Pro WF

Windows Workflow in .NET 3.0

Bruce Bukovics

Apress®

Pro WF: Windows Workflow in .NET 3.0

ISBN-13 (pbk): 978-1-59059-778-1

ISBN-10 (pbk): 1-59059-778-8

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Ewan Buckingham
Technical Reviewer: Sylvain Groulx
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser, Matt Wade
Project Manager: Beth Christmas
Copy Edit Manager: Nicole Flores
Copy Editor: Jennifer Whipple
Assistant Production Director: Kari Brooks-Copony
Production Editor: Katie Stence
Compositor: Susan Glinert
Proofreader: Nancy Riddiough
Indexer: Julie Grady
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

The source code for this book is available to readers at http://www.apress.com in the Source Code/Download section. You will need to answer questions pertaining to this book in order to successfully download the code.

For Teresa

Contents at a Glance

Contents

About the Author

BRUCE BUKOVICS has been a working developer for more than 25 years. During this time, he has designed and developed applications in such widely varying areas as banking, corporate finance, credit card processing, payroll processing, and retail systems.

He has firsthand developer experience with a variety of languages, including C, C++, Delphi, Java, VB, and C#. His design and development experience includes everything from mainframe and client/server to widely distributed n-tier and SOA applications. Most recently, he has been immersed in the .NET 3.0 technology stack, leveraging Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF), and, of course, Windows Workflow Foundation (WF) to help build a new generation of applications.

He considers himself a pragmatic programmer and test-driven development evangelist. He doesn't stand on formality and doesn't do things in a particular way just because they have always been done that way. He's willing to look at alternate or unorthodox solutions to a problem if that's what it takes.

He is currently employed at Radiant Systems Inc. in Alpharetta, Georgia, as a lead developer and architect in the central technology group.

About the Technical Reviewer

SYLVAIN GROULX is an independent database software consultant based in Montreal, Quebec. He's been an application developer and DBA over the past 20 years. His current interests include the .NET Framework 3.0 Windows Presentation Foundation, Windows Workflow Foundation, and SQL Server 2005 Services (Broker, Integration, Notification, and Reporting).

As the founder of the Microsoft .NET Architecture User Group in Montreal, he has been an active proponent of .NET technologies and community-based learning initiatives. He is a great enthusiast for C#, which is why he has been actively involved with .NET since the first bits were released in 2000.

He enjoyed many great years as a Microsoft MVP before joining Microsoft Consulting Services. His past roles at Microsoft include project lead and developer consultant. He also spent many years architecting and building custom applications for large enterprise customers.

When not sitting in front of a keyboard, Sylvain is busy playing a round of golf whenever possible.

Acknowledgments

Once again, the folks at Apress have done an outstanding job with this book. It takes a team of very professional people to produce a book like this. You truly made the task of writing this book an enjoyable one. A big thank-you goes out to all of you.

I would like to personally thank Ewan Buckingham who was the editor of this book. Ewan allowed me to write the book that I wanted to write, but he was also there with guidance when I needed it most. Matt Moodie also did some pinch-hitting for Ewan when things got backed up. Jennifer Whipple was the primary copy editor on the book, along with some assistance from Nicole Flores. They both did a great job correcting my prose, refining it without changing my original intent. Katie Stence was the production editor who worked her magic to transform my raw text into the pages that you see before you. And Beth Christmas was the project manager for the book. She was the traffic cop and task master who kept things running smoothly and on schedule.

A good technical book needs a great technical reviewer, and Sylvain Groulx stepped up to the challenge. This was an especially challenging job for Sylvain because I began this book while WF was in a constant state of change, churning with what seemed like monthly CTPs or betas. Yet in spite of this, he was able to keep up with all of the class and namespace changes from one version of WF to the next. Thank you, Sylvain, for making sure that this book is as accurate as possible.

I'd also like to thank all of the folks who contributed to the Microsoft Windows Workflow Foundation forum hosted on MSDN. This active community of developers was an invaluable resource in understanding how some of the undocumented features of WF worked during the early beta and CTP stage.

Most importantly, I must thank my wife, Teresa, and son, Brennen, for their support and patience during this project. When I finished my first book, I honestly thought that it would be a long time before I wrote my second. Little did I know that I was about to embark on this little project without any downtime at all between books. Sorry about that. I love both of you very much, and I'm looking forward to having time to spend with you again. I promise: No more books for a while.

Introduction

I started working with the new Microsoft WinFX technology stack early in the beta and CTP (Community Technology Preview) stage. The foundations in WinFX (Windows Presentation, Windows Communication, and Windows Workflow) have now finally made their way into a shipping Microsoft product: .NET 3.0. I actually started to learn and use all three of these foundations at the same time in my day job. Talk about a massive learning curve.

While I was impressed with the flexibility and capabilities of Windows Presentation Foundation and Windows Communication Foundation, I was somehow inexplicably drawn to Windows Workflow Foundation (WF). WF isn't just a new way to implement a user interface, or a new way to communicate between applications and services. WF represents a completely new way to develop applications. It is declarative, visual, and infinitely flexible. It promotes a model that cleanly separates *what* to do from *when* to do it. This separation allows you to change the workflow model (the *when*) without affecting the *what*. Business logic is implemented as a set of discrete, testable components that are assembled into workflows like building blocks.

Workflow isn't a new concept. But when Microsoft spends years developing a workflow foundation and provides it to us without cost, it is an event worth noting. Other workflow frameworks exist, but WF will soon become the de facto standard workflow framework for Windows applications.

I wrote this book because I'm excited about workflow, and WF in particular. I'm excited about the opportunities that it holds for application developers like us. My hope is that this book will help you to use WF to build an exciting new generation of workflow-enabled applications.

Who Should Read This Book

This book is for all .NET developers who want to learn how to use Windows Workflow Foundation in their own applications. This book is not a primer on .NET or the C# language. To get the most out of the examples that I present in this book, you need a good working knowledge of .NET 1.1 and preferably .NET 2.0. All of the examples are presented in C#, so you should be proficient with C#.

An Overview of This Book

The material in this book is a WF tutorial presented in 17 chapters, with each chapter building upon the ones before it. I've tried to organize the material so that you don't have to jump ahead in order to understand how something works. But since the chapters build upon each other, I do assume that you have read each chapter in order and understand the material that has already been presented.

The short sections that follow provide a brief summary of each chapter.

Chapter 1: A Quick Tour of Windows Workflow Foundation

This chapter provides a brief introduction to WF. In this chapter, you jump right in and develop your first workflow ("Hello Workflow"). You are introduced to some of the fundamental concepts of WF, such as how to pass parameters to a workflow and how to make decisions within a workflow.

Chapter 2: Foundation Overview

The goal of this chapter is to provide a high-level overview of WF in its entirety. This chapter doesn't teach you how to use each individual WF feature, but it does acquaint you with the design time and runtime features that are available with WF. This chapter is a roadmap for the material that is covered in the remainder of the book.

Chapter 3: Activities

Activities are the building blocks of WF and are used to construct complete workflows. This chapter provides a summary of the standard activities that are distributed with WF. This chapter also contrasts two ways to introduce business logic into a workflow: the `CodeActivity` and building your own custom activities.

Chapter 4: Hosting the Workflow Runtime

WF is not a stand-alone application. It is a framework for building your own workflow-enabled applications. This chapter demonstrates how to host the workflow runtime in your own application. Included in this chapter is a set of custom workflow manager classes that assist with hosting of the workflow runtime. These helper classes are used in most of the chapters that follow this one.

Chapter 5: Flow Control

WF includes a rich set of standard activities that support everything from simple branching decisions and while loops to parallel execution and replication. These flow control activities control the execution sequence within a workflow and are covered in this chapter. Most of these activities support `Boolean` conditions that can be specified in code or as declarative rule conditions. These two types of conditions are contrasted in this chapter.

Chapter 6: Local Services

Several core features of WF are implemented as pluggable services. This allows you to choose the implementation of each service that meets your needs. WF also provides for local services that can be implemented by you to serve any purpose. One common use of local services is to facilitate communication between workflow instances and your host application. The focus of this chapter is on implementing and using your own local services.

Chapter 7: Event-Driven Activities

This chapter covers event-driven activities that allow your workflows to wait for the receipt of an external event. Chapter 6 shows you how to implement local services and invoke methods of those services from workflow instances. This chapter demonstrates how to raise events from those local services and handle the events within a workflow.

Chapter 8: Workflow Persistence

Workflow persistence allows you to automatically save the state of running workflow instances and then reload them at a later time. The use of persistence is especially important for long-running workflows where a workflow can be unloaded from memory while it is idle and waiting for an external event.

Chapter 9: State Machine Workflows

WF supports two main types of workflows: sequential and state machine. Up until this point in the book, you have been working with sequential workflows that target system interaction problems. Sequential workflows are best used when the exact sequence of tasks is known at design time. State machine workflows are the focus of this chapter and are designed to easily react to external events. They are especially useful for problems that involve human interaction since the exact sequence of tasks can't be determined at design time.

Chapter 10: Transactions and Compensation

The goal of this chapter is to demonstrate two ways to control the integrity and consistency of work that is performed by a workflow. Transactions allow you to enlist multiple activities into a single logical unit of work. When transactions are used, all of the work is committed or rolled back together without any partial updates. On the other hand, compensation is the process of undoing work that has already completed. Compensation might be necessary if individual activities in a workflow have completed successfully, but later the workflow determines that the work must be undone.

Chapter 11: Workflow Rules

WF includes a general-purpose rules engine that you can also use as an alternate way to declare your business logic. Rules are best thought of as simple statements or assertions about data and not as procedural instructions. Individual rules are grouped into rule sets and are evaluated by the rules engine that is included with WF. Each rule allows you to define the actions to execute when the rule evaluates to `true`, and a separate set of actions when it is `false`.

Chapter 12: Exception and Error Handling

Exception handling is important in any application, and WF provides a way to declaratively handle exceptions. The goal of this chapter is to demonstrate various ways to handle exceptions within the workflow model. This chapter also covers cancellation handlers that are used to execute a set of activities when an executing activity is canceled.

Chapter 13: Dynamic Workflow Updates

Most of the time, you will statically define a workflow and then create instances of it at runtime. WF also provides the ability to dynamically apply updates to an executing workflow, altering the internal structure of the workflow. This chapter demonstrates how to apply dynamic workflow updates from the host application, as well as from within an executing workflow.

Chapter 14: Workflow Tracking

WF provides an instrumentation framework for tracking the execution of each workflow. The tracking framework supports pluggable tracking services that you can implement to meet your needs. The framework is based on tracking profiles that allow you to customize the amount and type of data tracked for each workflow type. The focus of this chapter is using the standard tracking service and also developing your own custom tracking service.

Chapter 15: Web Services and ASP.NET

WF allows you to declaratively access web services from within a workflow. You can also expose a workflow as a web service that can be accessed by any web service client. These topics are covered in this chapter along with the use of WF from an ASP.NET Web Forms application.

Chapter 16: Workflow Serialization and Markup

The goal of this chapter is to demonstrate the use of workflow markup and serialization. Each workflow definition can be declared and expressed in several forms, including markup. Markup declares a workflow in a simple XML form that doesn't require compilation and can be parsed and executed directly by the workflow runtime engine. The advantage of using markup is that it is much easier to modify the workflow definition outside of Visual Studio, since it doesn't require compilation.

Chapter 17: Hosting the Workflow Designers

After workflow serialization and markup is presented in Chapter 16, this chapter shows you how to build your own workflow designer. WF includes the classes that you need to host the workflow designers in your own application. The bulk of this chapter presents a working designer application that enables you to define and modify markup-only workflows.

What You Need to Use This Book

To execute the examples presented in this book, you need a development machine with an OS that supports the .NET 3.0 runtime. Currently, that is Windows XP Service Pack 2, Windows 2003 Server Service Pack 1, or Windows Vista. A fully licensed version of Visual Studio 2005 is also necessary in order to use the visual workflow designers and to work with the example C# code. The workflow designers are not supported on the Express editions of Visual Studio or Visual C#.

The runtime for Windows Workflow Foundation is distributed as part of .NET 3.0. In addition to this, you will need to install the designated version of the Windows SDK that supports workflow development. A separate set of extensions to Visual Studio 2005 must also be downloaded and installed to use the workflow designers, project templates, and workflow debugger.

You should be able to locate download links for all of the necessary files from the Microsoft .NET Framework Development Center (`http://msdn2.microsoft.com/en-us/netframework/default.aspx`).

Obtaining This Book's Source Code

I have found that the best way to learn and retain a new skill is through hands-on examples. For this reason, this book contains a lot of example source code. I've been frustrated on more than one occasion with technical books that don't print all of the source code in the book. The code may be available for download, but then you need to have a computer handy while you are reading the book. That doesn't work well on the beach. So I've made it a point to present all of the code that is necessary to actually build and execute the examples.

When you are ready to execute the example code, you don't have to enter it yourself. You can download all of the code presented in this book from the Apress site at `http://www.apress.com` and go to the Source Code/Download section. I've organized all of the downloadable code into separate folders for each chapter with a separate Visual Studio solution for each chapter. The only exception is one shared project that is referenced by projects in most of the chapters of this book. I'd suggest that you also keep your code separate by chapter as you work through the examples in this book.

How to Reach Me

If you have questions or comments about this book or Windows Workflow, I'd love to hear from you. Just send your e-mail to `workflow@bukovics.com`. To make sure your mail makes it past my spam filters you might want to include the text **ProWF** somewhere in the subject line.

CHAPTER 1

A Quick Tour of Windows Workflow Foundation

This chapter presents a brief introduction to Windows Workflow Foundation (WF). Instead of diving deeply into any single workflow topic, it provides you with a sampling of topics that are fully presented in other chapters.

You'll learn why workflows are important and why you might want to develop applications using them. You'll then jump right in and implement your very first functioning workflow. Additional hands-on examples are presented that demonstrate other features of Windows Workflow Foundation.

Why Workflow?

As developers, our job is to solve real business problems. The type and complexity of the problems will vary broadly depending on the nature of the business. But regardless of the complexity of any given problem, we tend to solve problems in the same way: we break the problem down into manageable parts. Those parts are further divided into smaller tasks, and so on.

When we've finally reached a point where each task is the right size to understand and manage, we identify the steps needed to accomplish the task. The steps usually have an order associated with them. They represent a sequence of individual instructions that will only yield the expected behavior when they are executed in the correct order.

In the traditional programming model, you implement a task in code using your language of choice. The code specifies what to do (the execution instructions) along with the sequence of those instructions (the flow of control). You also include code to make decisions (rules) based on the value of variables, the receipt of events, and the current state of the application.

A workflow is simply an ordered series of steps that accomplish some defined purpose according to a set of rules. By that definition, what I just described is a *workflow*.

It might be defined entirely in code, but it is no less a type of workflow. We already use workflows every day we develop software. We might not consider affixing the workflow label to our work, but we do use the concepts even if we are not consciously aware of them.

So why all of this talk about workflows? Why did I write a book about them? Why are you reading it right now?

Workflows Are Different

The workflow definition that I gave previously doesn't tell the whole story, of course. There must be more to it, and there is. To a developer, the word *workflow* typically conjures up images of a highly visual environment where complex business rules and flow of control are declared graphically.

It's an environment that allows you to easily visualize and model the activities (steps) that have been declared to solve a problem. And since you can visualize the activities, it's easier to change, enhance, and customize them.

But there is still more to workflows than just the development environment. Workflows represent a different programming model. It's a model that promotes a clear separation between *what* to do and *when* to do it. This separation allows you to change the *when* without affecting the *what*. Workflows generally use a declarative programming model rather than a procedural one. With this model, business logic can be encapsulated in discrete components. But the rules that govern the flow of control between components are declarative.

General purpose languages such as C# or Visual Basic can obviously be used to solve business problems. But the workflow programming model really enables you to implement your own domain-specific language. With such a language, you can express business rules using terms that are common to a specific problem domain. Experts in that domain are able to view a workflow and easily understand it, since it is declared in terminology that they understand.

For example, if your domain is banking and finance, you might refer to accounts, checks, loans, debits, credits, customers, tellers, branches, and so on. But if the problem domain is pizza delivery, those entities don't make much sense. Instead, you would model your problems using terms such as menus, specials, ingredients, addresses, phone numbers, drivers, tips, and so on. The workflow model allows you to define the problem using terminology that is appropriate for each problem domain.

Workflows allow you to easily model system and human interactions. A *system interaction* is how we as developers would typically approach a problem. You define the steps to execute and write code that controls the sequence of those steps. The code is always in total control.

Human interactions are those that involve real live people. The problem is that people are not always as predictable as your code. For example, you might need to model a mortgage loan application. The process might include steps that must be executed by real people in order to complete the process. How much control do you have over the order of those steps? Does the credit approval always occur first, or is it possible for the appraisal to be done first? What about the property survey? Is it done before or after the appraisal? And what activities must be completed before you can schedule the loan closing? The point is that these types of problems are difficult to express using a purely procedural model because human beings are in control. The exact sequence of steps is not always predictable. The workflow model really shines when it comes to solving human interaction problems.

Why Windows Workflow Foundation?

If workflows are important, then why use Windows Workflow Foundation? Microsoft has provided this foundation in order to simplify and enhance your .NET development. It is not a stand-alone application. It is a software foundation that is designed to enable workflows within your applications. Regardless of the type of application you are developing, there is something in WF that you can leverage.

If you are developing line-of-business applications, you can use WF to orchestrate the business rules. If your application is comprised of a series of human interactions, you can use a WF state machine workflow to implement logic that can react to those interactions. If you need a highly customizable application, you can use the declarative nature of WF workflows to separate the business logic from the execution flow. This allows customization of the flow of control without affecting the underlying business logic. And if you are looking for a better way to encapsulate and independently test your application logic, implement the logic as discrete custom activities that are executed within the WF runtime environment.

There are a number of good reasons to use WF, and here are a few of them:

- It provides a flexible and powerful framework for developing workflows. You can spend your time and energy developing your own framework, visual workflow designer, and runtime environment. Or you can use a foundation that Microsoft provides and spend your valuable time solving real business problems.
- It promotes a consistent way to develop your applications. One workflow looks very similar to the next. This consistency in the programming model and tools improves your productivity when developing new applications and maintaining existing ones.
- It supports *sequential* and *state machine* workflows. Sequential workflows are generally used for system interactions. State machine workflows are well-suited to solving problems that focus on human interaction.
- It supports workflow persistence. The ability to save and later reload the state of a running workflow is especially important when modeling human interactions.
- It supports problem solving using a domain-specific model. Microsoft encourages you to develop your own custom activity components. Each custom component addresses a problem that is specific to your problem domain and uses terminology that is common to the domain.
- It provides a complete workflow ecosystem. In addition to the workflow runtime itself, Microsoft also provides a suite of standard activities, workflow persistence, workflow monitoring and tracking, and a workflow designer that is integrated with Visual Studio which you can also host in your own applications.
- It is free of charge. Because of this and its tight integration with Visual Studio, it will become the de facto standard workflow framework for Windows developers. A growing community of other WF developers is already in place. They are already sharing their ideas, their custom activity components, and other code.

Your Development Environment

In order to develop applications using Windows Workflow Foundation, you'll need to install a minimum set of software components. The minimum requirements are the following:

- Visual Studio 2005 Enterprise, Professional, or Standard
- The .NET 3.0 runtime
- A designated version of the Windows SDK that supports WF
- The Windows Workflow add-in to Visual Studio

Please refer to the Microsoft MSDN site for the latest download and installation instructions.

Hello Workflow

At this point you are ready to create your first workflow. In the world of technology in which we work, it has become customary to begin any new technical encounter with a "Hello World" example.

Not wanting to break with tradition, I present a "Hello Workflow" example in the pages that follow. If you follow along with the steps as I present them, you will have a really simple yet functional workflow application.

In this example, and in the other examples in this chapter, I present important concepts that are the basis for working with all workflows, regardless of their complexity. If you already have experience working with Windows Workflow Foundation, you might feel compelled to skip over this information. If so, go ahead, but you might want to give this chapter a quick read anyway.

To implement the "Hello Workflow" example, you'll create a sequential workflow project, add one of the standard activities to the workflow, and then add code to display "Hello Workflow" on the `Console`.

Creating the Workflow Project

Workflow projects are created in the same way as other project types in Visual Studio. After starting Visual Studio 2005, you select File ➤ New ➤ Project. A New Project dialog is presented that should look similar to the one shown in Figure 1-1.

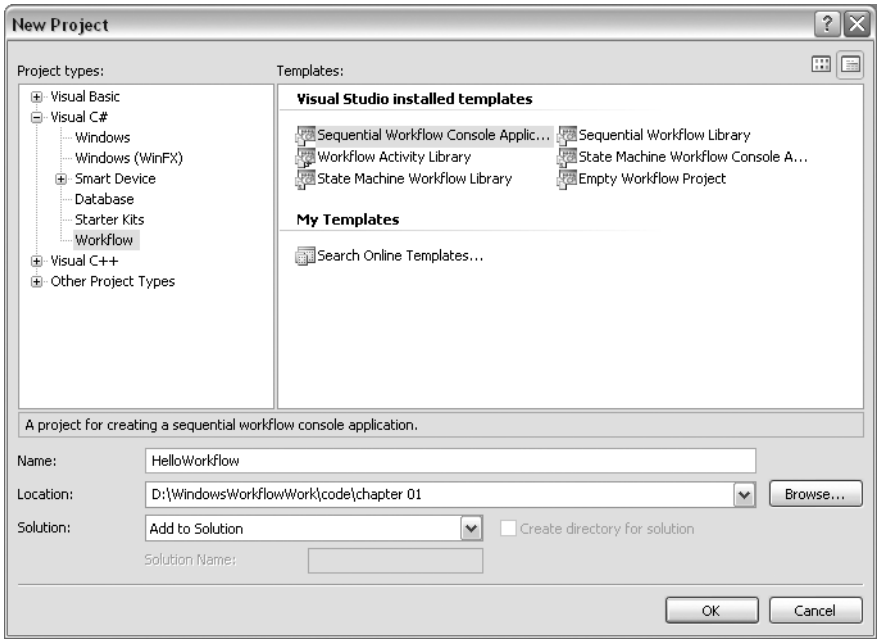

Figure 1-1. *Sequential workflow console application New Project dialog*

After selecting Visual C# as the language, you'll see Workflow as one of the available project template categories. As shown in Figure 1-1, there are several workflow project templates available. For this example, you should choose Sequential Workflow Console Application. This produces a console application that supports the use of Windows Workflow Foundation. A *sequential workflow* is one that executes a series of steps in a defined sequence. That's exactly the type of workflow that we need for this example.

■ **Note** Visual Basic developers don't have to worry. The same Workflow project templates are also available for Visual Basic if that's your language of choice.

You should now enter a meaningful name for the project, such as **HelloWorkflow**, select a location, and press OK to create the new project.

Note In the example shown in Figure 1-1, the project will be added to an existing solution named Chapter 01. An existing solution is used in order to neatly organize all of the examples for this book by chapter. This is not a strict requirement and you are free to create individual solutions for each project if you prefer. If you create a new project without first opening or creating a solution, the New Project dialog does not include the Add to Solution option.

After a second or two, the new project is created. The Solution Explorer window shows the source files that are created as part of the project, as shown in Figure 1-2.

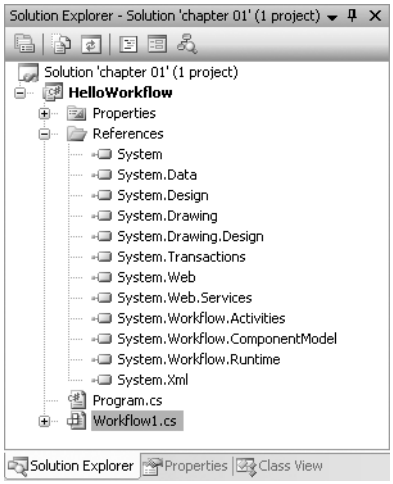

Figure 1-2. *Solution Explorer for the new workflow project*

Notice that I've expanded the References folder in order to show the assembly references for the project. By selecting a workflow project as the template, the assembly references necessary to use WF are added for you. The workflow-related assemblies are the following:

- System.Workflow.Activities
- System.Workflow.ComponentModel
- System.Workflow.Runtime

Within these assemblies, the workflow-related classes are organized into a number of namespaces. In your code, you only need to reference the namespaces that you are actually using.

The project template created a file named `Program.cs`. Since this is a console application, this file contains the `Main` method associated with the application. We'll review the generated code for this file shortly.

Introducing the Workflow Designer

Also generated is a file named `Workflow1.cs`. This file contains the code that defines the workflow and is associated with the visual workflow designer as its editor. When this file is opened, as it is when the project is first created, the initial view of the designer looks like Figure 1-3.

Figure 1-3. *Initial view of the visual workflow designer*

The workflow designer is the primary canvas that you will use to define your workflows. You can also define a workflow entirely in code, in much the same way that you can define an entire Windows form or other user interface elements in code. But one of the best features of WF is the designer, and using it will greatly increase your productivity when defining workflows. The designer supports dragging and dropping of activities onto the workflow canvas from the Visual Studio Toolbox.

Using Workflow Activities

An *activity* represents a step in the workflow and is the fundamental building block of all WF workflows. All activities either directly or indirectly derive from the base `System.Workflow.ComponentModel.Activity` class. Microsoft supplies a set of standard activities that you can use, but you are encouraged to also develop your own custom activities. Each activity is designed to serve a unique purpose and encapsulates the logic needed to fulfill that purpose.

For a sequential workflow such as this one, the order of the activities in the workflow determines their execution sequence. A sequential workflow has a defined beginning and ending point. As shown in Figure 1-3, previously, these points are represented by the arrow at the top of the workflow and the circle symbol at the bottom. What takes place between these two points is yours to define by dropping activities onto the canvas. Once you've dropped a series of activities onto a workflow, you can modify their execution order by simply dragging them to a new location.

Activities wouldn't be very useful if there wasn't a way to change their default behavior. Therefore, most activities provide a set of properties that can be set at design time. The workflow itself also has properties that can be set at design time.

Figure 1-4 shows just a few of the standard activities that are supplied by Microsoft. I review all of the available activities in Chapter 3.