certified Professional and has been working professionally on Dynamics AX since 2011. He has worked with different versions of Axapta, such as AX 2009, AX 2012, R2, and R3. He has had a wide range of development, consulting, and leading roles, while always maintaining a significant role as a business application developer. Though his strengths are rooted in X++ development, he is a highly regarded developer and has knowledge of technical aspects of Dynamics AX development and customization. He has also worked on base product development with the Microsoft team.

He was awarded the Most Valuable Professional (MVP) on Dynamics AX in 2013 and 2014 by Microsoft.

He has also contributed to the following books:

- Microsoft Dynamics AX 2012 Reporting Cookbook
- Microsoft Dynamics AX 2012 Programming
- Microsoft System Centre Configuration Manager

Deepak shares his experience with Dynamics AX on his blog at http://theaxapta.blogspot.in/.
**Chhavi Aggarwal** started working on Dynamics AX in 2012. She is a Microsoft Certified Professional and has worked on both the 2009 and 2012 versions of Dynamics AX. She is an expert in SSRS reports and has also done a lot of customization and development in Dynamics AX R2/R3 through X++. She has a very deep knowledge of the technical aspects related to Dynamics AX R2/R3, as well as sound technical and logical skills in customization and development. She has also worked with the Microsoft team for standard base development.

**Kamalakannan Elangovan** has over 8 years of development experience in Dynamics AX. He shares a passion for product development and has pioneered multiple ISV solutions on Dynamics AX. In the past, he has worked with Innovites to create the first multidimensional ISV solution for cable industries called "InnoVites for cable". Building the solution from scratch, he gained great insights into building, selling, and promoting a product among customers and partners in the Microsoft Ecosystem. You can find out more about him at [http://about.me/casperkamal](http://about.me/casperkamal)

Kamal is enthusiastic about sharing his learnings with the community, which led him to create one of the first few blogs for AX in 2006. It is currently available at [http://kamalblogs.wordpress.com](http://kamalblogs.wordpress.com) He is active on Twitter and is very well known in the community by his pseudonym, "Casperkamal".
Microsoft Dynamics AX 2012 R3 Reporting Cookbook

Reporting provides consolidated, factual, and up-to-date information about any area of business in an organization. This will help the organization member to take the right decision for their business. It acts as a treasure trove of reliable information for long-term planning and decision making.

In Microsoft Dynamics R2, AX provides the tool to build the SSRS reports but in Microsoft Dynamics R3, the tool to build the SSRS reports in a more simple and efficient way. This book will give over 90 more recipes for beginners to understand the SSRS reports in Microsoft Dynamics AX R3 faster and in a simplified manner.

What This Book Covers

Chapter 1, Understanding and Creating Simple SSRS Reports, helps you to walk through the basis of SSRS reports and create a simple report using queries. This chapter will also help you understand the basic characteristics of reports.

Chapter 2, Enhancing Your Report – Visualization and Interaction, covers how to enhance the visualization and interaction of reports through parameters, creating data regions and charts, and formatting the reports.

Chapter 3, A Report Programming Model, helps you build the SSRS reports programmatically inside Dynamics AX and add the validations programmatically.

Chapter 4, Report Programming Model – RDP, helps you to develop the advanced reports using the data provider framework. In this chapter, we will create a report using RDP classes.

Chapter 5, Integrating External Datasources, covers how the SSRS reports can be extended to include other data sources or through the integration framework.

Chapter 6, Beyond Tabular Reports, helps you understand the other formats of reports such as using matrices and charts. This book also covers the interesting components of reports such as gauges, rectangles, and lists.

Chapter 7, Upgrading and Analyzing Reports, discusses the approaches for moving the MorphX-based reports into SSRS reports and also covers the patterns of reports.
Chapter 8, *Troubleshooting and Other Advanced Recipes*, dives into the performance and troubleshooting of SSRS reports. This will also introduce the log viewer through which you can analyze the report usage and its log.

Chapter 9, *Developing Reports with Complex Databases*, discusses developing the SSRS reports that have complex data and how to drive that data into reports using maps, views, and queries.

Chapter 10, *Unit Test Class and Best Practices Used for Reports*, helps you create the unit test classes for a report, which will help you test the business logic being written to develop reports. It will also tell you best practices to be followed when developing the reports.
This chapter will cover the following topics:

- Using a query as a data source in a report
- Creating auto designs from datasets
- Grouping in reports
- Adding ranges to the report
- Deploying a report
- Creating a menu item for a report
- Creating a report using a query in Warehouse Management

**Introduction**

Reports are a basic necessity for any business process, as they aid in making critical decisions by analyzing all the data together in a customized manner. Reports can be fetched in many types, such as ad-hoc, analytical, transactional, general statements, and many more by using images, pie charts, and many other graphical representations. These reports help the user to undertake required actions. Microsoft SQL Reporting Services (SSRS) is the basic primary reporting tool of Dynamics AX 2012 R2 and R3.
This chapter will help you to understand the development of SSRS reports in AX 2012 R3 by developing and designing reports using simple steps. These steps have further been detailed into simpler and smaller recipes. In this chapter, you will design a report using queries with simple formatting, and then deploy the report to the reporting server to make it available for the user. This is made easily accessible inside the rich client.

**Reporting overview**

**Microsoft SQL Server Reporting Services** (SSRS) is the most important feature of Dynamics AX 2012 R2 and R3 reporting. It is the best way to generate analytical, high user scale, transactional, and cost-effective reports. SSRS reports offer ease of customization of reports so that you can get what you want to see. SSRS provides a complete reporting platform that enables the development, design, deployment, and delivery of interactive reports. SSRS reports use Visual Studio (VS) to design and customize reports. They have extensive reporting capabilities and can easily be exported to Excel, Word, and PDF formats.

Dynamics AX 2012 has extensive reporting capabilities like Excel, Word, Power Pivot, Management Reporter, and most importantly, SSRS reports. While there are many methodologies to generate reports, SSRS remains the prominent way to generate analytical and transactional reports. SSRS reports were first seen integrated in AX 2009, and today, they have replaced the legacy reporting system in AX 2012.

SSRS reports can be developed using classes and queries. In this chapter, we will discuss query-based reports. Reports using classes will be discussed in later chapters.

In query-based reports, a query is used as the data source to fetch the data from Dynamics AX 2012 R3. We add the grouping and ranges in the query to filter the data. We use the auto design reporting feature to create a report, which is then deployed to the reporting server. After deploying the report, a menu item is attached to the report in Dynamics AX R3 so that the user can display the report from AX R3.

Through the recipes in this chapter, we will build a vendor master report. This report will list all the vendors under each vendor group. It will use the query data source to fetch data from Dynamics AX and subsequently create an auto design-based report. So that this report can be accessed from a rich client, it will then be deployed to the reporting server and attached to a menu item in AX.

Here are some important links to get started with this chapter:


Using a query as a data source in a report

Queries offer the simplest and easiest way to retrieve data for SSRS reports in Dynamics AX R3. They are very advantageous as they are reusable, and the same query can be used as the data source of another SSRS report in Dynamics AX R3. They are also very easy to design.

We can create queries in two ways: either by using the query class, or under the Queries node in Application Object Tree (AOT). In this recipe, we will create a query under the Queries node in AOT and use it as a data source in SSRS reports. Later on, we will guide you on how to add the query as a data source through Visual Studio.

Getting ready

To work through this recipe, you will require AX 2012 R2 or AX 2012 R3 rich clients with developer permission.

How to do it...

Create a new query named PKTVendorDetails under the Queries node in AOT, and add some fields in the query. Create a new SSRS report in Visual Studio 2010 and add the PKTVendorDetails query into that report as a data source.

1. Open the AX Development Workspace (Ctrl + D).
2. Go to AOT | Queries and add a new query.
3. Rename the query to PKTVendorDetails.
4. Go to query's data source node and add the new data source.
5. Rename the data source to VendTable and set property table to VendTable.
6. There are two steps to select fields from VendTable. You can use any of these:
   - Go to the Fields node under the VendTable data source and set Dynamic Property to Yes. This will automatically add all the fields in the VendTable to the query.
Understanding and Creating Simple SSRS Reports

- Drag and drop the required field directly from the table. Drag **VendGroup**, **AccountNum**, **InvoiceAccount**, and **Blocked** from **VendTable** as shown in the following screenshot. This is the best way to optimize the query and, consequently, reduce the fetch time, so we will go for this option in our recipes.

7. Save the query.
9. Navigate to **File | New | Project**.
10. In the new project dialog, select **Microsoft Dynamics AX**, and then **Report Model**.
11. Set the name as **PKTVendorDetailsReport**.
12. Now, right-click on project in Solution Explorer and Add a new Report to the PKTVendDetailReport project as shown in the following screenshot:

13. Rename the report as PKTVendorDetailsReport.
14. Now open the report by double-clicking on it in Solution Explorer.
15. Right-click on the Dataset node and select the New Dataset option.
16. Rename the dataset as VendorMaster.
17. Now, right-click on the VendorMaster dataset and select Properties.
18. Click on the ellipsis (…) button in Query.
19. Select the PKTVendorDetail query from the list and click on the Next button.

20. Select All Fields since we dropped all the unwanted fields during the creation of the query. From the All Display Methods node, select Name. Click on the OK button.
This will generate the fields list for the dataset. This completes the addition of a dataset to a report.

**How it works...**

In this receipe we have used queries as a datasource in SSRS report in Dynamics AX R3 as queries are reusable and can help to add the ranges in a report.

**Connecting VS to AX**

When creating a new report project in Visual Studio, if there is no option such as Microsoft Dynamics AX, then ensure that you have your reporting extensions installed. When you have multiple instances of Dynamics AX installed, Visual Studio identifies the instance to connect to from the client configuration. The active client configuration is used to establish the connection. The layer in which the report must be created is also fetched from the client configuration.
Retrieving metadata and data

With AX 2012, Windows Communication Foundation (WCF) based system services have been introduced. This includes the metadata service, query service, and user session service. The SSRS reporting extension uses the query and metadata services. The metadata service helps the report designer in Visual Studio to retrieve the metadata information of Queries, Tables, and Extended Data Types (EDT). The query service is used to fetch the data.

Verify the query

In the case of a complex query, a better approach would be validating the query before it is included in the report. Write a job in Dynamics AX that will use the query to retrieve the data and print the values to the infolog. This will help in identifying the problem when there is an issue with the report.

No joins

The report supports multiple datasets, but as in AX forms these datasets cannot be joined and they remain independent.

Creating auto designs from datasets

There are two ways to design an SSRS report in Visual Studio:

- **Precision design**: This is for advanced structured design
- **Auto design**: This is for general tabular design

In this recipe, we will create a simple auto design report by using the dataset added in the previous recipe, which will fetch the data and show it in the report.

In auto design, there are layouts that are already built in, and we can choose one in which we want to show the data, based on our preferences.

Getting ready

To develop reports in Dynamics AX 2012 R3, you need Visual Studio, through which you can design, develop, and deploy the reports. SQL Reporting Services must be properly installed and configured. You must also have access to the reporting manager to manage and see the reports present in AX 2012 R3.

How to do it...

In this recipe, we will add an auto design under the **Design** node of the report. We will then assign its layout properties to **ReportLayoutStyleTemplate** and print the report.
There are some standard report templates in AX. You can choose any of them for pre-designed layouts.

1. Right-click on the Designs node, select Add, and then select Auto Design. This will create a new auto design under the Design node. Rename it as VendorMaster.

2. In the VendorMaster properties, set the LayoutTemplate property to ReportLayoutStyleTemplate and set the Name property to VendorMaster.

3. Under the new auto design node, right-click on VendorMaster and select Add | Table. Set the properties for this table as shown in the following screenshot:
4. Notice that the fields are added to the table design automatically.

5. Right-click on Auto design (VendorMaster) and select Preview. This will show a preview of the report.

6. To rearrange the fields as per the user requirement, go to the Data node under the Table node. You can move fields in two ways, by:

   - Using the right-click menu options, such as Move to Top, Move Up, Move Down, Move to Bottom

   - Using shortcuts, like Alt + Up/Down arrow
The new format will look like the following screenshot:

![VendorMaster](image)

**How it works...**

Auto design is much easier to design and develop when compared to precision design. The default type (table/chart) for auto design is defined in the properties of the dataset. The default type determines what kind of control is added when the dataset is dragged and dropped into the auto design node.

**Templates:** Templates are responsible for printing the header, footer, and company name on a report. They also manage the font and colors. Currently, AX does not support printing the company image in the header through auto design.

**Report preview:** This accesses the default company in AX to show a report preview. So, ensure that the default company in AX has data, otherwise you may not find data in the preview.

> Standard SSRS reporting doesn't have the concept of auto design. This is only available in the AX SSRS implementation.

**Grouping in reports**

Grouping means putting things into groups. In the previous recipe, all the data shown in the report was listed sequentially. Grouping data simplifies the structure of the report and makes it more readable. It also helps you to find details, if required.
We can group the data in the query as well as in the auto design node in Visual Studio. In this recipe, we will structure the report by grouping the VendorMaster report based on the VendGroup to make the report more readable.

How to do it...

In this recipe, we will add fields under the grouping node of the dataset created earlier in Visual Studio. The fields that have been added in the grouping node will be added and shown automatically in the SSRS report.

1. Go to Dataset and select the VendGroup field.
2. Drag and drop it to the Groupings node under the VendorMaster auto design.

This will automatically create a new grouping node and add the VendGroup field to the group. Each grouping has a header row where even fields that don’t belong to the group but need to be displayed in the grouped node can be added.

This groups the record and also acts like a header, as seen in the following screenshot:
**How it works...**

Grouping can also be done based on multiple fields. Use the row header to specify the fields that must be displayed in the header. A grouping can be added manually but dragging and dropping prevents a lot of tasks such as setting the row header.

**Adding ranges to the report**

Ranges are very important and useful while developing an SSRS report in AX 2012 R3. They help to show only limited data, which is filtered based on given ranges, in the report.

The user can filter the data in a report on the basis of the field added as a range. The range must be specified in the query. In this recipe, we will show how we can filter the data and use a query field as a range.

**How to do it...**

In this recipe, we will add the field under the **Ranges** node in the query that we made in the previous recipe. By adding the field as a range, you can now filter the data on the basis of **VendGroup** and show only the limited data in the report.

1. Open the PKTVendorDetails query in AOT.
2. Drag the **VendGroup** and **Blocked** fields to the **Ranges** node in AOT and save your query.

3. In the Visual Studio project, right-click on **Datasets** and select **Refresh**.
4. Under the parameter node, VendorMaster_DynamicParameter collectively represents any parameter that will be added dynamically through the ranges. This parameter must be set to true to make additional ranges available during runtime. This adds a Select button to the report dialog, which the user can use to specify additional ranges other than what is added.

5. Right-click on the VendorMaster auto design and select Preview. The preview should display the range that was added in the query. Click on the Select button and set the VendGroup value to 10. Click on the OK button, and then select the Report tab, as shown in the following screenshot:

6. Save your changes and rebuild the report from Solution Explorer. Then, deploy the solution.
How it works...

The report dialog uses the query service UI builder (that we will cover in later chapters) to translate the ranges and to expose additional ranges through the query.

**Dynamic parameter**: The dynamic parameter unanimously represents all the parameters that are added at runtime. It adds the Select button to the dialog from where the user can invoke an advanced query filter window. From this filter window, more ranges and sorting can be added. The dynamic parameter is available per dataset and can be enabled or disabled by setting up the **Dynamic Filters** property to True or False.

The Report Wizard in AX 2012 still uses MorphX reports to auto-create reports using the wizard. The auto report option is available on every form that uses a new AX SSRS report.

Deploying a report

SSRS, being a server side solution, needs to deploy reports in Dynamics AX 2012 R3. Until the reports are deployed, the user will not be able to see them or the changes made in them, neither from Visual Studio nor from the Dynamics AX rich client.

Reports can be deployed in multiple ways and the developer must make this decision. In this recipe, we will show you how we can deploy reports using the following:

- Microsoft Dynamics AX R3
- Microsoft Visual Studio
- Microsoft PowerShell
Getting ready

In order to deploy reports, you must have the permission and rights to deploy them to SQL Reporting Services. You must also have the permission to access the reporting manager configuration.

Before deploying reports using Microsoft PowerShell, you must ensure that Windows PowerShell 2.0 is installed.

How to do it...

Microsoft Dynamics AX R3 supports the following ways to deploy SSRS reports.

Location of deployment

For each of the following deployment locations, let’s have a look at the steps that need to be followed:

1. Microsoft Dynamics AX R3:
   1. Reports can be deployed individually from a developer workspace in Microsoft Dynamics AX.
   2. SSRS reports can be deployed by using the developer client in Microsoft Dynamics AX R3.
   3. In AOT, expand the SSRS Reports node, expand the Reports node, select the particular report that needs to be deployed, expand the selected report node, right-click on the report, and then select and click on Deploy Element.

4. The developer can deploy as many reports as need to be deployed, but individually.
5. Reports can be deployed for all the translated languages.

2. Microsoft Visual Studio:
   1. Individual reports can be deployed using Visual Studio.
   2. Open Visual Studio. In Solution Explorer, right-click on the reporting project that contains the report that you want to deploy, and click on Deploy.

3. The reports are deployed for the neutral (invariant) language only.

3. Microsoft PowerShell:
   1. This is used to deploy the default reports that exist within Microsoft Dynamics AX R3.
   2. Open Windows PowerShell and by using this, you can deploy multiple reports at the same time.


4. To verify whether a report has been deployed, open the report manager in the browser and open the Dynamics AX folder. The PKTVendorDetails report should be found in the list of reports.

5. The report can be previewed from Reporting Services also. Open Reporting Services and click on the name of the report to preview it.
Understanding and Creating Simple SSRS Reports

How it works

Report deployment is the process of actually moving all the information related to a report to a central location, which is the server, from where it can be made available to the end user. The following list indicates the typical set of actions performed during deployment:

1. The RDL file is copied to the server.
2. The business logic is placed in the server location in the format of a DLL.

Deployment ensures that the RDL and business logic are cross-referenced to each other.

The MorphX IDE from AX 2009 is still available. Any custom reports that are designed can be imported. This support is only for the purpose of backward compatibility. In AX 2012 R3, there is no concept of MorphX reports.

Creating a menu item for a report

The final step of developing a report in AX 2012 R3 is creating a menu item inside AX to make it available for users to open from the UI end.

This recipe will tell you how to create a new menu item for a report and set the major properties for it. Also, it will teach you to add this menu item to a module to make it available for business users to access this report.

How to do it...

You can create the new menu item under the Menu Item node in AOT. In this recipe, the output menu item is created and linked with the menu item with SSRS report.

1. Go to AOT | Menu Items | Output, right-click and select New Menu Item. Name it PKTVendorMasterDetails and set the properties as highlighted in the following screenshot:
2. Open the **Menu Item** to run the report. A dialog appears with the **Vendor hold** and **Group** ranges added to the query, followed by a **Select** button. The **Select** button is similar to the MorphX reports option where the user can specify additional conditions. To disable the **Select** option, go to the **Dynamic Filter** property in the dataset of the query and set it to **False**.
Understanding and Creating Simple SSRS Reports

The report output should appear as seen in the following screenshot:

<table>
<thead>
<tr>
<th>VendorMaster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contoso Entertainment System India</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Vendor account</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpine ski Mouse India Ltd.</td>
<td>INMF-000001</td>
<td>10</td>
</tr>
<tr>
<td>Coho Vineyard India</td>
<td>INMF-000004</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Vendor account</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue Younder India Airlines Ltd.</td>
<td>INMF-000002</td>
<td>20</td>
</tr>
<tr>
<td>Litware India</td>
<td>INMF-000003</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Vendor account</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northwind Traders India Ltd.</td>
<td>INMF-000005</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Vendor account</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAT Authority (Kar)</td>
<td>INMF-000006</td>
<td>50</td>
</tr>
<tr>
<td>Customs Authority</td>
<td>INMF-000007</td>
<td>50</td>
</tr>
<tr>
<td>Excise Authority (Kar)</td>
<td>INMF-000008</td>
<td>50</td>
</tr>
<tr>
<td>Service Tax Authority (Kar)</td>
<td>INMF-000009</td>
<td>50</td>
</tr>
</tbody>
</table>

How it works...

The report viewer in Dynamics AX is actually a form with an embedded browser control. The browser constructs the report URL at runtime and navigates to the reports URL. Unlike in AX 2009, when the report is rendering, the data it doesn't hold up using AX. Instead, the user can use the other parts of the application while the report is rendering. This is particularly beneficial for the end users as they can proceed with other tasks as the report executes.

The permission setup is important as it helps in controlling the access to a report. However, SSRS reports inherit user permission from the AX setup itself.

Creating a report using a query in Warehouse Management

In Dynamics AX 2012 R3, Warehouse Management is a new module. In the earlier version of AX (2012 or R2), there was a single module for Inventory and Warehouse Management. However, in AX R3, there is a separate module.
Chapter 1

AX queries are the simplest and fastest way to create SSRS reports in Microsoft Dynamics AX R3. In this recipe, we will develop an SSRS report on Warehouse Management.

In AX R3, Warehouse Management is integrated with bar-coding devices such as RF-SMART, which supports purchase and receiving processes: picking, packing and shipping, transferring and stock counts, issuing materials for production orders, and reporting production as well. AX R3 also supports the workflow for the Warehouse Management module, which is used to optimize picking, packing, and loading of goods for delivery to customers.

Getting ready

To work through this recipe, Visual Studio must be installed on your system to design and deploy the report. You must have the permission to access all the rights of the reporting server, and reporting extensions must be installed.

How to do it...

Similar to other modules, Warehouse Management also has its tables with the "WHS" prefix. We start the recipe by creating a query, which consists of WHSRFMenuTable and WHSRFMenuLine as the data source. We will provide a range of Menus in the query. After creating a query, we will create an SSRS report in Visual Studio and use that query as the data source and will generate the report on warehouse management.

Open AOT, add a new query, and name it PKTWarehouseMobileDeviceMenuDetails.

1. Add a WHSRFMenuTable table.
2. Go to Fields and set the Dynamics property to Yes.
3. Add a WHSRFMenuLine table and set the Relation property to Yes. This will create an auto relation that will inherit from table relation node.

4. Go to Fields and set the Dynamics property to Yes.
Understanding and Creating Simple SSRS Reports

5. Now open Visual Studio and add a new Dynamics AX report model project. Name it PKTWarehouseMobileDeviceMenuDetails.

6. Add a new report to this project and name it PKTWarehouseMobileDeviceDetails.

7. Add a new dataset and name it MobileDeviceDetails.

8. Select the PKTWarehouseMobileDeviceMenuDetails query in the Dataset property.

9. Select all fields from both tables. Click on OK.

10. Now drag and drop this dataset in the design node. It will automatically create an auto design. Rename it MobileMenuDetails.

11. In the properties, set the layout property to ReportLayoutStyleTemplate.

12. Now preview your report.

![Report Screenshot]

How it works

When we start creating an SSRS report, VS must be connected with Microsoft Dynamics AX R3. If the Microsoft Dynamics AX option is visible in Visual Studio while creating the new project, then the reporting extensions are installed. Otherwise, we need to install the reporting extensions properly.
Where to buy this book

You can buy Microsoft Dynamics AX 2012 R3 Reporting Cookbook from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.