



Quick answers to common problems

Microsoft Dynamics NAV 7 Programming Cookbook

Learn to customize, integrate and administer NAV 7 using
practical, hands-on recipes

Rakesh Raul

[PACKT] enterprise 
PUBLISHING professional expertise distilled

Microsoft Dynamics NAV 7 Programming Cookbook

Learn to customize, integrate and administer NAV 7 using practical, hands-on recipes

Rakesh Raul



BIRMINGHAM - MUMBAI

Microsoft Dynamics NAV 7 Programming Cookbook

Copyright © 2013 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: September 2013

Production Reference: 1170913

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84968-910-6

www.packtpub.com

Cover Image by Jarek Blaminsky (milak6@wp.pl)

Credits

Author

Rakesh Raul

Reviewers

Danilo Capuano

Neil Murray

Giancarlo Zavala

Acquisition Editor

James Jones

Lead Technical Editor

Anila Vincent

Technical Editors

Veena Pagare

Jinesh Kampani

Iram Malik

Sandeep Madnaik

Shashank Desai

Copy Editors

Tanvi Gaitonde

Sayanee Mukherjee

Kirti Pai

Alfida Paiva

Adhiti Shetty

Project Coordinator

Navu Dhillon

Proofreader

Bridget Braund

Indexer

Monica Ajmera

Graphics

Abhinash Sahu

Production Coordinator

Arvindkumar Gupta

Cover Work

Arvindkumar Gupta

About the Author

Rakesh Raul is from a small town in India, with a vision of doing something big in programming. He completed his first diploma in programming at the age of 16, and continued higher studies in computer software development.

He started his programming career with a small software development company in Mumbai. After 2 years of development in Visual Basic, he was introduced to Microsoft Dynamics NAV Version 3. For the first 2-3 years he worked as a Microsoft Dynamics NAV developer and at the same time he learned all the areas of the product and earned his first Microsoft Certification-Business Solutions Professional. He continues to stay updated with new releases of the product and is certified in multiple areas for versions 4.0, 5.0, 2009, and 2013. Apart from Microsoft Dynamics NAV, he also has good knowledge of Microsoft SQL Server and Business Intelligence.

His seven-year journey with Microsoft Dynamics NAV includes more than 30 implementations; one horizontal and two vertical solution designs and development.

Currently, he works in Tectura, India, as a Senior Technical Consultant. Tectura is a worldwide provider of business consulting services delivering innovative solutions.

I would like to thank my wife, Ashwini, for supporting and always standing by my side in good and bad days.

I would like to take this opportunity to thank all the mentors I was blessed with, who unconditionally shared their knowledge and inspired me.

Mibuso and all Microsoft Dynamics NAV related blogs are a great boon for all NAV consultants. I would like to thank all the contributors of these great sites.

Love you Aabha, my cute little princess!

About the Reviewers

Danilo Capuano is a Software Engineer with over seven years of industry experience. He lives in Naples, Italy, where he earned a degree in computer science. He currently works as a developer on Microsoft Dynamics NAV in an IT company where he also completed the MCTS certification. You can visit his website: www.capuanodanilo.com or his Twitter account: @capuanodanilo.

Neil Murray began his development career as a C++ and Visual Basic developer, qualifying as a Microsoft Certified Solution Developer on Visual Basic 6.0 in 1999. He has been a Microsoft Dynamics NAV developer since 2001, providing consulting, customization, and support to customers across sub-Saharan Africa. He currently works for a large multi-national IT organization, providing technical and business process support to dairy, manufacturing, and retail clients.

I would like to thank my wife, Justine, and lovely daughters, Ember and Danica, for their love and understanding while I have dedicated precious family time to conduct the technical review of this book.

Giancarlo Zavala is an all-round expert in Microsoft Dynamics NAV technologies specializing in Application Analysis, Design and Development, and ERP implementations. He began as a network administrator in 1999 and eventually transitioned into application management. He also has a strong background with more than 15 years on Microsoft Server and networking technologies, Database Administration and Server Infrastructure deployment.

He has worked on a wide range of implementations and development projects in his career; including working as lead technical and functional consulting roles, as well as project management roles. He earned his first Microsoft Certification in SQL Database Administration in 2003, and later studied the Microsoft Dynamics ERP and CRM technologies.

He has now been managing business applications for over 10 years. He has built a unique set of skills working on full end-to-end implementations and application rollouts in various industries. He has helped various mid- to large-scale organizations successfully implement Microsoft Dynamics NAV in multiple countries around the globe, including Europe and Latin America.

Recently, he spent a couple of years in Houston working on the Oil and Energy industry with one of the largest NAV application setups around the globe. He worked on dozens of implementations as lead consultant, getting extensive knowledge of methodologies, business workflows, and best operational practices including Manufacturing, Distribution, Servicing, Warehouse Management, Intercompany Operations, Cost Accounting, and Financial Reporting amongst others. During this time, he also transitioned into design and development for Microsoft Dynamics NAV. He has worked with many of Microsoft's top partners and other well-known software vendors on multiple projects.

His passion is to always learn new skills and technologies related to Microsoft Dynamics, to create business specific solutions and to pass on his knowledge by training companies and coaching other colleagues. He enjoys building and working with a good team and taking on challenging projects with mission critical operations.

Currently, he lives in Miami and works at Tectura, U.S. as a Senior Consultant. He works on leading implementations, doing system analysis and in program development. Tectura is one of Microsoft's leading worldwide gold partners, providing consulting services and innovative solutions for small to large businesses.

Outside of work he enjoys travelling, surfing, and painting.

Acknowledgements

I want to thank everyone who helped me so that I could have the time and opportunity to work on this book. I've taken time away from the people I love in order to be able to accomplish this task. I wouldn't have been successful without the support from my family and friends who have always been there for me; and from my colleagues who have given me endless help and motivation. A very special thank you to my loving wife, Carolina, for her unconditional support on this journey. While I spent many nights working away from my loved ones, she took care of our two beautiful daughters on her own and made it all possible. I would be nowhere without her and I can only hope that in time I can repay her for the time I took away from us.

I would like to give my gratitude to Rakesh Raul (the author of this book), and Anila Vincent (Lead Technical Editor) for giving me the opportunity to work on this cookbook project.

Big thanks to my grandfather who gave me strong roots and good moral values to live a positive life filled with love. Much love to my mom and dad who I wish could be here today to see this. Big thanks to my Brother and Tia Magaly for believing in me always and showing me the right path. Thanks again to all my friends and family who have been with us for so many years and to those who have supported me throughout the review of this book. Thanks to my friend, Chuck Luciano, for always giving me moral support; thanks to Jan Verleur, John Byrne, David Diaz, Steve Bloch, Stacy Racca, Sowmya Sridhar, Brett Boullion, Lurleen Cloud, Kim Parker, Craig Sanders, and Elizabeth Peña for teaching me, keeping faith in my abilities, and giving me some of the most important opportunities in the early years of my career.

For every night that I have not been able to kiss you goodnight, this work is dedicated to the two most beautiful and brightest lights in my world, Sofia and Valentina.

Live, Learn, Create, Teach, and Love.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- ▶ Fully searchable across every book published by Packt
- ▶ Copy and paste, print and bookmark content
- ▶ On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Instant Updates on New Packt Books

Get notified! Find out when new books are published by following [@PacktEnterprise](#) on Twitter, or the *Packt Enterprise* Facebook page.

Table of Contents

Preface	1
Chapter 1: String, Dates, and Other Data Types	7
Introduction	7
Retrieving the system date and time	8
Retrieving the work date	9
Determining the day, month, and year from a given date	11
Using the date formula to calculate dates	13
Converting a value to a formatted string	15
Creating an array	17
Creating an option variable	19
Converting a string to another data type	21
Manipulating string contents	23
Chapter 2: General Development	27
Introduction	27
Displaying the progress bar and data in process	28
Repeating code using a loop	30
Checking for conditions using an IF statement	32
Using the CASE statement to test multiple conditions	34
Rounding decimal values	36
Creating functions	37
Passing parameters by reference	39
Referencing dynamic tables and fields	41
Using recursion	44
Chapter 3: Working with Tables, Records, and Queries	47
Introduction	48
Creating a table	49
Adding a key to a table	51
Retrieving data using the FIND and GET statements	52

Advanced filtering	55
Adding a FlowField	57
Creating a SumIndexField	59
Retrieving data from FlowField and SumIndexField	61
Using a temporary table	63
Retrieving data from other companies	65
Using a query to extract data	66
Creating a query to link three tables	69
Working with queries in C/AL	74
Chapter 4: Designing Pages	77
Introduction	77
Creating a page using a wizard	79
Using multiple options to run the page	84
Applying filters on the lookup page	86
Updating the subform page from a parent page	88
Creating a FactBox page	93
Creating a Queue page	95
Creating a Role Center page	99
Creating a wizard page	102
Displaying a .NET add-in on a page	107
Adding a chart to the page	112
Chapter 5: Report Design	117
Introduction	117
Creating an RDLC report	119
Using multiple options to run a report	123
Adding custom filters to the Request Page	124
Setting filters when report is loaded	128
Creating reports to process data	130
Creating a link from report to page	132
Creating a link from report to report	135
Adding totals on decimal field	136
Adding interactive sorting on reports	138
Creating a matrix report	140
Chapter 6: Diagnosing Code Problems	147
Introduction	147
Using the debugger	147
Setting breakpoints	154
Handling runtime errors	158
Using About This Page and About This Report	161
Finding errors while using NAS	164

Chapter 7: Roles and Security	167
Introduction	167
Assigning a role to a user	168
Creating a new role	170
Using the FILTERGROUP function	172
Using security filters	174
Applying security filter modes	176
Field-level security	177
Assigning permission to use the About This Page function	182
Killing a user session	185
Chapter 8: Leveraging Microsoft Office	187
Introduction	187
Sending data to Microsoft Word	188
Managing stylesheets	190
Sending an e-mail from NAV through SMTP	191
Exporting data using the Excel Buffer	193
Creating data connection from Excel to NAV	197
Showing data in Excel using PowerPivot	199
Creating an InfoPath form for the NAV data	204
Creating charts with Visio	208
Chapter 9: OS Interaction	213
Introduction	213
Using HYPERLINK to open external files	214
Working with environmental variables	216
Using SHELL to run external applications	220
Browsing for a file	221
Browsing for a folder	223
Checking file and folder access permissions	225
Querying the registry	228
Zipping folders and files within NAV	230
Chapter 10: Integration	233
Introduction	233
Sharing information through XMLports	234
Writing to and reading from a file using the C/AL code	238
Creating web services	239
Consuming web services	241
Sending data through FTP	244
Printing a report in a PDF, Excel, and Word format	246
Writing your own automation using C#	247
Using ADO to access outside data	249

Chapter 11: Working with the SQL Server	253
Introduction	253
Creating a basic SQL query	254
Understanding SIFT	256
Using the SQL profiler	259
Displaying data from a SQL view in NAV	262
Identifying Blocked and Blocking sessions from SQL	264
Setting up a backup plan	266
Maintaining the transaction logfiles	269
Chapter 12: NAV Server Administration	271
Introduction	271
Creating a NAV Server Instance	273
Configuring NAS to run Job Queue	276
Creating a user on NAV	278
Changing the NAV license	281
Creating a new database	284
Testing the NAV database	286
Index	289

Preface

Microsoft Dynamics NAV 7 is a product of the Microsoft Dynamics family. It's a business management solution that helps simplify and streamline business processes, such as finance, manufacturing, customer relationship management, supply chains, analytics, and electronic commerce for small and medium-sized enterprises. Microsoft Dynamics partners can have full access to the source code, which is very easy to customize. Learning NAV programming in NAV 7 will give a full inside view of the ERP system and open doors to many other exciting areas.

The Microsoft Dynamics NAV 7 Programming Cookbook will take you through interesting topics that span a wide range of areas, for example, integrating the NAV system with other software applications, such as Microsoft Office and creating reports to present information from multiple areas of the system. You will not only learn the basics of NAV programming, but you will also be exposed to the technologies that surround the NAV system, such as .NET programming, SQL Server, and NAV system administration.

The first half of the cookbook will help programmers using NAV for the first time by walking them through the building blocks of writing code and creating objects, such as tables, pages, and reports.

The second half focuses on using the technologies surrounding NAV to build better solutions and administration of the NAV service tier. You will learn how to write .NET code that works with the NAV system and how to integrate the system with other software applications, such as Microsoft Office or even custom programs.

What this book covers

Chapter 1, String, Dates, and Other Data Types, describes the method of working with the most common data types. You will learn how to use the functions related to data types. Every recipe includes actual NAV code with a brief explanation about code that will make the data type learning process very interesting.

Chapter 2, General Development, covers the C/AL development structure that includes loops, conditional statements, functions, and so on. You will find some recipes describing C/AL specific commands and functions.

Chapter 3, Working with Tables, Records, and Queries, focuses on the database structure and data retrieval. You will learn how to design a table using filters to retrieve specific data. This chapter will also discuss new object type Query.

Chapter 4, Designing Pages, focuses on data presentation using pages. You will learn how to develop different types of pages including Role Center, Queue, wizard, and many more.

Chapter 5, Report Design, explains how to design an RDLC report. You will find recipes describing the process of adding a request page, setting filters, linking two reports and many more interesting topics related to reports.

Chapter 6, Diagnosing Code Problems, explains how to use built-in tools to debug code problems. You will also learn about debugging the NAV application server.

Chapter 7, Roles and Security, focuses on NAV user security, which includes creating roles and assigning permissions to a role. It will also explain about security filters and filter groups.

Chapter 8, Leveraging Microsoft Office, describes different methods to integrate with the Microsoft Office suite, which includes Word, Excel, InfoPath, and Visio.

Chapter 9, OS Interaction, focuses on different ways to integrate with the Windows operating systems. You will learn how to search the filesystem as well as how to query the system registry.

Chapter 10, Integration, describes different ways of integrating NAV with other applications. You will learn how to exchange data using flat file and XMLport. You will find a recipe describing how to use ADO to access data stored in other databases.

Chapter 11, Working with the SQL Server, provides an introduction to the SQL Server environment. You will learn about writing queries, configuring automated backups, and maintaining SQL logfiles. There is a recipe that will help you to understand the Sum Index Field Technology.

Chapter 12, NAV Server Administration, will help you to learn and understand the NAV service tier. It will also explain about creating a user and maintaining a NAV license.

What you need for this book

The following software are required for the recipes in this book:

- ▶ Microsoft Dynamics NAV 7 with developer license
- ▶ Microsoft SQL Server 2008 R2
- ▶ Microsoft Visual Studio 2010
- ▶ Microsoft Office 2010

Who this book is for

If you are an entry-level NAV developer, then the first half of the book is designed primarily for you. You may or may not have any experience in programming. It focuses on the basics of NAV programming. It would be best if you have already gone through a brief introduction to the NAV client.

If you are a mid-level NAV developer, you will find the second half more useful. These chapters explain how to think outside the NAV box when building solutions. Towards the end of the book, we will learn NAV server tier configuration.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "The `sp_who` command returns a list of all connections to the server by querying the `sys.sysprocesses` system table."


A block of code is set as follows:


```
Customer.RESET;
IF Customer.FINDSET THEN
    REPEAT
        CustCount:=CustCount+1;
    UNTIL Customer.NEXT=0;
    MESSAGE('There are %1 customers in the database',
        CustCount);
```

Any command-line input or output is written as follows:

```
sn.exe -T "C:\Program Files (x86)\Microsoft Dynamics NAV\70\
RoleTailored Client\Add-ins\NAV_RSS.dll"
```


New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "From the Tools menu in the NAV client select **Debugger** | **Debug Session** (Shift + Ctrl + F11)".

 Warnings or important notes appear in a box like this.

 Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to feedback@packtpub.com, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

1

String, Dates, and Other Data Types

In this chapter, we will cover the following recipes:

- ▶ Retrieving the system date and time
- ▶ Retrieving the work date
- ▶ Determining the day, month, and year from a given date
- ▶ Using the date formula to calculate dates
- ▶ Converting a value to a formatted string
- ▶ Creating an array
- ▶ Creating an option variable
- ▶ Converting a string to another data type
- ▶ Manipulating string contents

Introduction

Data types are the base component in **C/AL (Client/server Application Language)** programming. Most of the data types are equivalent to the data types used in other programming language. Boolean, integer, decimal, dates, and strings are the most used data types in C/AL programming.

As developers, our job is to build a business tool that will manipulate the data input by users and make sure that data stored in tables is meaningful. Most of this data will be of the decimal, string, and date data types. NAV is, after all, a financial system at heart. At its most basic level, it cares about three things: "How much money?" (decimal), "What was it used for?" (string), and "When was it used?" (date).

The recipes in this chapter are very basic, but they will help you to understand the basics of C/AL coding. All recipes are accompanied by actual C/AL code from NAV objects.

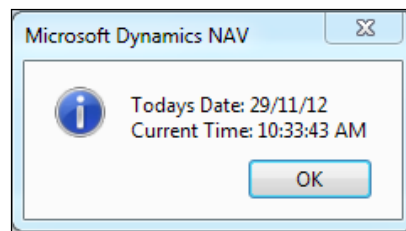
Retrieving the system date and time

Most times, we need to capture the system date and time of users' actions on NAV. This recipe will illustrate how to get the system date and time.

How to do it...

1. Let's create a new codeunit from **Object Designer**.
2. Now add the following code into the OnRun trigger of the codeunit:

```
MESSAGE('Today's Date: %1\Current Time: %2', TODAY, TIME);
```
3. To complete the development of the codeunit, save and close it.
4. On executing the codeunit, you should see a window similar to the one in the following screenshot:



How it works...

The `TODAY` keyword returns the date and the `TIME` keyword returns the time from the NAV Server system.

In the case of the older version of the NAV client—specifically the classic client—the date and time are taken from the client computer, which allows users to manipulate the system clock as per their personal requirement.

You can also retrieve the system date and time all at once using the `CURRENTDATETIME` function. The date and time can be extracted using the `DT2DATE` and `DT2TIME` functions respectively.



For a complete list of date functions, run a search for the `date` function and the `time` function in the **Developer and IT Pro Help** option in the **Help** menu of Microsoft NAV Development Environment

There's more...

The change log is a base NAV module that allows you to track changes to specific fields in tables. The following code can be found in the 423, Change Log Management codeunit in the `InsertLogEntry()` method:

```
ChangeLogEntry.INIT;
ChangeLogEntry."Date and Time" := CURRENTDATETIME;
ChangeLogEntry.Time := DT2TIME(ChangeLogEntry."Date and Time");
```

Here, instead of using the `WORKDATE` function, we use the `CURRENTDATETIME` function and then extract the time using the `DT2TIME` function. The system designers can just do the following setup:

```
ChangeLogEntry.Date := TODAY;
ChangeLogEntry.Time := TIME;
```

The advantage of using `CURRENTDATETIME` over `TODAY` and `TIME` is minimal. `CURRENTDATETIME` makes one request to the system while the second method makes two. It is possible that another operation or thread on the client machine could take over between retrieving the date and time from the computer; however, this is very unlikely. The operations could also take place right before and after midnight, generating some very strange data. The requirements for your modification will determine which method is best suited, but generally `CURRENTDATETIME` is the correct method to use.

See also

- ▶ *Retrieving the work date*
- ▶ *Determining the day, month, and year from a given date*
- ▶ *Converting a value to a formatted string*

Retrieving the work date

To perform tasks such as completing transactions for a date that is not the current date, you may have to temporarily change the work date. This recipe will show you how to determine what that actual work date is as well as when and where you should use it.

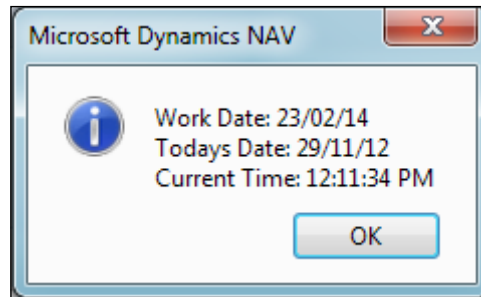
Getting ready

1. Navigate to **Application Menu | Set Work Date** or select the date in the status bar at the bottom of Microsoft Dynamics NAV.
2. Input the work date in the **Work Date** field or select it from the calendar.

How to do it...

1. Let's get started by creating a new codeunit from **Object Designer**.
2. Then add the following code into the OnRun trigger of the codeunit:

```
MESSAGE('Work Date: %1\Todays Date: %2\Current Time: %3',WORKDATE,  
TODAY, TIME);
```
3. To complete the task, save and close the codeunit.
4. On executing the codeunit, you should see a window similar to the following screenshot:



How it works...

To understand `WORKDATE`, we have used two more keywords in this recipe. The work date is a date internal to the NAV system. This date is returned using the `WORKDATE` keyword. It can be changed at any time by the user. The next date is `TODAY`; it's a keyword to retrieve the present date that provides the date from the system. In the end, we used the `TIME` keyword, which provides current time information from the system clock.



It is important to understand the difference between the NAV work date and the computer system date; they should be used in specific circumstances. When performing general work in the system, you should almost always use the `WORKDATE` keyword. In cases where you need to log information and the exact date or time when an action occurred, you should use `TODAY` or `TIME`, or `CURRENTDATETIME`.

There's more...

The following code can be found in the 38, Purchase Header table, in the `UpdateCurrencyFactor()` method:

```
IF "Posting Date" <> 0D THEN
    CurrencyDate := "Posting Date"
ELSE
    CurrencyDate := WORKDATE;
```

Looking at this code snippet, we can see that if a user has not provided any specific posting date, the system will assign the value `WORKDATE` as the default value for the posting date.

See also

- ▶ *Determining the day, month, and year from a given date*
- ▶ *Converting a value to a formatted string*
- ▶ *The Checking for conditions using an IF statement recipe in Chapter 2, General Development*
- ▶ *The Using the CASE statement to test multiple conditions recipe in Chapter 2, General Development*

Determining the day, month, and year from a given date

Sometimes it is necessary to retrieve only part of a date. NAV has built-in functions to do just that. We will show you how to use them in this recipe.

How to do it...

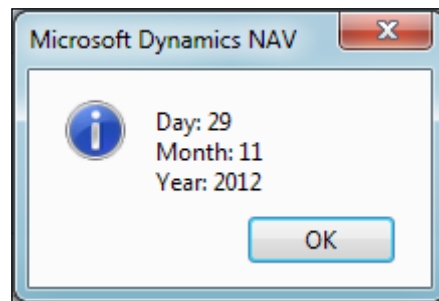
1. Let's create a new codeunit from **Object Designer**.
2. Then add the following global variables by navigating to **View | C/AL Globals** (*Alt + V + B*):

Name	Type
Day	Integer
Month	Integer
Year	Integer

3. Write the following code into the OnRun trigger of the codeunit:

```
Day := DATE2DMY(TODAY, 1);  
Month := DATE2DMY(TODAY, 2);  
Year := DATE2DMY(TODAY, 3);  
MESSAGE('Day: %1\Month: %2\Year: %3', Day, Month, Year);
```

4. To complete the task, save and close the codeunit.
5. On executing the codeunit, you should see a window similar to the following screenshot:



How it works...

The Date2DMY function is a basic feature of NAV. The first parameter is a date variable. This parameter can be retrieved from the system using `TODAY` or `WORKDATE`. Additionally, a hardcoded date such as `01312010D` or a field from a table, such as `Sales Header` or `Order Date` can be used as a first parameter. The second parameter is an integer that tells the function which part of the date to return. This number can be 1, 2, or 3, and corresponds to the day, month, and year (DMY) respectively.



NAV has a similar function called `Date2DWY`. It will return the week of the year instead of the month if 2 is passed as the second parameter.

There's more...

The following code can be found in the 485, `Business Chart Buffer` table in the `UpdateCurrencyFactor()` method of the `GetNumberOfYears()` function:

```
EXIT(DATE2DMY(ToDate, 3) - DATE2DMY(FromDate, 3));
```

This function has two parameters of type date and it returns the value in integer. The basic usage of this function is to calculate the duration between two dates in terms of years.

See also

- ▶ *Retrieving the system date and time*
- ▶ *Retrieving the work date*
- ▶ *The Repeating code using a loop recipe in Chapter 2, General Development*
- ▶ *The Checking for conditions using an IF statement recipe in Chapter 2, General Development*

Using the date formula to calculate dates

The date formula allows us to determine a new date based on a reference date. This recipe will show you how to use the built-in `CALCDATE` NAV function for date calculations.

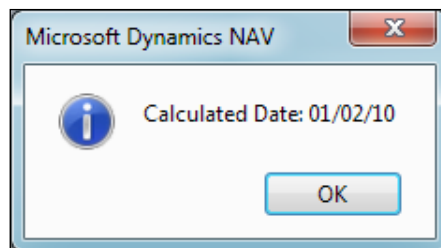
How to do it...

1. Let's start by creating a new codeunit from **Object Designer**.
2. Add the following global variable by navigating to **View | C/AL Globals (Alt + V + B)**:

Name	Type
CalculatedDate	Date

3. Write the following code into the OnRun trigger of the codeunit:


```
CalculatedDate := CALCDATE('CM+1D', 010110D);
MESSAGE('Calculated Date: %1', CalculatedDate);
```
4. Now save and close the codeunit.
5. On executing the codeunit, you should see a window similar to the following screenshot:



How it works...

The `CALCDATE()` function takes in two parameters: a calculation formula and a starting date. The calculation formula is a string that tells the function how to calculate the new date. The second parameter tells the function which date it should start with. A new date is returned by this function, so the value must be assigned to a variable.

The following units can be used in the calculation formula:

Unit	Description
D	Day
WD	Weekday
W	Week
M	Month
Q	Quarter
Y	Year

These units may be different depending on what language version NAV is running under.

You have two options to place the number before the unit. It can either be a standard number ranging between 1 and 9 or the letter C, which stands for current. These units can be added and subtracted to determine a new date based on any starting date.

Calculation formulas can become very complex. The best way to fully understand them is to write your own formulas to see the results. Start out with basic formulas such as `1M + 2W - 1D` and move on to more complex ones, such as `-CY + 2Q - 1W`.

There's more...

The following code is part of the `CalcNumberOfPeriods()` function of the 485, Business Chart Buffer table:

```
"Period Length"::Week:
    NumberOfPeriods := (CALCDATE('<-CW>',ToDate) -
        CALCDATE('<CW>',FromDate)) DIV 7;
```

The preceding code snippet will return the difference between two dates in terms of weeks. `<-CW>` will provide a week start date of `ToDate` whereas `<CW>` will provide a week end day of `FromDate`. The difference between the calculated days will be divided by 7 to get the total number of weeks.

For more details on `CALCDATE`, visit the following URL:

[http://msdn.microsoft.com/en-us/library/dd301368\(v=nav.70\).aspx](http://msdn.microsoft.com/en-us/library/dd301368(v=nav.70).aspx)

See also

- ▶ *Retrieving the system date and time*
- ▶ *Retrieving the work date*
- ▶ *Determining the day, month, and year from a given date*
- ▶ The *Checking for conditions using an IF statement* recipe in *Chapter 2, General Development*

Converting a value to a formatted string

There will be many occasions when you will need to display information in a certain way or multiple variable types on a single line. The `FORMAT` function will help you change almost any data type into a string that can be manipulated in any way you see fit.

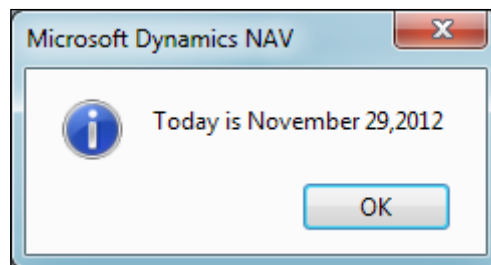
How to do it...

1. Let's get started by creating a new codeunit from **Object Designer**.
2. Then add the following global variable:

Name	Type	Length
FormattedDate	Text	30

3. Now write the following code into the `OnRun` trigger of the codeunit:


```
FormattedDate := FORMAT(TODAY, 0, '<Month Text> <Day,2>,<Year4>');
MESSAGE('Today is %1', FormattedDate);
```
4. To complete the task, save and close the codeunit.
5. On executing the codeunit, you should see a window similar to the following screenshot:



How it works...

The `FORMAT` function takes one to three parameters. The first parameter is required and can be of almost any type: date, time, integer, decimal, and so on. This parameter is returned as a string.

The second parameter is the length of the string to be returned. The default, zero, means that the entire string will be returned, a positive number tells the function to return a string of exactly that length, and a negative number returns a string not larger than that length.

There are two options for the third, and final, parameter. One is a number, representing a predefined format you want to use for the string, and the other is a literal string. In the example, we used the actual format string. The text contained in the angular brackets (< >) will be parsed and replaced with the data in the first parameter.



There are many predefined formats for dates. Run a search for `Format Property` in the **Developer and IT Pro Help** option in the **Help** menu of Microsoft NAV Development Environment or visit the following URL:

[http://msdn.microsoft.com/en-us/library/dd301059\(v=nav.70\).aspx](http://msdn.microsoft.com/en-us/library/dd301059(v=nav.70).aspx)

There's more...

The following code can be found on the `OnValidate()` trigger of the `Starting Date` field from the `50, Accounting Period` table:

```
Name := FORMAT("Starting Date",0,Text000);
```

In the preceding code, `Text000` is a text constant and carries the <Month Text> value. This code will return month of "Starting Date" in text format.

See also

- ▶ *Retrieving the system date and time*
- ▶ *Retrieving the work date*
- ▶ *Determining the day, month, and year from a given date*
- ▶ *Converting a string to another data type*
- ▶ *The Checking for conditions using an IF statement recipe in Chapter 2, General Development*
- ▶ *The Advanced filtering recipe in Chapter 3, Working with Tables, Records, and Queries*
- ▶ *The Retrieving data using the FIND and GET statements recipe in Chapter 3, Working with Tables, Records, and Queries*

Creating an array

Creating multiple variables to store related information can be time consuming. It leads to more code and more work. Using an array to store related and similar types of information can speed up development and lead to much more manageable code. This recipe will show you how to create and access array elements.

How to do it...

1. Let's create a new codeunit from **Object Designer**.
2. Add the following global variables by navigating to **View | C/AL Globals** (*Alt + V + B*):

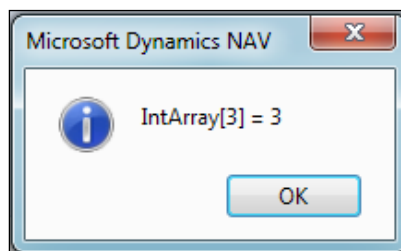
Name	Type
i	Integer
IntArray	Integer

3. Now, with the cursor on the IntArray variable, navigate to **View | Properties** (*Shift + F4*).
4. In the **Property** window, set the following property:

Property	Value
Dimensions	10

5. Write the following code into the OnRun trigger of the codeunit:


```
FOR i := 1 TO ARRAYLEN(IntArray) DO BEGIN
    IntArray[i] := i;
    MESSAGE('IntArray[%1] = %2', i, IntArray[i]);
END;
```
6. To complete the task, save and close the codeunit.
7. On executing the codeunit, you should see a window similar to the following screenshot:



How it works...

An array is a single variable that holds multiple values. The values are accessed using an integer index. The index is passed within square brackets ([]).



NAV provides several functions to work with arrays. For instance, `ARRAYLEN` returns the number of dimensions of the array and `COPYARRAY` will copy all of the values from one array into a new array variable. You can find a complete list of the array functions in the **Developer and IT Pro Help** option in the **Help** menu of Microsoft NAV Development Environment.

There's more...

Open the 365, `Format Address` codeunit. Notice that the first function, `FormatAddr`, has a parameter that is an array. This is the basic function that all of the address formats use. It is rather long, so we will discuss only a few parts of it here.

This first section determines how the address should be presented based on the country of the user. Variables are initialized depending on which line of the address should carry certain information. These variables will be the indexes of our array.

```
CASE Country."Contact Address Format" OF
  Country."Contact Address Format"::First:
    BEGIN
      NameLineNo := 2;
      Name2LineNo := 3;
      ContLineNo := 1;
      AddrLineNo := 4;
      Addr2LineNo := 5;
      PostCodeCityLineNo := 6;
      CountyLineNo := 7;
      CountryLineNo := 8;
    END;
```

Then we will fill in the array values in the following manner:

```
AddrArray[NameLineNo] := Name;
AddrArray[Name2LineNo] := Name2;
AddrArray[AddrLineNo] := Addr;
AddrArray[Addr2LineNo] := Addr2;
```

Scroll down and take a look at all of the other functions. You'll see that they all take in an array as the first parameter. It is always a text array of length 90 with eight dimensions. These are the functions you will call when you want to format an address. To use this codeunit correctly, we will need to create an empty array with the specifications listed before and pass it to the correct function. Our array will be populated with the appropriately formatted address data.

See also

- ▶ *Manipulating string contents*
- ▶ The *Using the CASE statement to test multiple conditions* recipe in *Chapter 2, General Development*

Creating an option variable

If you need to force the user to select a value from a predefined list, an **option** is the way to go. This recipe explains how to create an `Option` variable and access each of its values.

How to do it...

1. Let's create a new codeunit from **Object Designer**.
2. Then add the following global variable:

Name	Type
ColorOption	Option

3. With the cursor on the `ColorOption` variable, navigate to **View | Properties** or (*Shift + F4*).
4. In the **Property** window, set the following property:

Property	Value
OptionString	None,Red,Green,Blue

5. Now write the following code into the `OnRun` trigger of the codeunit:

```
ColorOption := ColorOption::Green;
CASE ColorOption OF
    ColorOption::None: MESSAGE('No Color Selected');
    ColorOption::Red: MESSAGE('Red');
    ColorOption::Green: MESSAGE('Green');
    ColorOption::Blue: MESSAGE('Blue');
END;
```

6. Save and close the codeunit.