



Essential Studio 2013 Volume 4 - v.11.4.0.26

Essential Schedule for Windows Forms



Contents

1	Overview	4
1.1	Introduction to Essential Schedule	4
1.2	Prerequisites and Compatibility	7
1.3	Documentation	8
2	Installation and Deployment	10
2.1	Installation.....	10
2.2	Sample and Location.....	13
2.3	Deployment Requirements	16
3	Getting Started	17
3.1	Control Structure	17
3.2	Class Diagram	23
3.3	Tutorial.....	24
3.3.1	Lesson 1: Using ScheduleControl and SimpleScheduleDataProvider.....	24
3.3.1.1	Lesson: Using ScheduleControl	25
3.4	Data used by ScheduleControl.....	39
3.4.1	ScheduleData Base Classes	41
3.4.1.1	The Appointments Data.....	41
3.4.1.2	The DropLists	47
3.4.2	IScheduleData Interfaces	48
3.4.2.1	The Appointments Data.....	49
3.4.2.2	The DropLists	53
4	Concepts and Features	55
4.1	ScheduleControl	55
4.1.1	Caption Panel	56
4.1.2	Navigation Panel	56
4.1.3	Navigation Calendar	56
4.2	Customizing Appearance	57
4.3	Metro Theme for Essential Schedule	62
4.3.1	Applying Metro Theme to the Schedule Control	62
4.4	Item Dragging Context in the ItemChanging event	63

4.5	Time Interval.....	66
4.5.1	Setting the Time Interval in Seconds Format	66
5	Frequently Asked Questions	67
5.1	How to disable the drag behavior of schedule appointments in the ScheduleControl	67
5.2	How to set the schedule appointments as read-only	68
5.3	How to change the current display to a desired Schedule View Type	68
5.4	How to obtain the date that is clicked in the ScheduleControl	69
5.5	How to prevent the switching of schedule view type	70
5.6	How to show the start and end time with scheduled appointments	70
5.7	How to set the Transparency Level of a Span Appointment	72

1 Overview

This section covers information on Essential Schedule control, its key features, and prerequisites to use the control, its compatibility with various OS and browsers and finally the documentation details complimentary with the product. It comprises the following sub sections:

1.1 Introduction to Essential Schedule

Syncfusion Essential Schedule is a Windows Forms class library built around the functionality that is found in the Windows Forms Grid control. The control allows you to add scheduling support to your applications. The scheduling support includes creating new appointments, displaying these appointments in a variety of views, including Monthly, Daily, Weekly, Work Week and multiple days. In the daily formats, you can use the UI to drag appointments to another time slot and to extend appointments. A flexible navigation calendar lets you easily home in on the dates you would like to see in the Schedule control.

The data displayed in the Schedule control is provided through any object that implements the `IScheduleDataProvider` interface that is defined in the library. Included in the library is one concrete implementation based on the `ArrayList` that uses disk files to persist the data. With this implementation, you can easily have several schedule files that can either be treated as individual schedules or merged to be treated as a single schedule.



Figure 1: Schedule Control

The Scheduler control finds a wide variety of applications such as Time Tables, Calendars, Event Scheduling, Sequences, Activities, Project Management, Reservations, Resource Usage Planners, and so on.

Key Features

Some of the key features of the Schedule control are listed below:

- **Caption Panel** displays a caption at the top of the Schedule control. There are also two button objects on this panel that will navigate the Schedule forward and backward. This panel is docked at the top of the ScheduleControl client area.
- **Navigation Panel** can be placed additional controls and make them appear adjacent to the Schedule control. This can be optionally docked to the left or right side of the ScheduleControl. You can also hide this panel. The ScheduleControl.Calendar which is a

NavigationCalendar object is docked at the top of this panel. There is also a Splitter docked under the Navigation Calendar that allows you to display more or fewer calendars in the NavigationCalendar. The default setting displays two such calendars.

- **Navigation Calendar** is a GridControl-derived object that displays multiple calendars allowing you to select the dates displayed in the Schedule control. This calendar is docked at the top of the NavigationPanel. The number of calendars displayed in the Navigation Calendar is determined by its client height. Enlarging the height of the Navigation Calendar will display more calendars. There is a Splitter docked under the Navigation Calendar to facilitate such sizing.
- **Schedule Grid** is a Grid control-derived object that displays the actual schedule content, i.e., the appointments for the various dates. The actual look of this Grid control is determined by the ScheduleViewType which is set by using the ScheduleControl.ScheduleType property.


You can have both timed and untimed (all-day) appointments using this Schedule control. The ScheduleControl is easily localized and can display customizable alerts.



The product comes with numerous samples as well as an extensive documentation to guide you. This User Guide provides detailed information on the features and functionalities of the Schedule control. It is organized into the following sections:

- **Overview**-This section gives a brief introduction to our product and its key features.
- **Installation and Deployment**-This section elaborates on the install location of the samples, license etc.
- **Getting Started**-This section guides you on getting started with Windows application, controls etc.
- **Concepts and Features**-The features of the Schedule control are illustrated with use case scenarios, code examples and screen shots under this section.
- **Frequently Asked Questions**-This section discusses various frequently asked questions with their answers with examples and code snippets.

Document Conventions

The conventions listed below will help you to quickly identify the important sections of information, while using the content:

Convention	ICON	Description
Note	 Note:	Represents important information

Example	Example	Represents an example
Tip		Represents useful hints that will help you in using the controls/features
Important Note		Represents additional information on the topic

1.2 Prerequisites and Compatibility

This section covers the requirements mandatory for using Essential Schedule control. It also lists operating systems and browsers compatible with the product.

Prerequisites

The prerequisites details are listed below:

Development Environments	<ul style="list-style-type: none">• Visual Studio 2013• Visual Studio 2012 (Ultimate, Premium, Professional and Express)• Visual Studio 2010 (Ultimate, Premium, Professional and Express)• Visual Studio 2008 (Team, Professional, Standard and Express)• Visual Studio 2005 (Team, Professional, Standard and Express)• Borland Delphi for .NET• SharpCode
.NET Framework versions	<ul style="list-style-type: none">• .NET 4.5• .NET 4.0• .NET 3.5• .NET 2.0
Operating Systems	<ul style="list-style-type: none">• Windows 8.1 (32 bit and 64 bit)

	<ul style="list-style-type: none">• Windows Server 2012 (32 bit and 64 bit)• Windows 7 (32 bit and 64 bit)• Windows Server 2008 (32 bit and 64 bit)• Windows Vista (32 bit and 64 bit)• Windows XP• Windows 2003
--	---

Compatibility


The compatibility details are listed below:

Operating Systems	<ul style="list-style-type: none">• Windows 8.1 (32 bit and 64 bit)• Windows Server 2012 (32 bit and 64 bit)• Windows 7 (32 bit and 64 bit)• Windows Server 2008 (32 bit and 64 bit)• Windows Vista (32 bit and 64 bit)• Windows XP• Windows 2003
--------------------------	---

1.3 Documentation

Syncfusion provides the following documentation segments to provide all necessary information for using Essential Schedule control in an efficient manner.

Type of documentation	Location
Readme	[drive:]\Program Files\Syncfusion\Essential Studio\x.x.x.x\Infrastructure\Data\Release Notes\readme.htm
Release Notes	[drive:]\Program Files\Syncfusion\Essential Studio\x.x.x.x\Infrastructure\Data\Release Notes\Release Notes.htm
User Guide (this document)	Online

	<p>http://help.syncfusion.com/resources(Navigate to the Schedule for Windows Forms User Guide.)</p> <p> Note: Click Download as PDF to access a PDF version.</p> <p>Installed Documentation</p> <p>Dashboard -> Documentation -> Installed Documentation.</p>
Class Reference	<p>Online</p> <p>http://help.syncfusion.com/resources(Navigate to the Windows Forms User Guide. Select <i>Schedule</i> in the second text box, and then click the Class Reference link found in the upper right section of the page.)</p> <p>Installed Documentation</p> <p>Dashboard -> Documentation -> Installed Documentation.</p>

2 Installation and Deployment

This section covers information on the install location, samples, licensing, patches update and updation of the recent version of Essential Studio. It comprises the following sub-sections:

2.1 Installation

For step-by-step installation procedure for the installation of Essential Studio, refer to the **Installation** topic under **Installation and Deployment** in the **Common UG**.

See Also

For licensing, patches and information on adding or removing selective components refer the following topics in Common UG under **Installation and Deployment**.

- Licensing
- Patches
- Add / Remove Components

Sample Installation Location

The Schedule Windows Forms samples are installed in the following location, locally on the disk:

**[Local Drive]:\.. \Syncfusion\Essential Studio\[VersionNumber]\
Windows\Schedule.Windows\Samples\2.0**

Viewing Samples

To view the samples, follow the steps below:

Click Start → All Programs → Syncfusion → Essential Studio <version number> → Dashboard.

Essential Studio Enterprise Edition window is displayed.

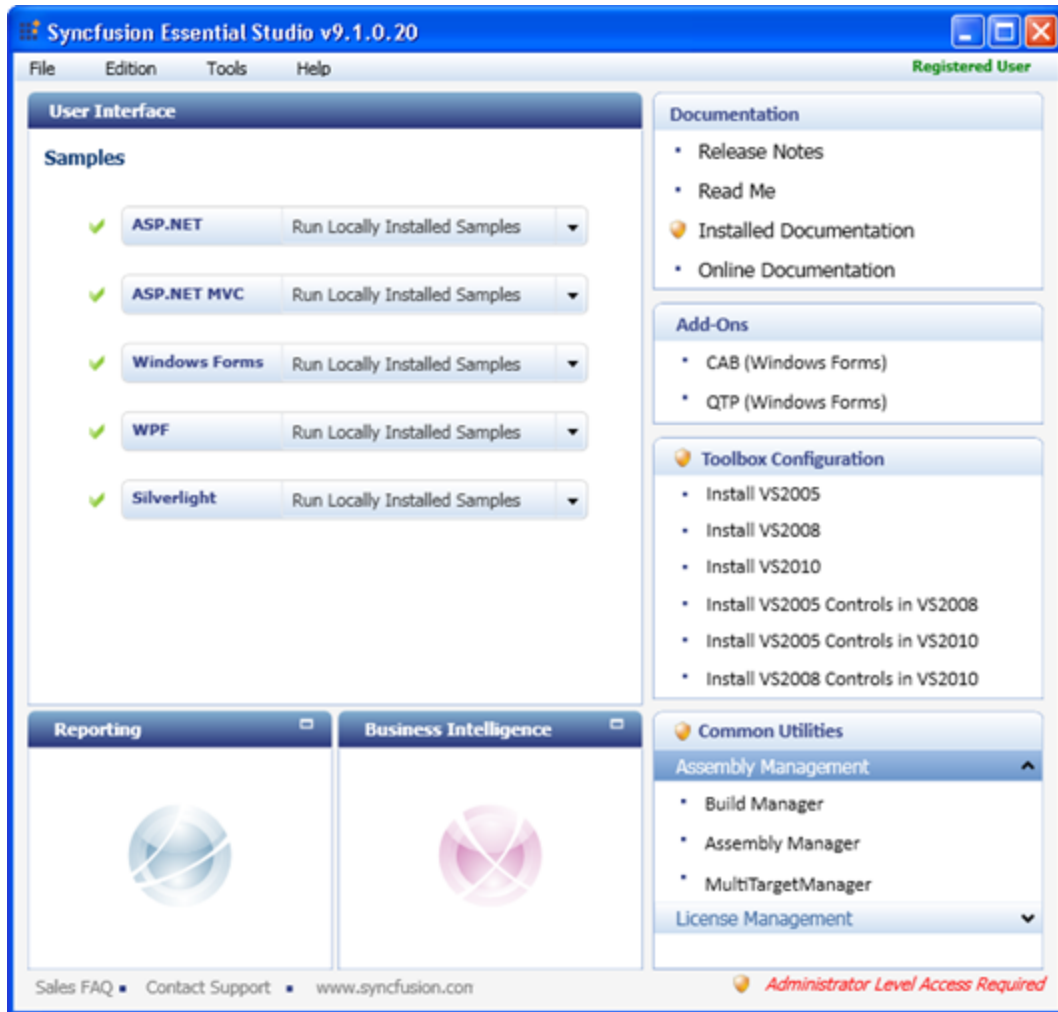


Figure 2: Syncfusion Essential Studio Dashboard

User Interface Edition panel is displayed by default.

Click the drop-down button of the Windows platform. The following options are displayed.

- **Run Locally Installed Samples** – View the locally installed Schedule samples for windows using the sample browser
- **Run Online Samples** – View the online Schedule samples for windows
- **Explore Samples** – Locate the Schedule samples on the disk

You can view the samples in the preceding three ways.

1. Click **Run, Locally Installed Samples** link. Essential Studio User Interface Edition Windows Forms sample browser is displayed.



Figure 3: User Interface Edition Windows Forms Sample Browser

2. Under **Contents** tab, expand the **Schedule samples** to view the samples of the control.

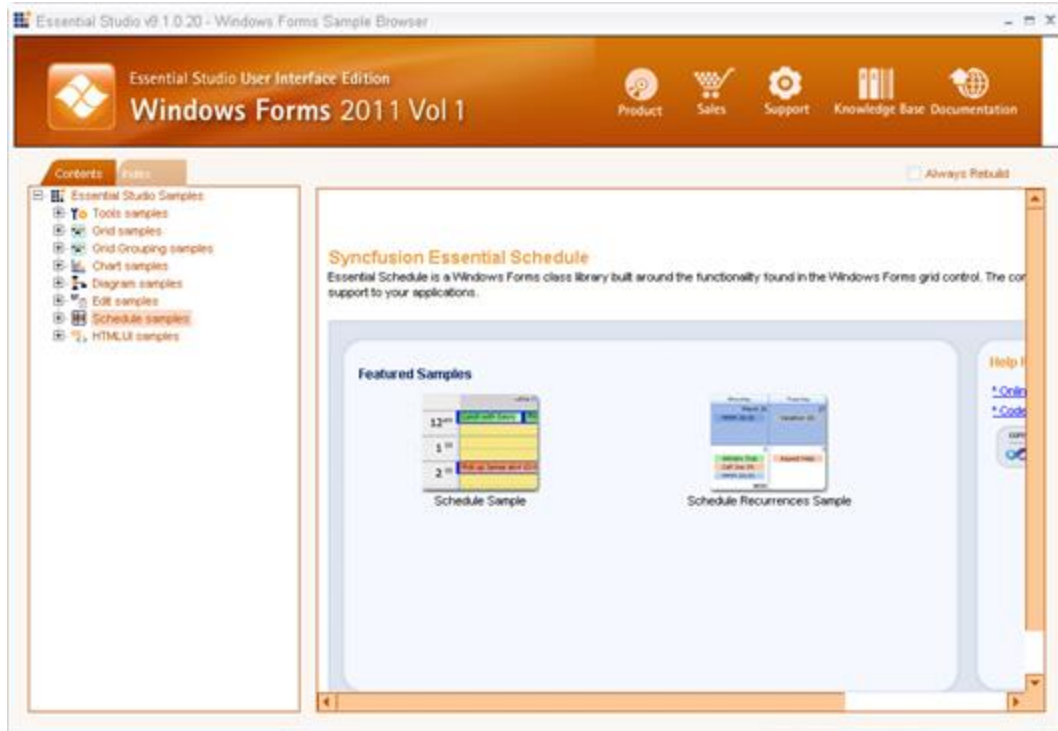


Figure 4: Schedule Samples for Windows

3. A list of samples will be displayed on the left hand side of the page. Select any sample and browse through the features.
4. In the right pane, click **Run Sample** icon to run the selected sample.

Source Code Location

The source code for Essential Schedule Windows is available at the following default location:

[System Drive]\Program Files\Syncfusion\Essential Studio[Version Number]\Windows\Schedule.Windows\Src

2.2 Sample and Location

This section covers the location of the installed samples and describes the procedure to run the samples through the sample browser. It also lists the location of source code.

Sample Installation Location

The Schedule Windows Forms samples are installed in the following location:

...My Documents\Syncfusion\EssentialStudio\Version Number\Windows\Schedule.Windows\Samples\2.0

Viewing Samples

To view the samples, follow the steps below:

1. Click **Start-->All Programs-->Syncfusion-->Essential Studio <version number> -->Dashboard**.
Essential Studio Enterprise Edition window is displayed.

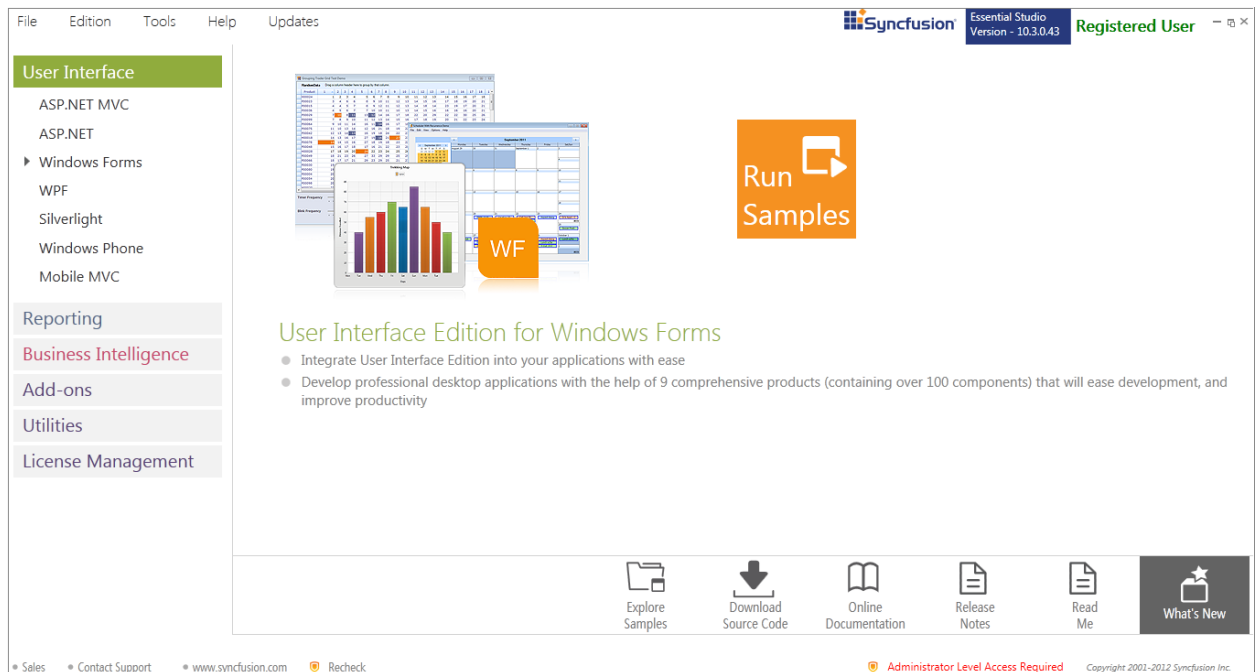


Figure 5: Syncfusion Essential Studio Dashboard

2. In the Dashboard window, click **Run Samples** for Windows Forms under UI Edition. The UI Windows Form Sample Browser window is displayed.



Note: You can view the samples in any of the following three ways:

- **Run Samples**-Click to view the locally installed samples
- **Online Samples**-Click to view online samples
- **Explore Samples**-Explore BI Web samples on disk

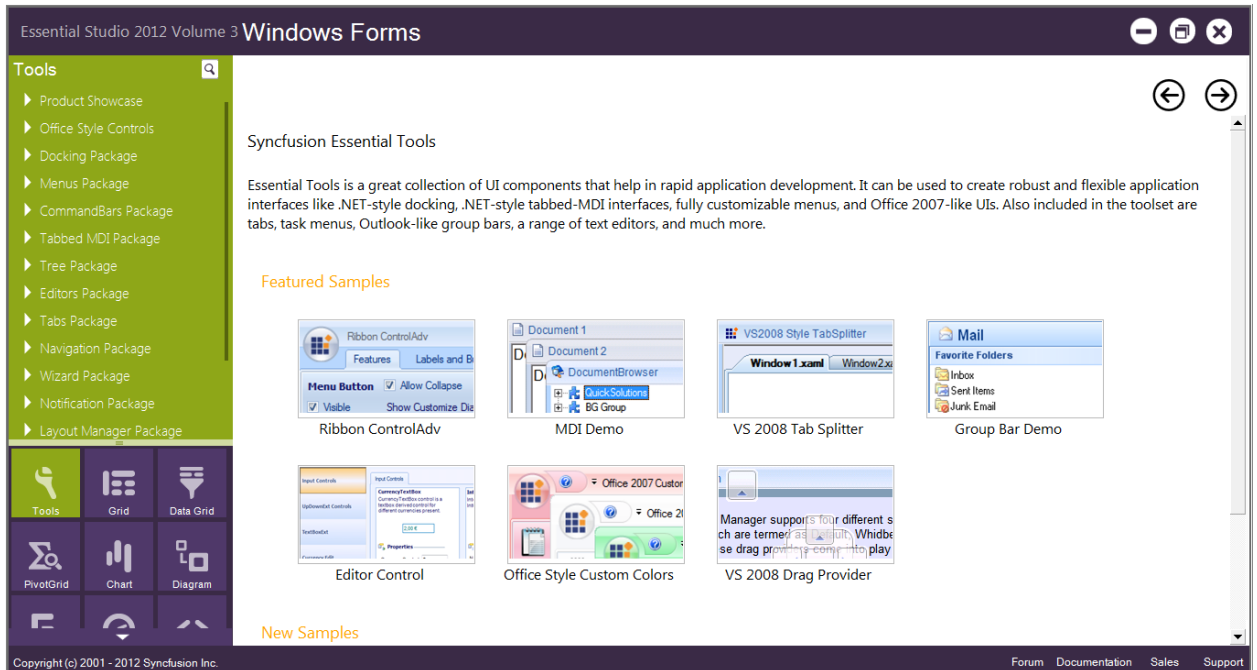


Figure 6: User Interface Edition Windows Forms Sample Browser

3. Click **Schedule** from the bottom-left pane. Schedule samples will be displayed.

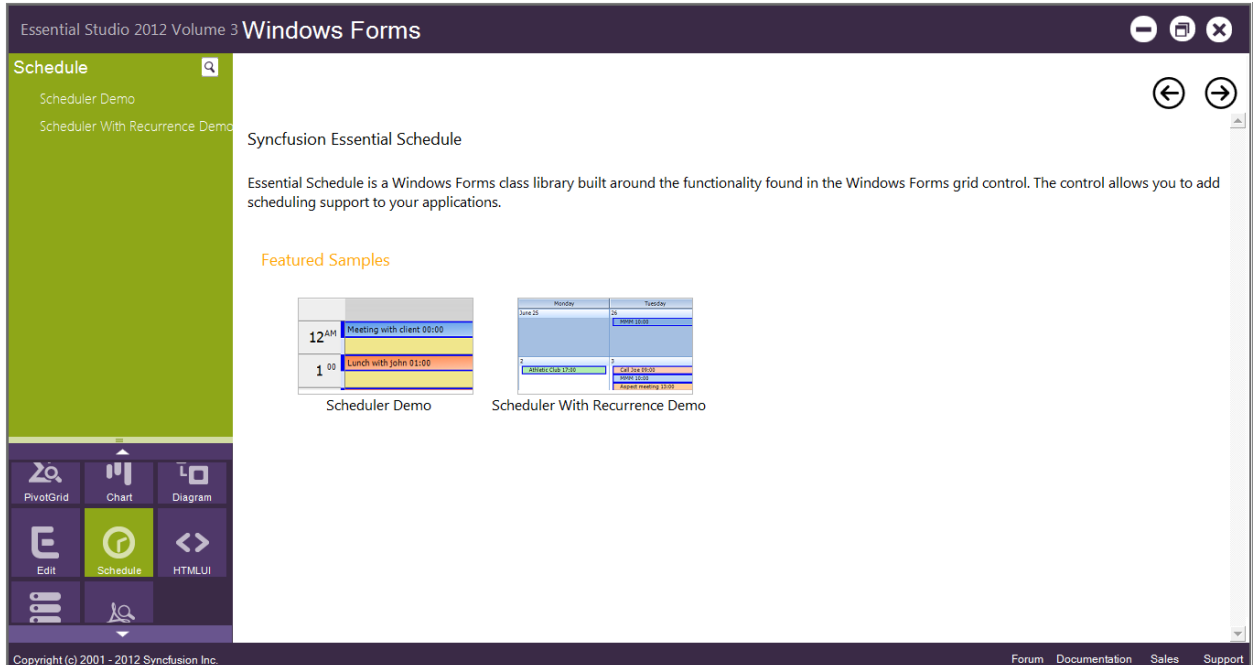


Figure 7: Schedule Samples for Windows

4. Select any sample and browse through the features.

Source Code Location

The source code for Essential Schedule Windows is available at the following default location:

[System Drive]:\Program Files\Syncfusion\Essential Studio\[Version Number]\Windows\Schedule.Windows\Src

2.3 Deployment Requirements

Toolbox Entries Made

Essential Schedule Windows places the following control into your Visual Studio .NET toolbox from where you can drag the control onto a form and start working with it.

- ScheduleControl

Dll List

While deploying an application that references a Syncfusion Essential Schedule Windows assembly, the following dependencies must be included in the distribution.

- Syncfusion.Core.dll
- Syncfusion.Shared.Base.dll
- Syncfusion.Shared.Windows.dll
- Syncfusion.Schedule.Base.dll
- Syncfusion.Schedule.Windows.dll
- Syncfusion.Grid.Base.dll
- Syncfusion.Grid.Windows.dll

3 Getting Started

This section guides you on getting started with Windows application, controls etc.

3.1 Control Structure

The following screen shot shows the structure of the ScheduleControl.

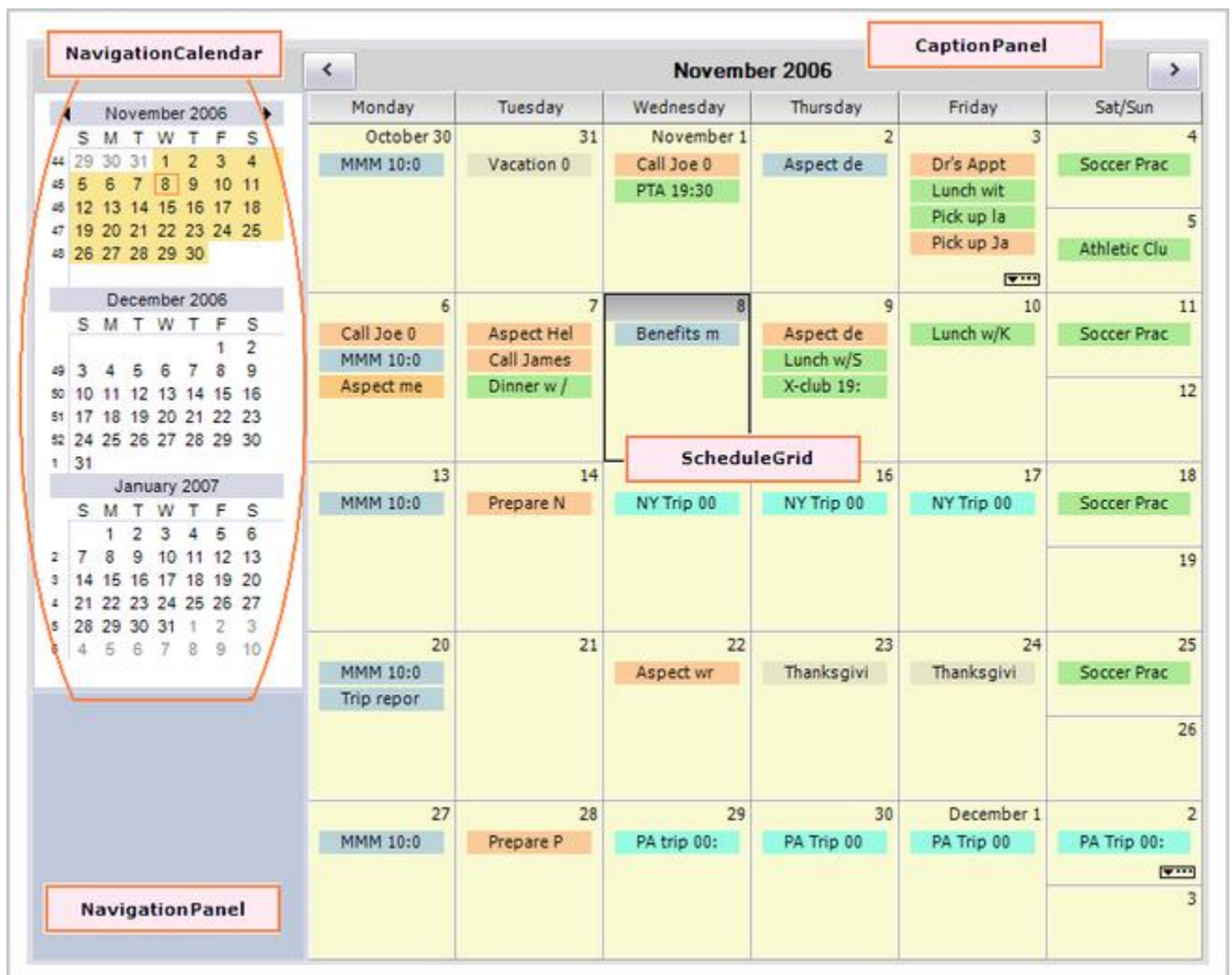


Figure 8: Structure of ScheduleControl

Essential Schedule primarily consists of a **UserControl derived class** named **ScheduleControl**.

This section discusses the main properties of the ScheduleControl. The data for the ScheduleControl comes from any object that implements IScheduleDataProvider. The following discussions elaborate on the concrete implementation of the IScheduleDataProvider, based on an ArrayList-derived object that serializes to a disk file.

The above screen shot shows a ScheduleControl displaying a Month view. The four marked areas are actually Control-derived objects (two Panels and two GridControls). These controls have been added to the **ScheduleControl.Controls** collection. Any of the four controls except the ScheduleGrid can be hidden through the property settings. Here is a short description of each of the 4 labeled areas:

CaptionPanel: This is a Panel that displays a caption at the top of the ScheduleControl. There are also two button objects on this panel that will navigate the Schedule forward and backward. You can hide this panel by using the ScheduleControl.Appearance.ShowCaption property. This panel is docked at the top of the ScheduleControl client area.

NavigationPanel: This is a Panel where you can place additional controls and make them appear adjacent to the ScheduleControl. This can be optionally docked to the left or right side of the ScheduleControl. You can also hide this panel. The ScheduleControl.Calendar which is a NavigationCalendar object is docked at the top of this panel. There is also a Splitter docked under the NavigationCalendar that allows you to display more or fewer calendars in the NavigationCalendar. The default setting displays two such calendars. The picture below displays three. You can easily put your own controls under the NavigationCalendar using code similar to these snippets.

[C#]

```
Panel p = new Panel();
p.BackColor = Color.Blue;
p.Dock = DockStyle.Fill;
p.BackgroundImage = Image.FromFile("../..\\sync.png");
p.BackgroundImageLayout = ImageLayout.Tile;

this.ScheduleControl1.AddControlToNavigationPanel(p);
```

NavigationCalendar: This is a GridControl-derived object that displays multiple calendars allowing you to select the dates displayed in the ScheduleControl. The NavigationCalendar is docked at the top of the NavigationPanel. The number of calendars displayed in the NavigationCalendar is determined by its client height. Enlarging the height of the NavigationCalendar, will display more calendars. This can be facilitated by using the Splitter docked under the NavigationCalendar.

ScheduleGrid: This is a GridControl-derived object that displays the actual schedule content, i.e., the appointments for the various dates. The actual look of this GridControl is determined by the ScheduleViewType which is set by using the ScheduleControl.ScheduleType property.

Here is a Day view that shows a panel added under the NavigationCalendar by using the **ScheduleControl1.AddControlToNavigationPanel** code mentioned above.

You can dock any control under the NavigationCalendar by using this method.

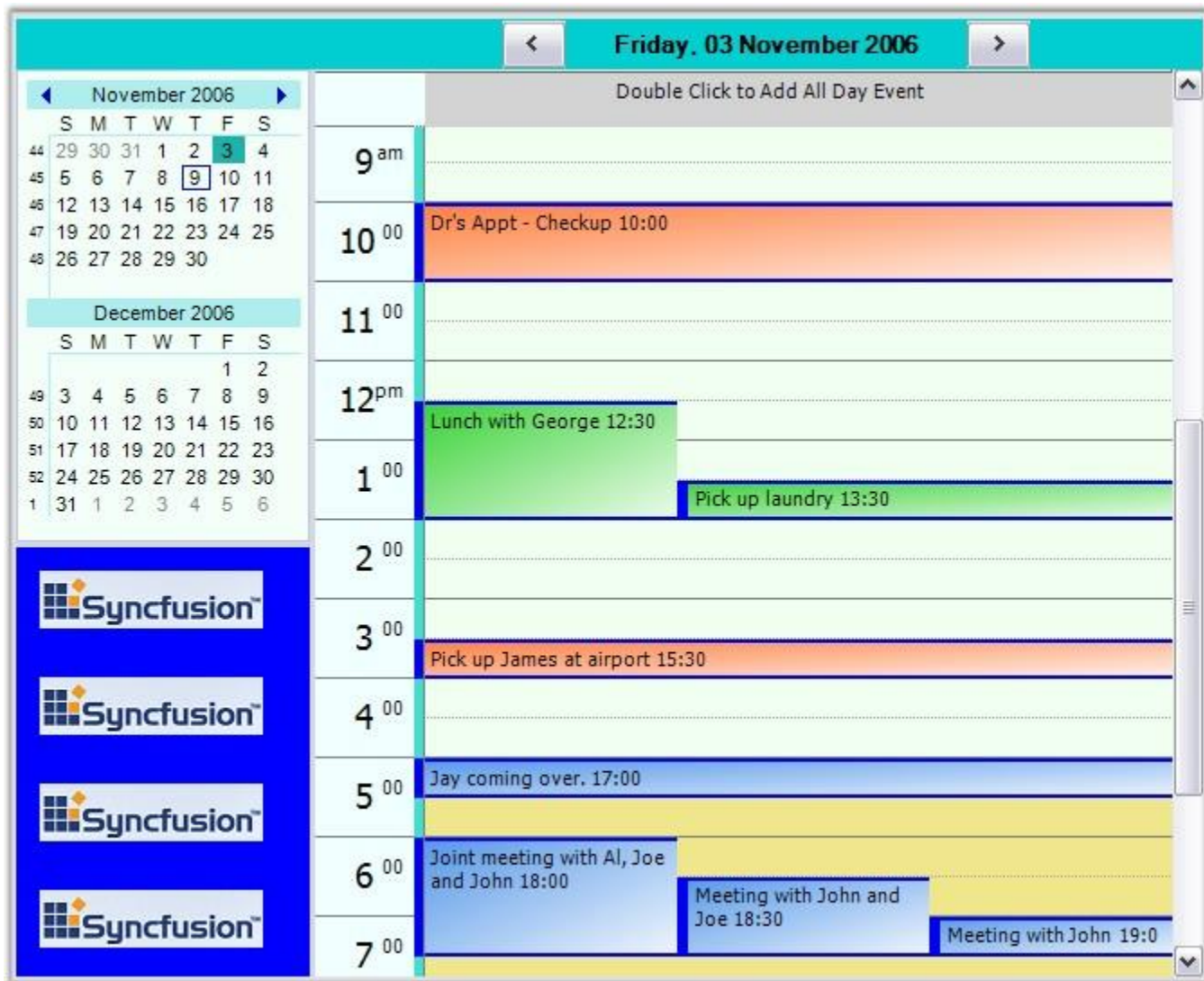


Figure 9: ScheduleControl Day View showing a Special Control added under the NavigationCalendar

In addition to the Month view, the ScheduleControl can also display Day, WorkWeek, Week and Custom views.

A Custom view is one where you can display up to eight individual days in the ScheduleGrid. You can easily switch views by using the **ScheduleControl.PerformSwitchToScheduleViewTypeClick** method.

Here are a series of screen shots illustrating these different views.

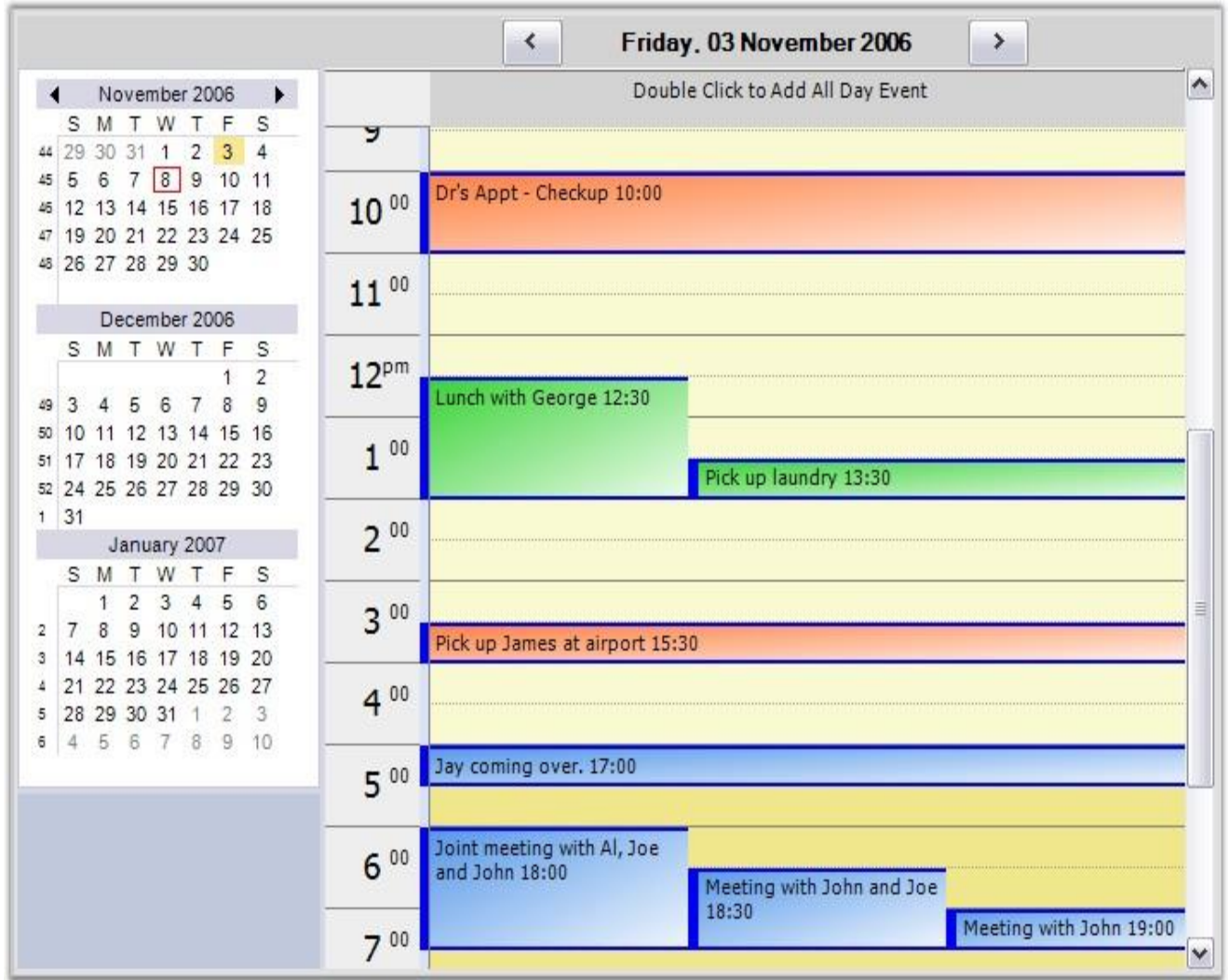


Figure 10: ScheduleControl Day View

Notice in the WorkWeek view snapshot below, there is a Vacation entry at the top of 10/31/2006. This entry is an All-Day entry which has no specific time assigned to it. It is simply associated with the particular date. For the Day, WorkWeek and Custom views, All-Day entries are displayed in a frozen row at the top of the ScheduleGrid. For Week and Month views, All-Day entries are listed with the time entries.

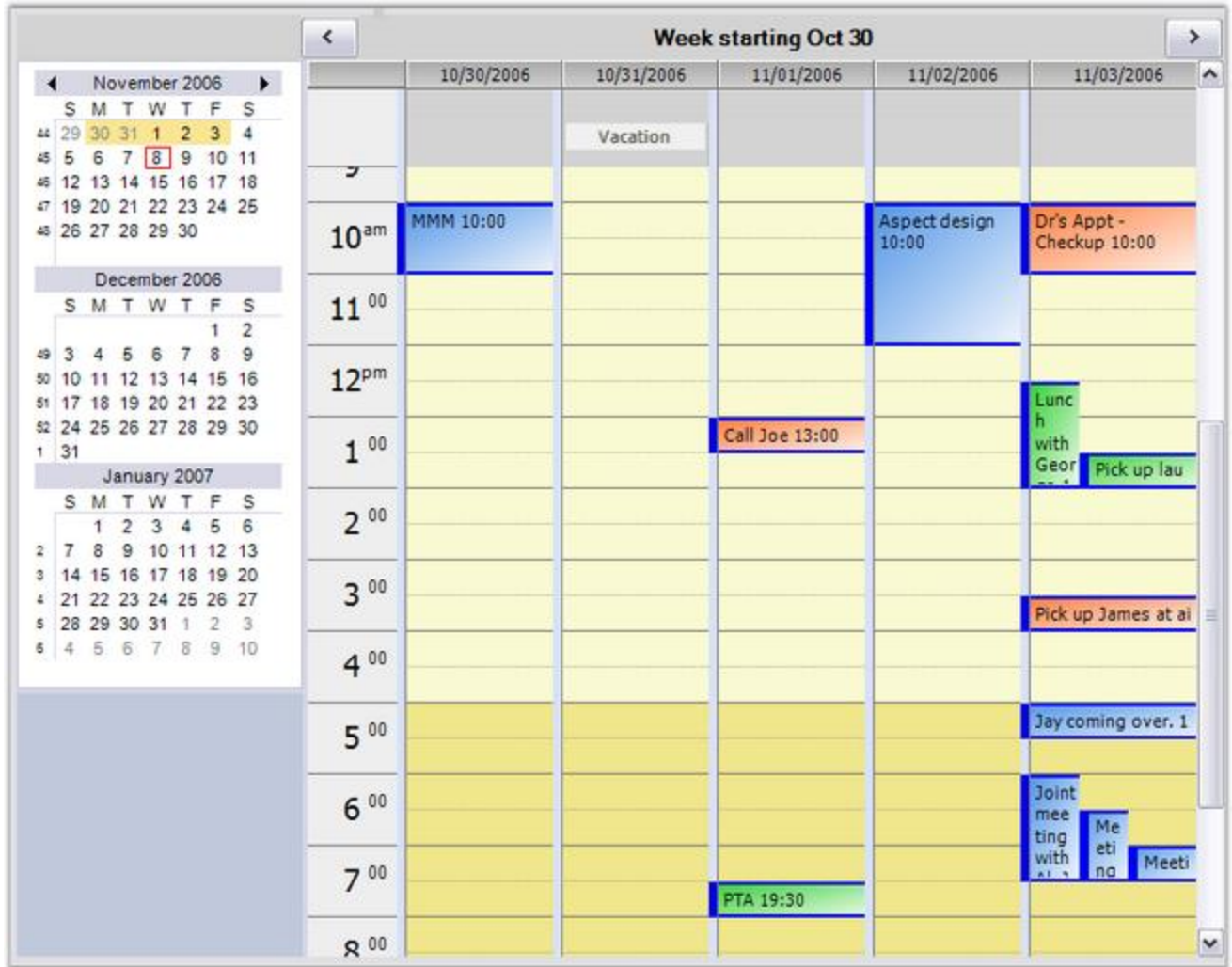


Figure 11: ScheduleControl showing a WorkWeek View

Here is a Week view snapshot.

Notice in the NavigationCalendar on the left, week numbers appear on the left side of each week in the NavigationCalendar. You can optionally turn these numbers off using the **ScheduleControl.Calendar.ShowWeekNumbers** property.

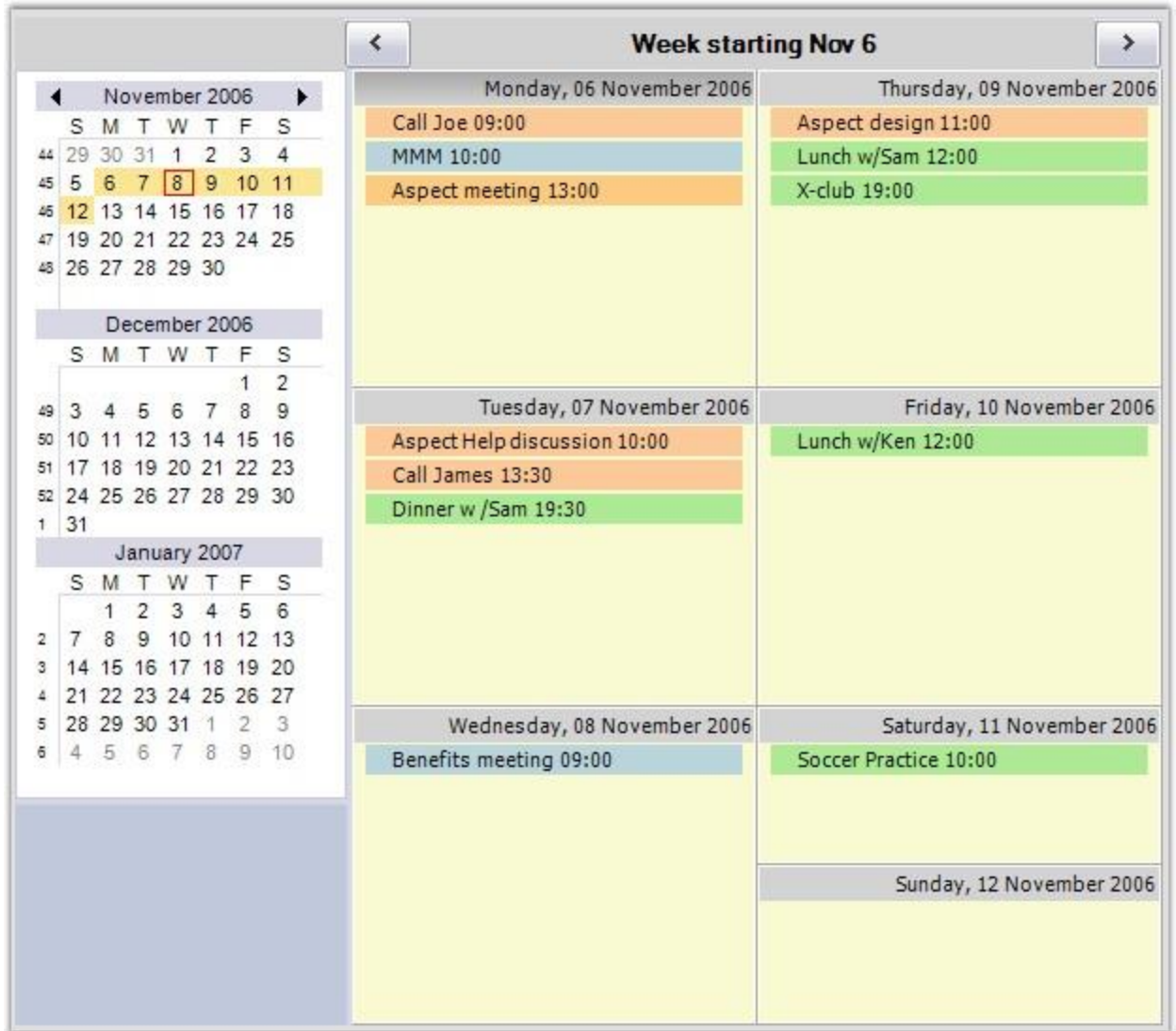


Figure 12: ScheduleControl showing a Week View

The snapshot below shows a ScheduleControl displaying three days. You can select any combination of up to either dates (either contiguous or not) to be displayed in this manner in a Custom view.

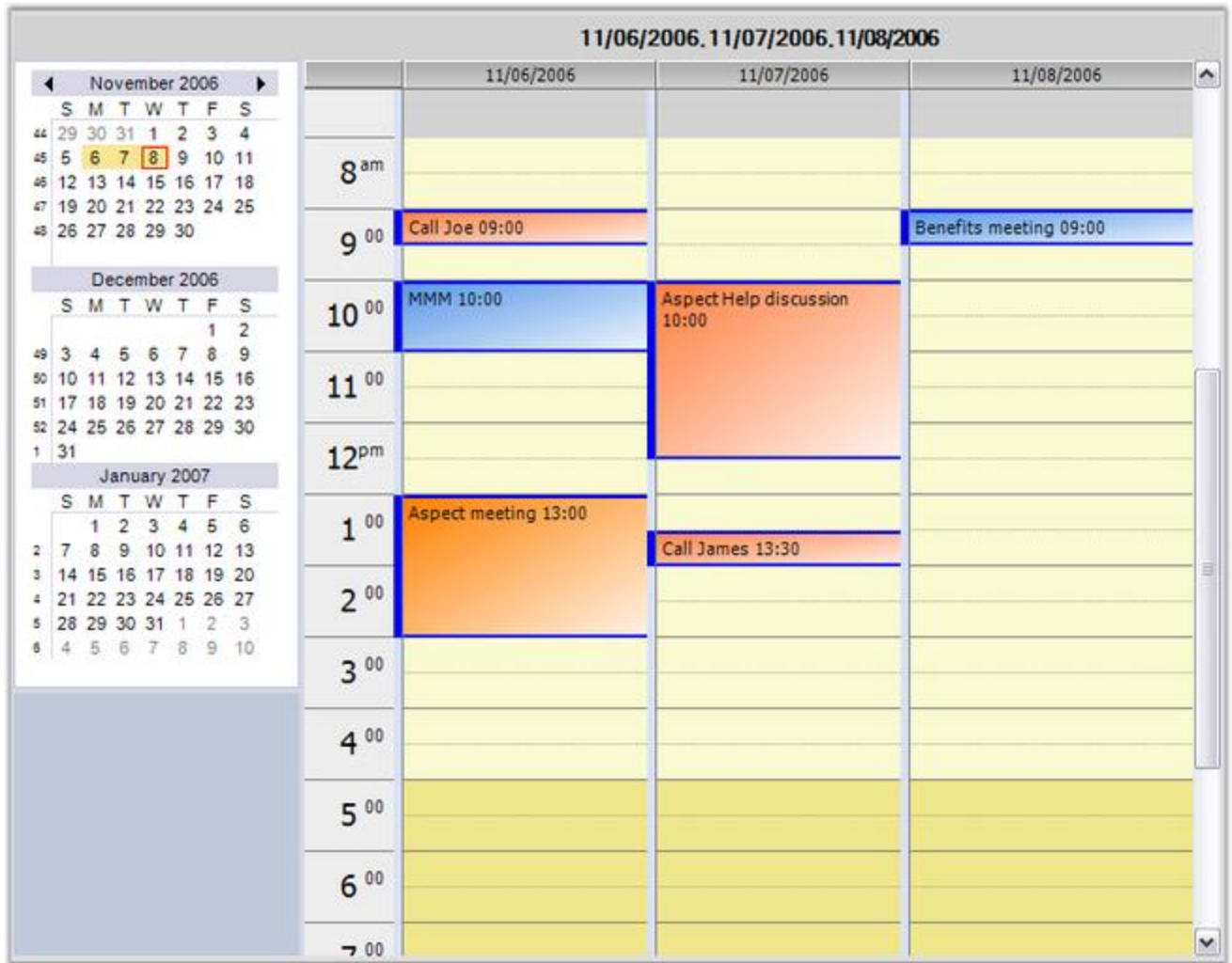


Figure 13: ScheduleControl showing a Custom View

3.2 Class Diagram

Essential Schedule primarily consists of a UserControl-derived class named **ScheduleControl**, which in turn is constructed with the help of four control derived objects, namely, **CaptionPanel**, **NavigationPanel**, **NavigationCalendar** and **ScheduleGrid**.

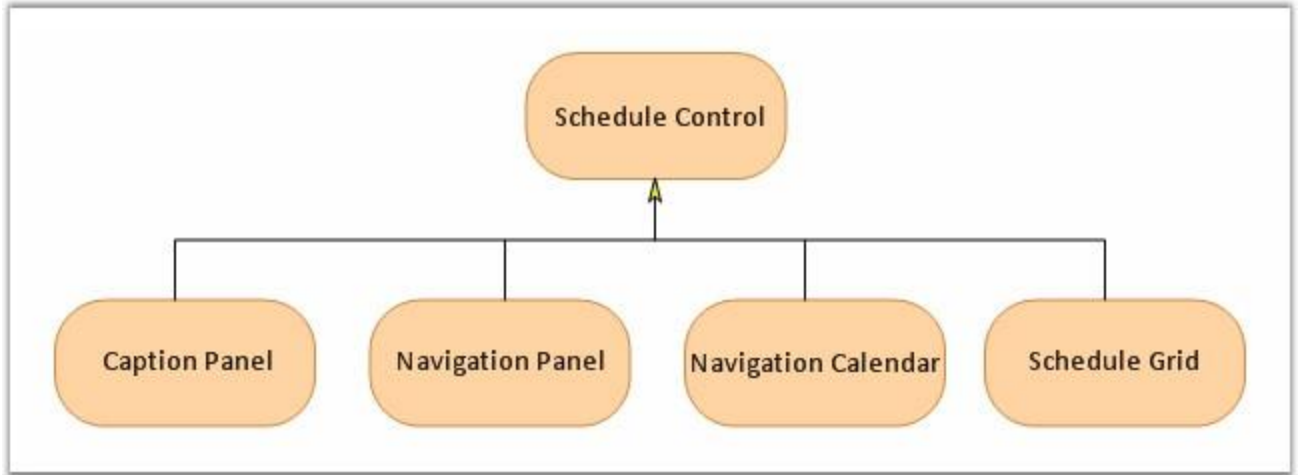


Figure 14: Class Diagram for Essential Schedule Windows

3.3 Tutorial

What You Will Learn

This tutorial will show you how easy it is to get started using Essential Schedule. It will give you a basic introduction to the concepts you need to know as well as some tips and ideas on how to use Essential Schedule in your projects. This tutorial uses step-by-step procedures to introduce you to Essential Schedule.

Windows Application using ScheduleControl and the SimpleScheduleDataProvider Class

In this lesson, you will create a Windows Application that simple displays a **ScheduleControl** on a form and uses the **SimpleScheduleDataProvider** class to provide a data store for the application. The **SimpleScheduleDataProvider** class is an implementation of the **IScheduleDataProvider** interface that **ScheduleControl** requires for its data source. **SimpleScheduleDataProvider.cs** and **SimpleScheduleDataProvider.vb** are files that ship in our **ScheduleSample** sample. As part of this tutorial, you need to copy this file to your tutorial project location.

3.3.1 Lesson 1: Using ScheduleControl and SimpleScheduleDataProvider

In this lesson, you will create a sample Windows Forms application, drop a ScheduleControl onto a form and then hook it up to a SimpleScheduleDataProvider object to serialize schedule items (appointments) to a disk file. Along the way, you will see how to change properties of the ScheduleControl and also serialize these changed settings to a disk file as well. Other tasks discussed in this tutorial are how to use the built-in functionality to add new schedule items, modify existing schedule items and other common tasks.

The following section explains in detail:

3.3.1.1 Lesson: Using ScheduleControl

1. From Visual Studio, go to the **File** menu, select **New** and then click **Project**, to create a new Windows Application named ScheduleSample.

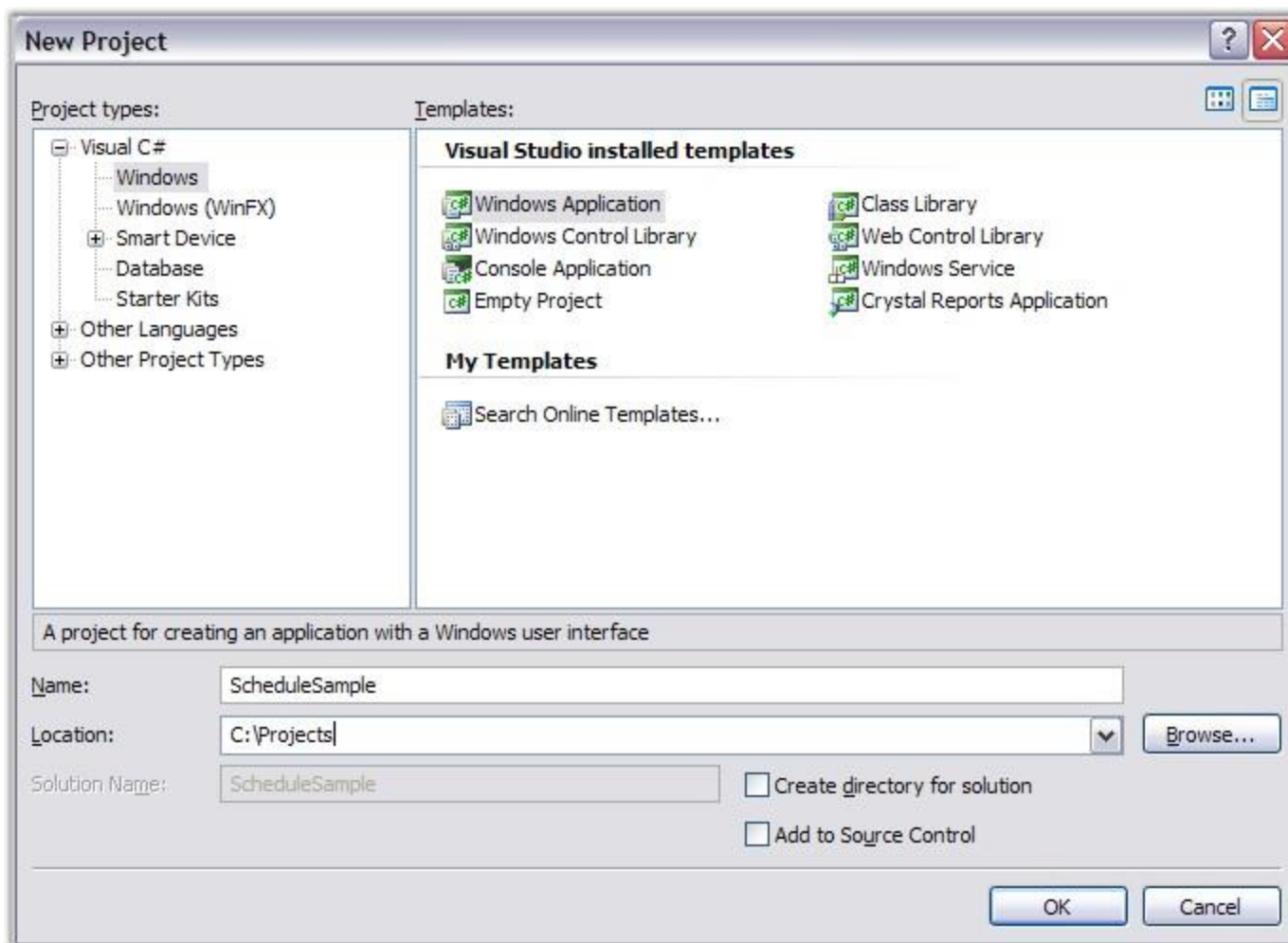


Figure 15: Creating a new Windows Application

2. When the Visual Studio Designer opens, drag the **ScheduleControl** from the Syncfusion Tab onto the Form.

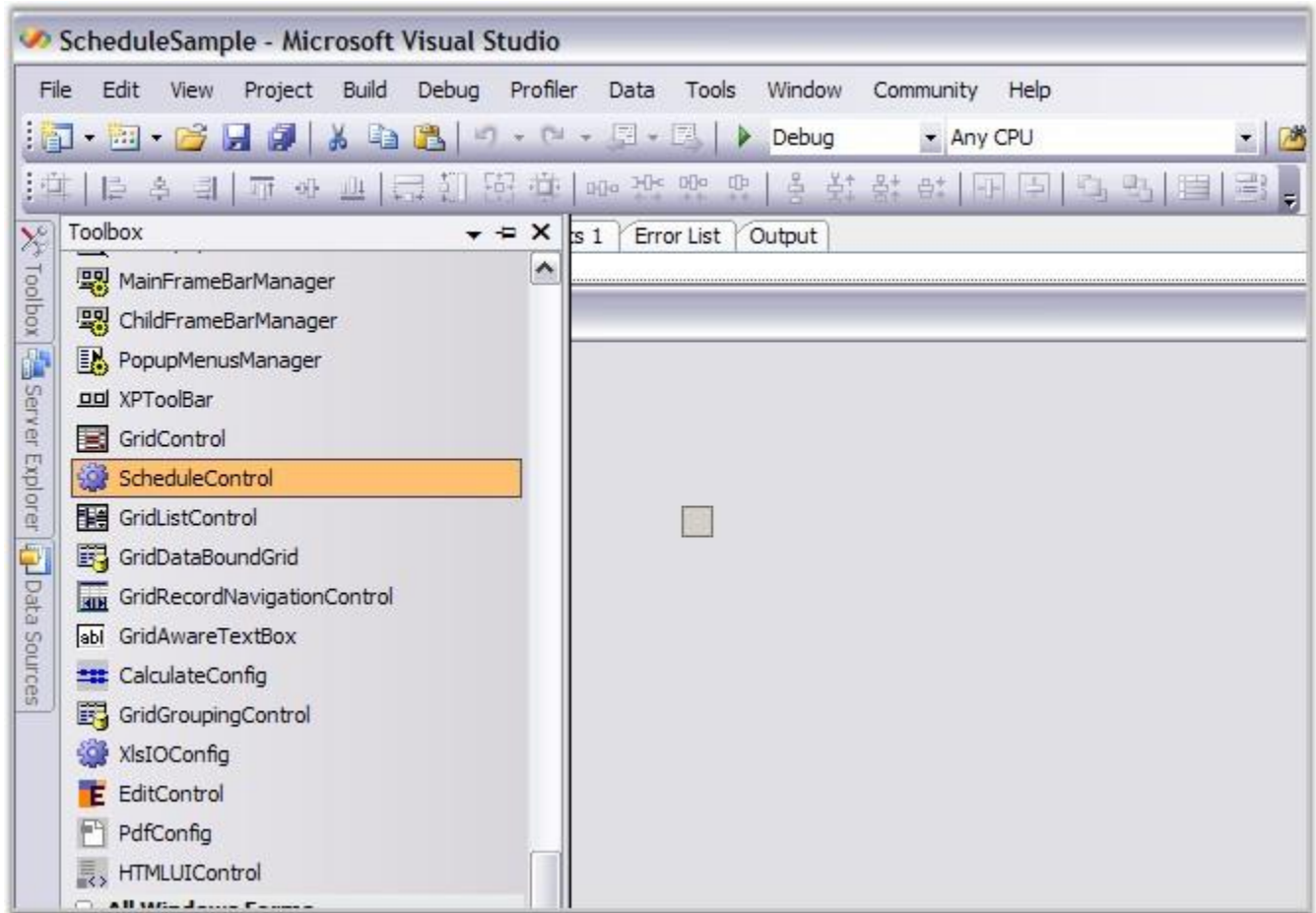


Figure 16: Dragging a ScheduleControl onto the Form

3. The ScheduleControl will show on the design surface. Below is a typical display of this. Notice the **Appearance** property in the property grid. This is the object that has many properties that affect the appearance of the ScheduleControl.

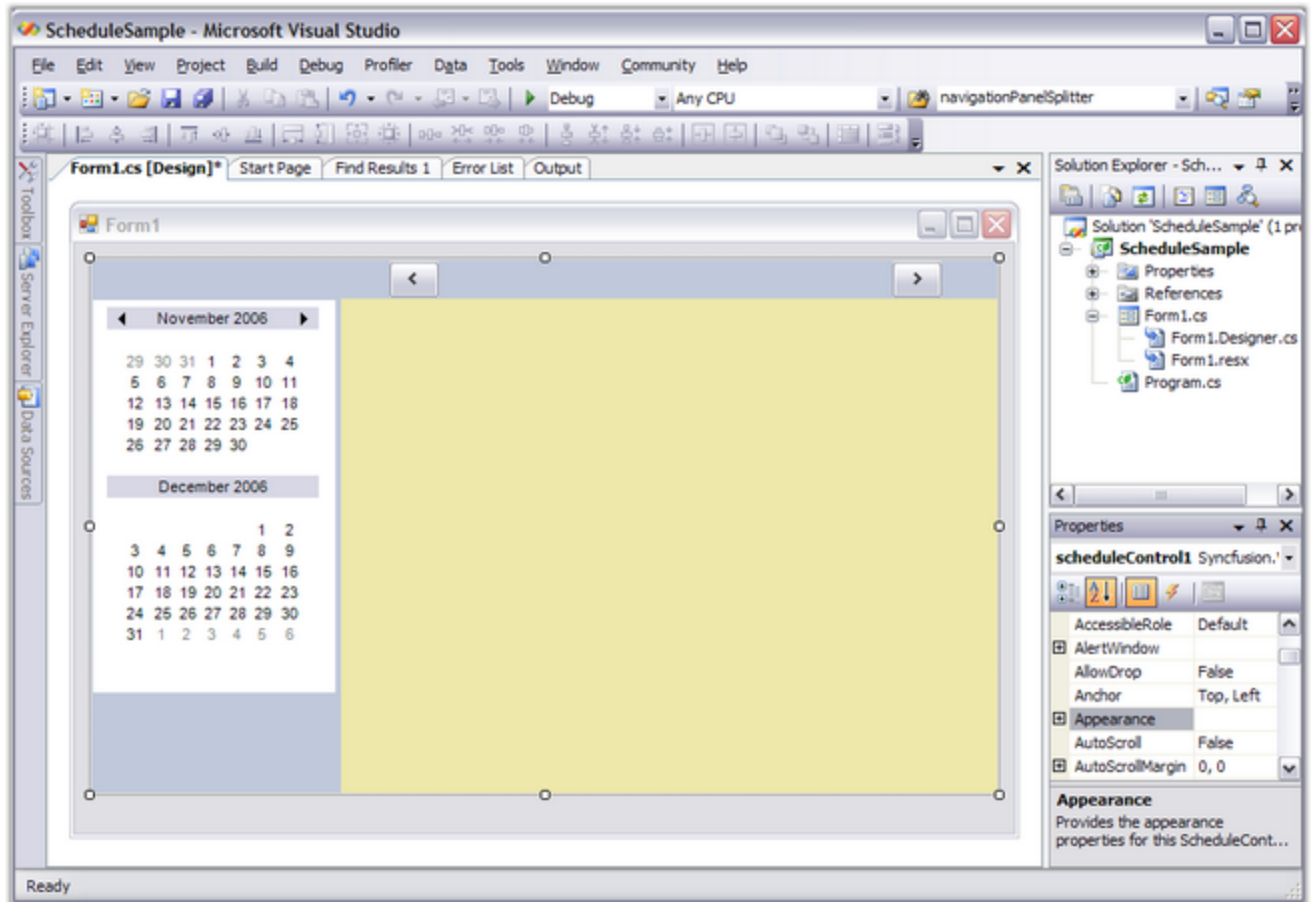


Figure 17: The ScheduleControl on the Design Surface

4. Add a **Form1_Load** handler to the **Form1.cs** file by double-clicking on the **Form** that is not covered by the **ScheduleControl**. This should display a code window showing code like this.

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace ScheduleSample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            |
        }
    }
}
```

Figure 18: Empty Form1_Load method added by double-clicking the Form in the Designer

5. Now add an existing file to this project, SimpleScheduleDataProvider.cs (or impleScheduleDataProvider.vb if you are using VB.NET).

This file defines several classes that implement the interfaces that the ScheduleControl needs to manage the data associated with the appointments that will appear in the calendar.

These interfaces are discussed in detail later in this UserGuide.

For now, just use the implementation provided in the SimpleScheduleDataProvider.cs file. This file ships as part of the **Syncfusion\Essential Studio\5.x.x\Windows\Schedule.Windows\Samples\2.0\ScheduleSample** sample.

Drill down to this folder, and add this file to your project by using the Solution Explorer window as shown here.

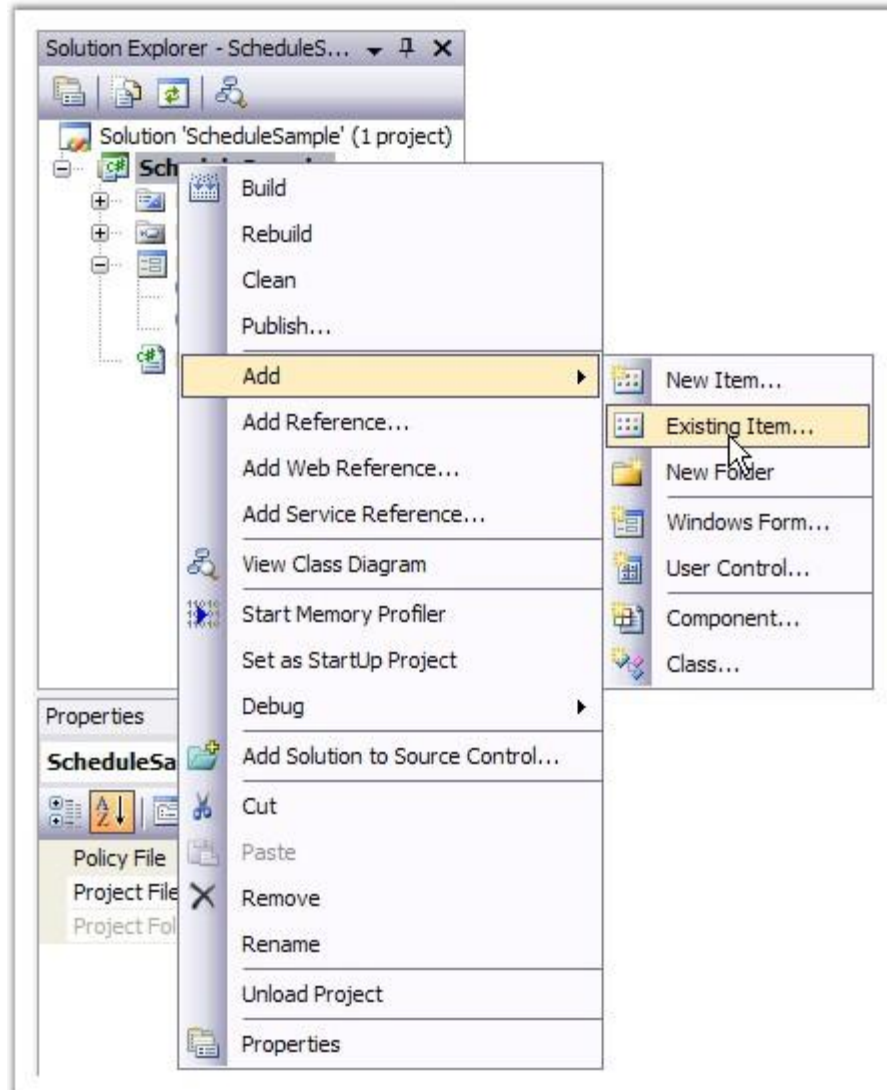


Figure 19: Menu selection showing how to add an existing file to the Project

6. Here, you can find the SimpleScheduleDataProvider.cs file in the **\Syncfusion\Essential Studio\5.x.x\Windows\Schedule.Windows\Samples\2.0\ScheduleSample\CS** folder. Drill down to this folder and add this file to our project.

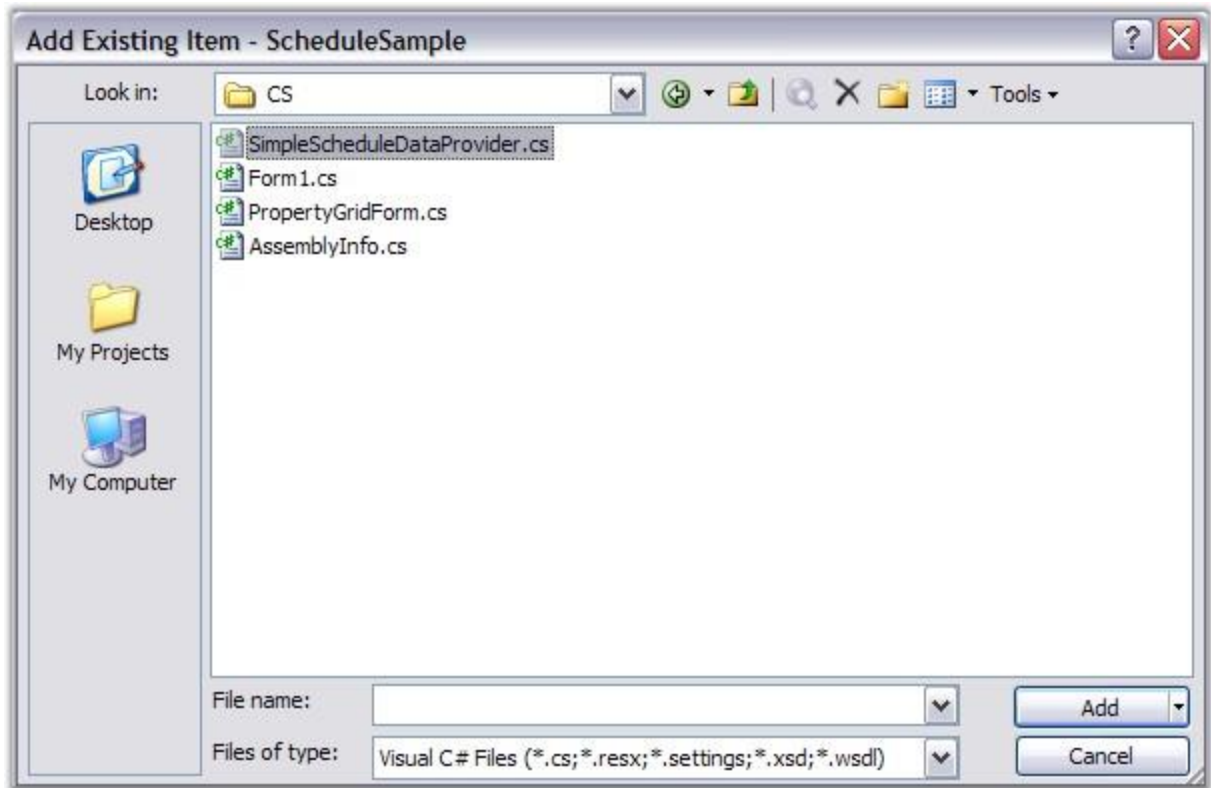


Figure 20: Adding the SimpleScheduleDataProvider.cs file to the Project

7. After adding the file containing our **SimpleScheduleDataProvider.cs** code, add some code to your **Form.cs** to provide data support to your ScheduleControl.
 - i. The first thing to do is to add a using statement to allow us to reference the class names in the SimpleScheduleDataProvider.cs file without adding the Namespace used in that file.

The other is added in the **Form_Load** code to hook up the data support.
 - ii. In the Form_Load, create an instance of the DataProvider and a MasterList to hold the data.
 - iii. Then set some properties to provide a filename, the **ScheduleViewType** for the initial display and the **DataSource** property for your ScheduleControl.
 - iv. Copy this code to your Form1.cs file. (If you are not using the 2.0 Framework, remove the partial keyword.)

[C#]

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Text;
using System.Windows.Forms;
using Syncfusion.Windows.Forms.Schedule;
using GridScheduleSample;

namespace ScheduleSample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            SimpleScheduleDataProvider data = new
            SimpleScheduleDataProvider();
            data.MasterList = new SimpleScheduleItemList();
            data.FileName = "default.schedule";
            this.scheduleControl1.ScheduleType = ScheduleViewType.Month;
            this.scheduleControl1.DataSource = data;
        }
    }
}
```

8. Now press **F5** key to compile and run your application. A screen similar to this one should appear.

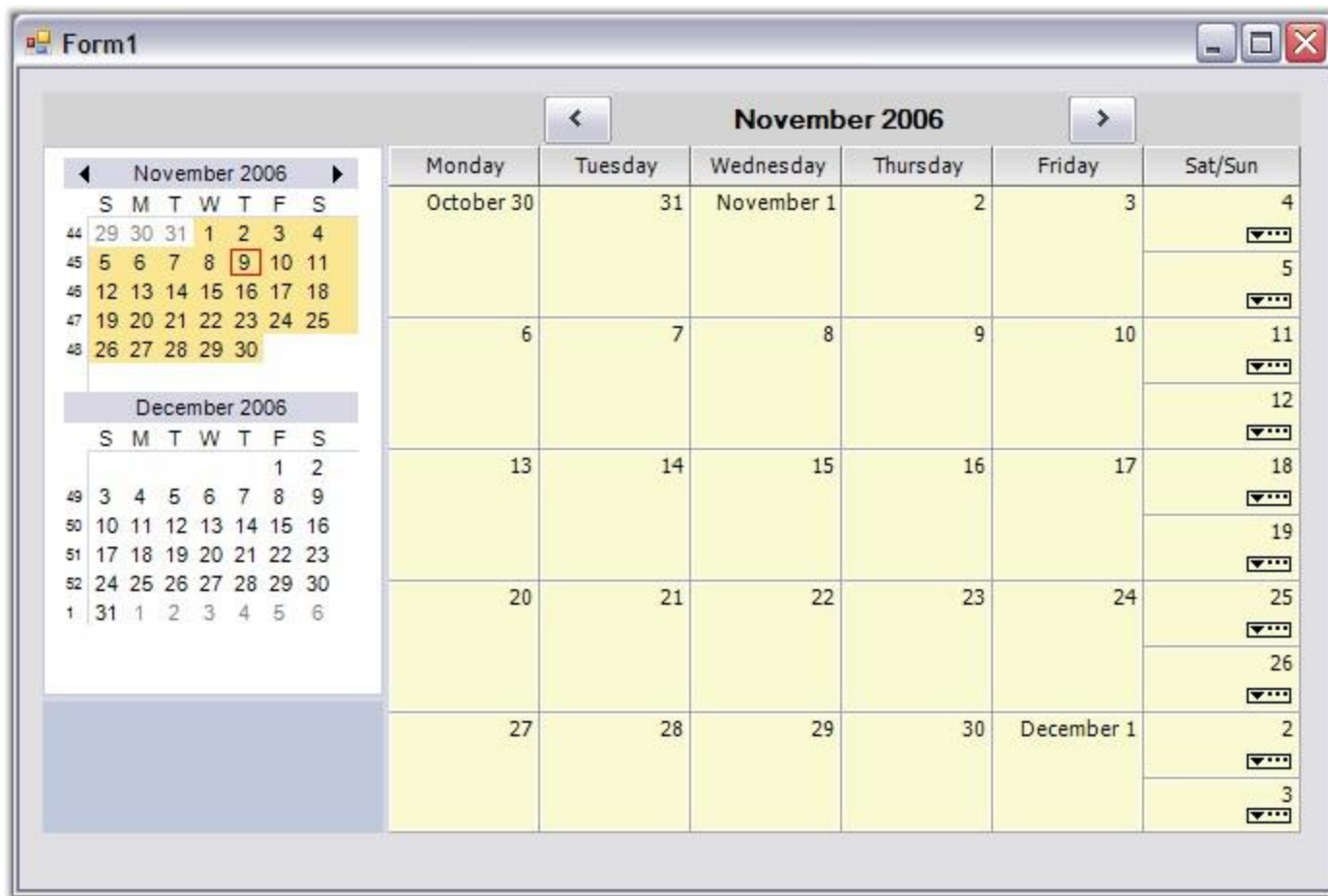


Figure 21: A Month View Schedule displayed in the Sample Application

9. To change the Month view to a Day view, right-click the ScheduleGrid area of the ScheduleControl to display a ContextMenu and select **Day**.

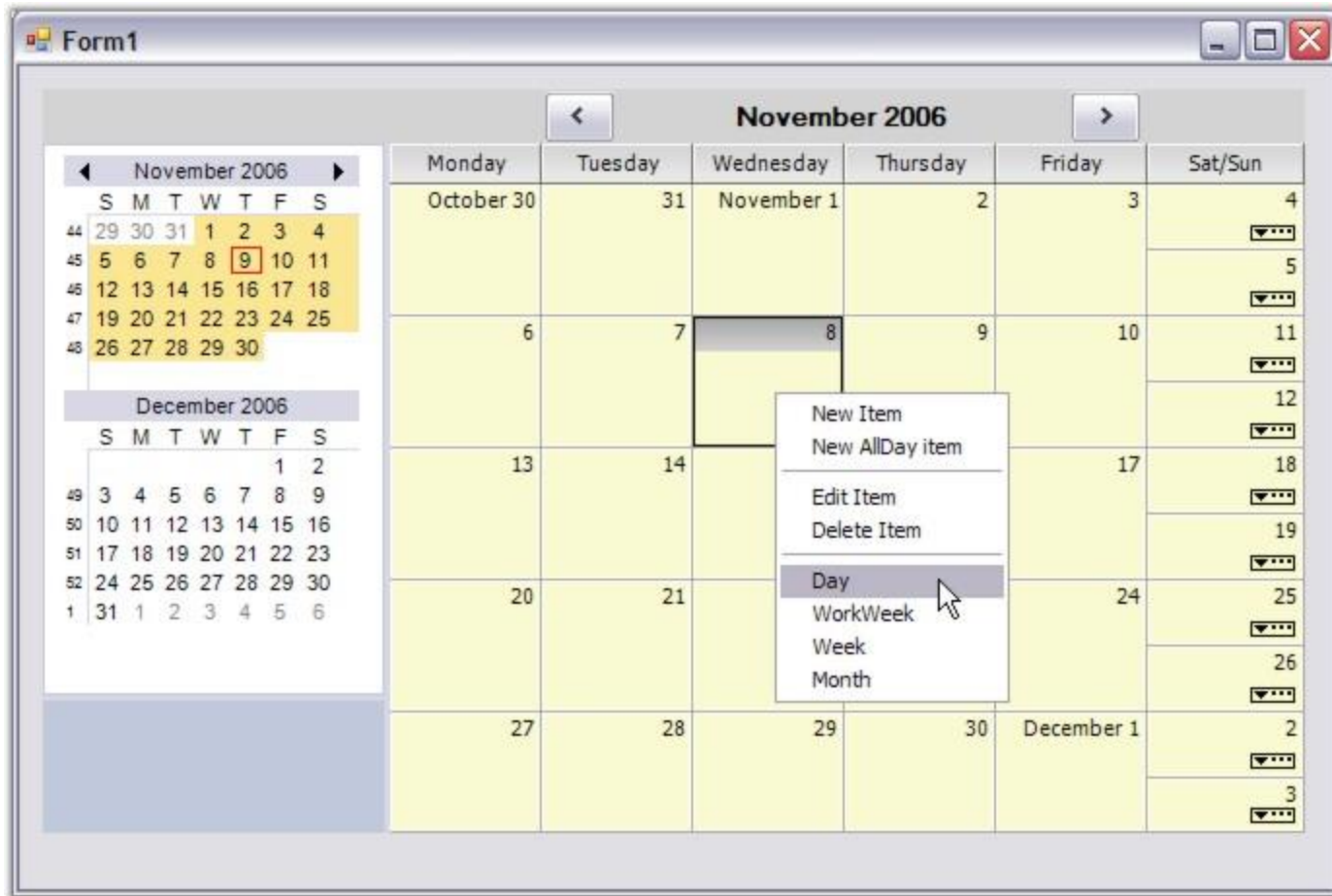


Figure 22: Using the ContextMenu to change the ScheduleViewType

10. Here is the Day view that appears after the execution of the ContextMenu selection done in step 9.

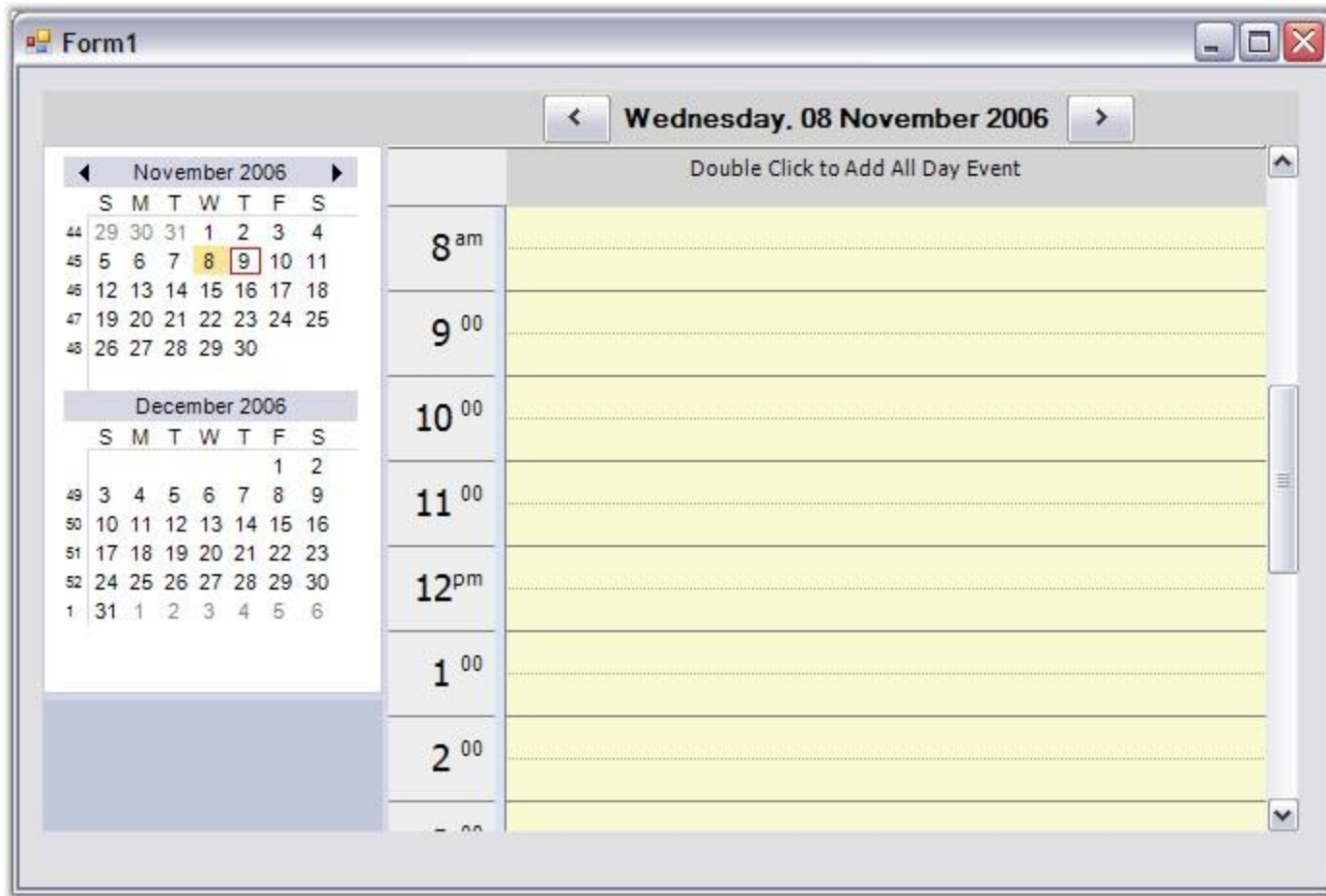


Figure 23: The Day view resulting from the ContextMenu

11. Double-click one of the time slots on the ScheduleGrid in the ScheduleControl. This action will display a new appointment screen where you can enter a new schedule item as shown below.

Enter Appointment

Subject: Dr Appt

Location: Ashville Med Center Label: Important

Start Time: 11/ 9/2006 10:00 AM ☐ All Day Event

End Time: 11/ 9/2006 11:00 AM (1 hours)

☐ Reminder: 0 Show time as: Busy

Regular checkup - make sure to bring insurance card

Save and Close Cancel

Figure 24: A filled-in New Appointment Screen

- Clicking the **Save** and **Close** button on the Appointment screen will re-display the Day view ScheduleControl with the new appointment displayed. If you hover over the appointment in the ScheduleGrid, a tooltip will display as shown below.

The screenshot shows a Windows Forms application window titled "Form1". Inside the window, there is a calendar control on the left and a schedule control on the right. The calendar is set to November 2006, with the 8th and 9th highlighted. The schedule control shows a timeline from 8 am to 2 pm. An appointment titled "Dr Appt 10:00" is scheduled for 10:00. A tooltip is displayed over the appointment, showing the text "Dr Appt" and "Regular checkup - make sure to bring insurance card".

Figure 25: The New Appointment with a ToolTip Displayed

13. Click the **Close** button on the form system menu on the upper-right corner of the form. Since the data has been modified in this ScheduleControl, a dialog will appear as below, asking whether you want to save these changes to a disk file. Click **Yes** to save the changes.

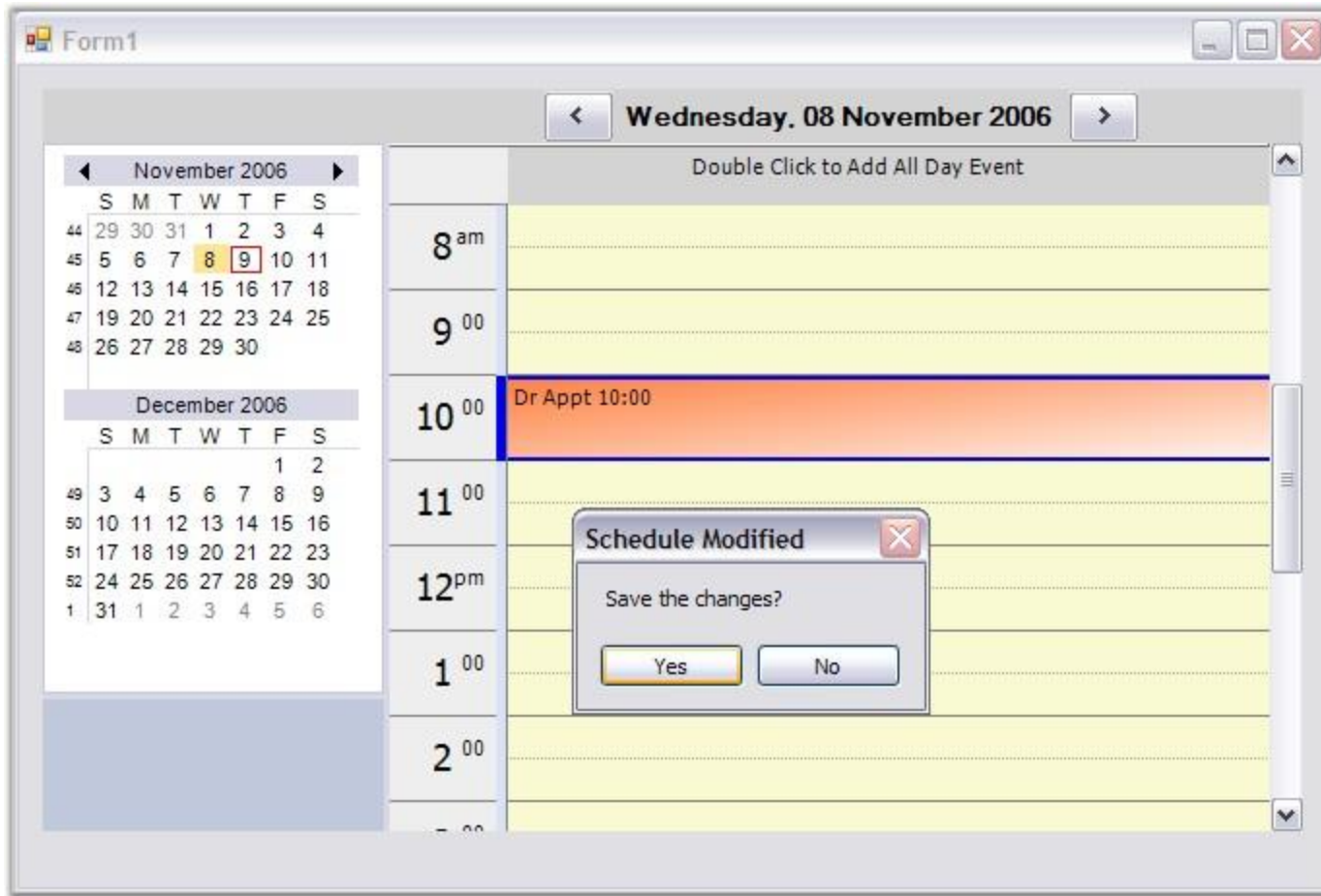


Figure 26: A Prompt is displayed as you try to close the Modified Form

14. Next modify our **Form_Load** code to conditionally reload the saved data if the file is present on the disk. Here is the new code. Copy this code to your Form1.cs file. Notice that you have added a "using" statement to reference the System.IO namespace in addition to the new code in the Form1_Load. (If you are not using the 2.0 Framework, remove the partial keyword)

[C#]

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Syncfusion.Windows.Forms.Schedule;
using GridScheduleSample;
using System.IO;

namespace ScheduleSample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            SimpleScheduleDataProvider data;
            if (File.Exists("default.schedule"))
            {
                data =
                SimpleScheduleDataProvider.LoadBinary("default.schedule");
                data.FileName = "default.schedule";
            }
            else
            {
                data = new SimpleScheduleDataProvider();
                data.MasterList = new SimpleScheduleItemList();
                data.FileName = "default.schedule";
            }
            this.scheduleControl1.ScheduleType = ScheduleViewType.Month;
            this.scheduleControl1.DataSource = data;
        }
    }
}
```

15. As our last step, compile and run the application again. The Month view should reappear but, this time the appointment you added earlier should appear.

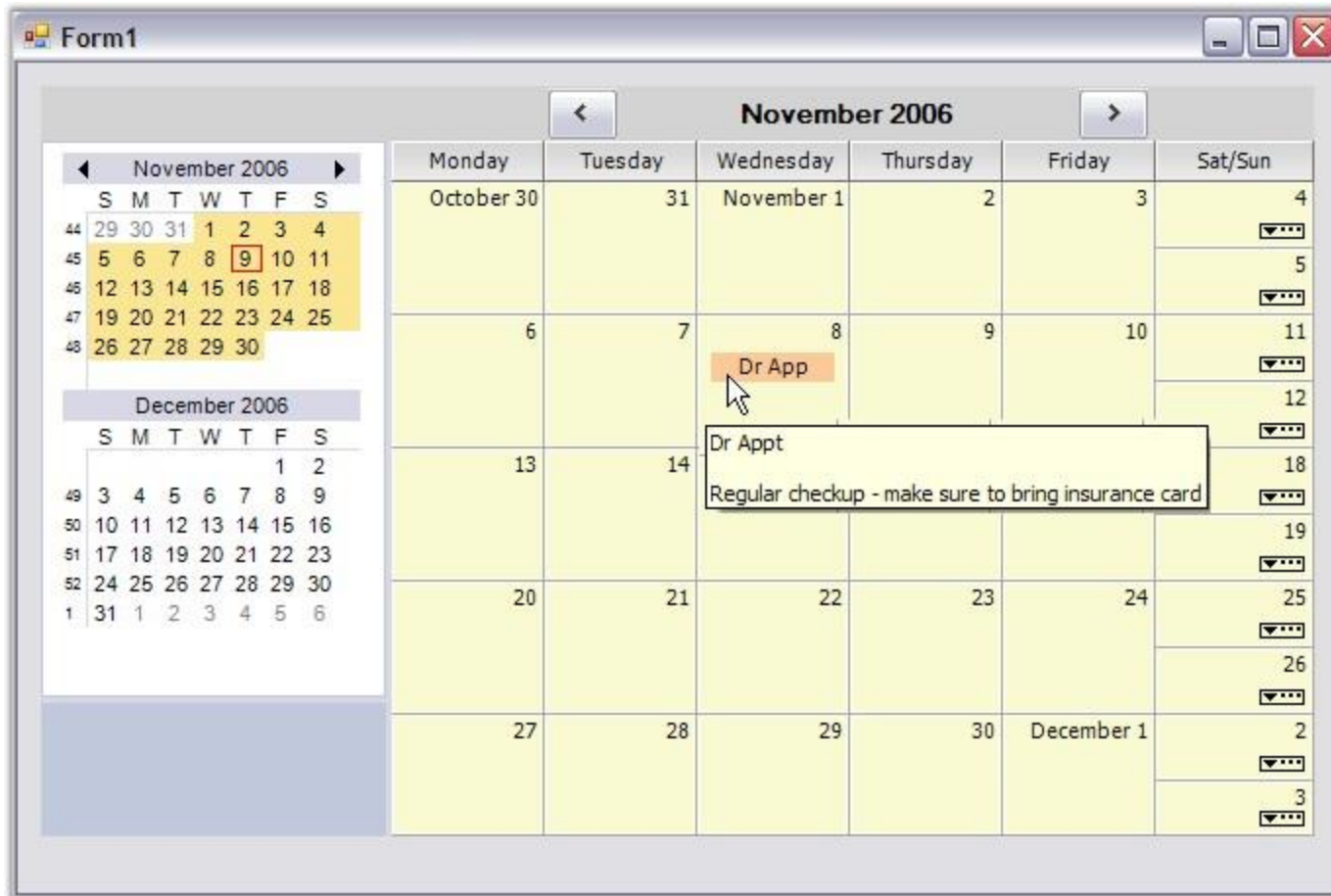


Figure 27: The Month View showing the previously saved Appointment

3.4 Data used by ScheduleControl

There are generally two types of data required by ScheduleControl. Most of the data is what we described as **Appointments data**. This data includes the time, subject, body, and so on. of each of the actual appointments.

The second type of data we refer to as the **DropLists data**. This data consists of the several option lists that go into describing the actual appointment data. For example, each appointment may have a marker associated with it that indicates something of the nature of the appointment like whether it is business, personal, a must-attend, etc. This second type of data is more like schema data, as it suggests the content options of the actual Appointments data.

The ScheduleControl does all its data access through interfaces. To support custom data objects, you would have your data objects implement these particular interfaces that are discussed in the following sections. In addition, included in the Essential Schedule library are base classes that implement these required interfaces. So, you can also create data sources for the ScheduleControl by deriving these base classes. The SimpleScheduleDataProvider classes that were used in the Tutorial are derived from these base classes.

Base Classes

- **ScheduleDataProvider Class:** provides an empty implementation of the IScheduleDataProvider.
The implementation is done through virtual methods. You can then derive this class and through its overrides, set up an IScheduleDataProvider. See the SimpleScheduleDataProvider class in the ScheduleSample sample.
- **ScheduleAppointmentList Class:** provides an implementation of IScheduleAppointmentList and is essentially a wrapper class for an ArrayList that holds ScheduleAppointments
- **ScheduleAppointment Class:** provides an implementation of IScheduleAppointment and defines the objects that represent appointments in the ScheduleControl.
- **LookUpObjectList Class:** strongly typed ArrayList that holds list option values that are used in the new appointment form.
- **LookUpObject Class:** wrapper class for list choices that can have a valueMember, displayMember and colorMember associated with them.

The lists for the ShowTime and Label options on the Appointment forms use these objects.

Interfaces

- **IScheduleDataProvider Interface:** provides the framework for providing schedule item data to the ScheduleControl.
- **IScheduleAppointmentList Interface:** serves as a collection of ISchedule objects.
- **IScheduleAppointment Interface:** defines individual schedule items.
- **ILookUpObjectList Interface:** serves as a collection of IlookUpObjects.
- **ILookUpObject Interface:** enables Choice lists within the ScheduleControl, that are used to provide possible schedule item information (like location or a reminder), to have a

ValueMember / DisplayMember associated with them, as well as a color that will be used in drop-downs showing these lists.

Value members are normally the values serialized to data stores.

3.4.1 ScheduleData Base Classes

The **ScheduleControl** gets its data through its DataSource property, an IScheduleDataProvider object. So, it is this IScheduleDataProvider interface (and several other associated interfaces) that gives you the ability and facility to provide data to the ScheduleControl. To simplify this process of providing data, Essential Schedule also exposes these interfaces as base classes that include some pre-determine droplist settings that allow you to use the ScheduleControl with less coding work. But, you do have the option of working directly through the interfaces to construct your own data provider for the ScheduleControl.

The **Essential Schedule** library contains several base classes that implement the various data interfaces required by the ScheduleControl. These base classes use virtual methods which, you can override to provide a concrete data implementation. The classes in the SimpleScheduleDataProvider file that is shipped with the samples and used in the Tutorial section of this User Guide are derived from the ScheduleData base classes. Check out the shipped sample that uses the SimpleScheduleDataProvider as a data source for ScheduleControl.

The following sections discuss these ScheduleData base classes in more detail.

3.4.1.1 The Appointments Data

Here are the ScheduleData base classes that provide the Appointments data used by ScheduleControl. For code details of deriving these ScheduleData base classes to implement a data provider for the ScheduleControl, please see the SimpleScheduleDataProvider code file that ships as part of the ScheduleSample sample.

ScheduleAppointment Class

ScheduleAppointment is the class that defines the objects that represent appointments in the Schedule Control. This class implements **IScheduleAppointment** to provide an object to hold the concrete data associated with appointments. You can either derive this class or implement IScheduleAppointment yourself to extend or modify the information managed by the ScheduleAppointment class. Here are the properties exposed in ScheduleAppointment.

- **UniqueID**: gets or sets a unique integer associated with this item

- **Owner:** gets or sets an integer that can be used to identify the owner (if any) of this item
- **StartTime:** gets or sets the start time for this item
- **EndTime:** gets or sets the end time for this item
- **Subject:** gets or sets a text string identifying the topic of this item
- **Content:** gets or sets a text string holding the details or comments for this appointment item
- **AllDay:** gets or sets whether this appointment is an all-day appointment
- **LabelValue:** gets or sets an integer categorizer value for this item
- **MarkerValue:** gets or sets an integer categorizer value for this item
- **Reminder:** gets or sets whether you want a reminder event raised when the StartTime of this item gets close
- **ReminderValue:** gets or sets the type of reminder event raised when the StartTime of this item gets close
- **LocationValue:** gets or sets a string associated with this item
- **Version:** gets in integer format the version number (used to support data format versioning)
- **Tag:** gets or sets an arbitrary object associated with this item
- **Dirty:** gets or sets whether this item has been modified
- **IgnoreChanges:** gets or sets whether changes to this item affect the Dirty property

ScheduleAppointmentList Class

ScheduleAppointmentList is a collection of **IScheduleAppointments** that serves as the data for the Schedule Control. This class is a wrapper class for an **ArrayList** and implements **IComparer** to order this list by the item's **StartTime**. If two items start at the same time, then the **EndTime** is used as well to determine the order. Longer appointments rank higher. Here are the properties and methods exposed in **ScheduleAppointmentList**.

[C#]

```
/// Gets or sets the i-th IScheduleAppointment in this list.
public virtual IScheduleAppointment this[int i];

/// Gets the number of IScheduleAppointments in this list.
public virtual int Count

/// Sorts this list on the IScheduleAppointment.StartTime property.
public virtual void SortStartTime()

/// Adds an IScheduleAppointment to this list.
/// item - The IScheduleAppointment to be added.
public virtual void Add(IScheduleAppointment item)

/// Inserts an IScheduleAppointment into this list.
/// index - The position in the list where the item is to be inserted.
/// item - The IScheduleAppointment to be inserted.
public virtual void Insert(int index, IScheduleAppointment item)

/// Removes an IScheduleAppointment from this list.
/// item - The IScheduleAppointment to be removed.
public virtual void Remove(IScheduleAppointment item)

/// Removes an IScheduleAppointment from this list.
/// index - The position of the item to be removed.
public virtual void RemoveAt(int index)

/// Returns the position of the specified item within this list.
/// item - The search item.
public virtual int IndexOf(IScheduleAppointment item)

/// Returns a new ScheduleAppointment populated with default values.
public virtual IScheduleAppointment NewScheduleAppointment()
```

ScheduleDataProvider Class

ScheduleDataProvider has two functional roles.

One is to implement **IScheduleDataProvider** in a virtual manner so that derived classes can provide concrete implementations through virtual overrides. The **IScheduleDataProvider** virtual methods exposed in **ScheduleDataProvider** have empty implementations, so you are required to derive this class to use it.

The second role is to provide the **DropList** data. For this second role, the `ScheduleDataProvider` does provide concrete implementations for the virtual methods it exposes. So, in your derived class, you would have populated droplists without doing further work, though you can choose to customize these droplists through virtual overrides. Here is a list of the stub methods exposed by `ScheduleDataProvider` in its first role.

[C#]

```
/// Return an IScheduleAppointmentList holding the schedule items for
the given date.
public virtual IScheduleAppointmentList GetScheduleForDay(DateTime day)

//// Return an IScheduleAppointmentList holding the schedule items
between the given dates.
public virtual IScheduleAppointmentList GetSchedule(DateTime startDate,
DateTime endDate)

/// Return an IScheduleAppointmentList holding the schedule items for a
particular owner on the given date.
public virtual IScheduleAppointmentList GetScheduleForDay(DateTime day,
int owner)

/// Return an IScheduleAppointmentList holding the schedule items for a
particular owner between the given dates.
public virtual IScheduleAppointmentList GetSchedule(DateTime startDate,
DateTime endDate, int owner)

/// Saves any modified ScheduleAppointments.
public virtual void CommitChanges()

/// Gets or sets whether CommitChanges is called when the
ScheduleControl is disposed.
public SaveOnCloseBehavior SaveOnCloseBehaviorAction

/// Gets or sets whether data source is modified or not.
public virtual bool IsDirty

/// Returns a new ScheduleAppointment populated with default values.
public virtual IScheduleAppointment NewScheduleAppointment()

/// Adds a ScheduleAppointment to the list.
public virtual void AddItem(IScheduleAppointment item)

/// Removes a ScheduleAppointment from the list.
public virtual void RemoveItem(IScheduleAppointment item)
```

Here are the methods and properties used as part of the ScheduleDataProvider's second role, providing the DropList data. The following is the actual implementation code which gives an indication of the exposed functionality.

[C#]

```
/// Provides default droplists for entering IScheduleAppointment data.
/// You can override this method to provide customized droplists.
public virtual void InitLists()
{
    labelList = new ListObjectList();
    labelList.Add(new ListObject(0, "None", Color.White));
    labelList.Add(new ListObject(1, "Important",
        Color.FromArgb(255, 128, 64)));
    labelList.Add(new ListObject(2, "Business",
        Color.FromArgb(86, 152, 233)));
    labelList.Add(new ListObject(3, "Personal",
        Color.FromArgb(57, 210, 53)));
    labelList.Add(new ListObject(4, "Vacation",
        Color.FromArgb(199, 198, 182)));
    labelList.Add(new ListObject(5, "Must Attend",
        Color.FromArgb(255, 128, 0)));
    labelList.Add(new ListObject(6, "Travel Required",
        Color.FromArgb(0, 255, 255)));
    labelList.Add(new ListObject(7, "Needs Preparation",
        Color.FromArgb(171, 171, 88)));
    labelList.Add(new ListObject(8, "Birthday",
        Color.FromArgb(186, 117, 255)));
    labelList.Add(new ListObject(9, "Anniversary",
        Color.FromArgb(255, 128, 64)));
    labelList.Add(new ListObject(10, "Phone Call",
        Color.FromArgb(255, 128, 64)));

    markerList = new ListObjectList();
    //same as no Mark Color
    markerList.Add(new ListObject(0, "Free", Color.FromArgb(50,
        Color.RoyalBlue)));
    markerList.Add(new ListObject(1, "Tentative", Color.FromArgb(255,
        206, 206)));
    markerList.Add(new ListObject(2, "Busy", Color.FromArgb(0, 0, 242)));
    markerList.Add(new ListObject(3, "Out of Office",
        Color.FromArgb(128, 0, 64)));

    reminderList = new ListObjectList();
    reminderList.Add(new ListObject(0, "0 minutes", Color.White));
    reminderList.Add(new ListObject(1, "5 minutes", Color.White));
    reminderList.Add(new ListObject(2, "10 minutes", Color.White));
    reminderList.Add(new ListObject(3, "15 minutes", Color.White));
}
```

```
reminderList.Add(new ListObject(4, "30 minutes", Color.White));
reminderList.Add(new ListObject(5, "1 hour", Color.White));
reminderList.Add(new ListObject(6, "2 hours", Color.White));
reminderList.Add(new ListObject(7, "3 hours", Color.White));
reminderList.Add(new ListObject(8, "4 hours", Color.White));

this.locationList = new ListObjectList();
locationList.Add(new ListObject(0, "", Color.White));
locationList.Add(new ListObject(1, "RoomB", Color.White));
locationList.Add(new ListObject(2, "RoomC", Color.White));
locationList.Add(new ListObject(3, "RoomD", Color.White));
locationList.Add(new ListObject(4, "RoomE", Color.White));
}

/// Returns the list for the LabelValue options.
public virtual ILookupObjectList GetLabels()
{
    return LabelList;
}

/// Gets or sets the list for the LabelList options.
protected ListObjectList LabelList
{
    get{return labelList;}
    set{labelList = value;}
}

/// Returns the list for the ReminderValue options.
public virtual ILookupObjectList GetReminders()
{
    return ReminderList;
}

/// Gets or sets the list for the ReminderValue options.
protected ListObjectList ReminderList
{
    get{return reminderList;}
    set{reminderList = value;}
}

/// Returns the list for the MarkerValue options.
public virtual ILookupObjectList GetMarkers()
{
    return MarkerList;
}
```

```
/// Gets or sets the list for the MarkerValue options.
protected ListObjectList MarkerList
{
    get{return markerList;}
    set{markerList = value;}
}

/// Returns the list for the LocationValue options.
public virtual ILookupObjectList GetLocations()
{
    return LocationList;
}

/// Gets or sets the list for the LocationValue options.
protected ListObjectList LocationList
{
    get{return locationList;}
    set{locationList = value;}
}

/// Returns the list for the Owner options.
public virtual ILookupObjectList GetOwners()
{
    return OwnerList;
}

/// Gets or sets the list for the Owner options.
protected ListObjectList OwnerList
{
    get{return ownerList;}
    set{ownerList = value;}
}
```

3.4.1.2 The DropLists

The second type of data required of the ScheduleControl is the **DropList** data. You have seen a concrete implementation of providing this DropList data in the [The Appointments Data](#) discussion. Two classes that can provide such data are listed below.

- **ListObjectClass**

The **ListObject** is a wrapper class for list choices that can have a ValueMember, DisplayMember and ColorMember associated with them. The class is an implementation of the **IListObject** that exposes the IListObject functionality as virtual members. This allows you to implement the IListObject by deriving the ListObject and overriding virtual properties. Here are the properties exposed by this class.

[C#]

```
/// An integer that is stored in the data objects to represent this
object.
public virtual int ValueMember

/// A string that is used when this object is displayed.
public virtual string DisplayMember

/// A color associated with this object.
public virtual Color ColorMember
```

- **ListObjectList**

The **ListObjectList** is a strongly-typed **ArrayList** that holds a collection of **ListObjects**. The class is derived from **ArrayList** and implements both **ITypedList** and **IlistListObjectList**. Here are the properties and methods exposed in this class.

[C#]

```
/// Returns the property descriptors for each property in
ListObject.</returns>
public PropertyDescriptorCollection
GetItemProperties(PropertyDescriptor[] listAccessors)

/// Returns a list name</returns>
public string GetListName(PropertyDescriptor[] listAccessors)

/// Gets or sets the i-th item in the list.
public new ILookupObject this[int i]
```

3.4.2 IScheduleData Interfaces

The **ScheduleControl** gets its data through its **DataSource** property, an **IScheduleDataProvider** object.

So, it is this **IScheduleDataProvider** interface (and several other associated interfaces) that gives you the ability and facility to provide data to the **ScheduleControl**.

This section discusses the actual interfaces required to provide data to the **ScheduleControl**. If you need the access your own custom datastore, then you can create objects that implement these interfaces on which the **ScheduleControl** relies to provide data from your custom datastore.

If you just need a local disk file datastore, then using the implementation provided by the classes in the **SimpleScheduleDataProvider** file that is shipped with the samples, may serve your purpose.

You also have the option of deriving the `ScheduleData` base classes to provide custom data to the `ScheduleControl`. But, implementing the required interfaces directly will give you the most flexibility.

There are five interfaces that you can use to provide data for a `ScheduleControl`. There are two 'object' interfaces, **`IScheduleAppointment`** and **`ILookUpObject`**. These interfaces are primarily wrappers for a collection of properties.

`IScheduleAppointment` wraps individual appointment data. **`ILookUpObject`** wraps the items you can see in droplists.

There are two 'list' interfaces, **`IScheduleAppointmentList`** and **`ILookUpObjectList`**. As their names suggest, these two interfaces are essentially lists of **`IScheduleAppointments`** and **`ILookUpItems`** respectively.

The last interface, **`IScheduleDataProvider`**, is a wrapper that holds multiple `ILookUpObjectLists` and one `IScheduleAppointmentList`. It is through this interface that the `ScheduleControl` interacts with the source of data, and in fact, `ScheduleControl.DataSource` is an `IScheduleDataProvider` object.

The `IScheduleDataProvider` object exposes methods of interacting with the data like retrieving lookup lists and providing appointments for specified time periods. The Essential Schedule source code file **`ScheduleAppointment.cs`** provides a base class implementation of these interfaces, exposing a partially abstract set of classes (the `ScheduleData` classes) that you can use to indirectly implement these interfaces.

The **`SimpleScheduleDataProvider`** classes that were used in the Tutorial are derived from these base classes.

The following sections discuss these required data interfaces in more detail.

3.4.2.1 The Appointments Data

Three of the five data interfaces work directly with the Appointment data. Here are the three interfaces.

- **`IScheduleAppointment` Interface**
`IScheduleAppointment` defines the objects that represent appointments in the `ScheduleControl`.

[C#]

```
public interface IScheduleAppointment : IComparable, ICloneable
{
    /// A unique identifier for this schedule item.
    int UniqueID {get; set;}

    /// Identifies the owner of this schedule item.
    int Owner {get; set;}

    /// The start time of this item.
    DateTime StartTime {get; set;}

    /// The end time for this item.
    DateTime EndTime {get; set;}

    /// The subject or topic title for this schedule item.
    string Subject {get; set;}

    /// The text displayed as the main content of this schedule item.
    string Content {get; set;}

    /// Whether or not this item is to appear as an all day item.
    bool AllDay {get; set;}

    /// A categorizer value for this item.
    int LabelValue {get; set;}

    /// A categorizer value for this item.
    int MarkerValue {get; set;}

    /// Whether or not some reminder action should be taken as this item
    becomes current.
    bool Reminder {get; set;}

    /// Indicates when the reminder action should be taken.
    int ReminderValue {get; set;}

    /// Some item dependent string-like the location of a meeting.
    string LocationValue {get; set;}

    /// Whether or not this item has been modified.
    bool Dirty {get; set;}

    /// Whether or not changes to this item should be ignored.
    bool IgnoreChanges {get; set;}
```

```
/// An arbitrary object associated with this item.  
object Tag {get; set;}  
  
/// A version associated with the data schema for this item.  
int Version {get;}  
}
```

- **IScheduleAppointmentList Interface**

IScheduleAppointmentList represents a collection of **IScheduleAppointments** that serve as the data for the **ScheduleControl**.

```
[C#]  
  
/// A collection of ISchedule objects.  
public interface IScheduleAppointmentList  
{  
    /// An IScheduleAppointment referenced through an indexer.  
    IScheduleAppointment this[int i]{get; set;}  
  
    /// The number of IScheduleAppointments in this list.  
    int Count {get;}  
  
    /// Arranges the IScheduleAppointments in this list according to  
    IScheduleAppointment.StartTime.  
    void SortStartTime();  
  
    /// Adds an IScheduleAppointment at the end of this collection.  
    void Add(IScheduleAppointment item);  
  
    /// Inserts the given IScheduleAppointment at a particular position  
    in this collection.  
    void Insert(int index, IScheduleAppointment item);  
  
    /// Removes a given IScheduleAppointment from this collection.  
    void Remove(IScheduleAppointment item);  
  
    /// Removes an IScheduleAppointment at the given index from this  
    collection.  
    void RemoveAt(int index);  
  
    /// Locates the index of a particular IScheduleAppointment in this  
    collection.  
    int IndexOf(IScheduleAppointment item);  
}
```

```
/// Finds an IScheduleAppointment in the collection using its
IScheduleAppointment.UniqueID.
IScheduleAppointment Find(object uniqueID);

/// Returns an instance of a new schedule item.
IScheduleAppointment NewScheduleAppointment();

/// Returns an IEnumerator that iterates through this list.
IEnumerator GetEnumerator();
}
```

- **IScheduleDataProvider Interface**

IScheduleDataProvider has two functional roles. One is to manage the list of Appointment data needed by the ScheduleControl. The second role is to provide the DropList data.

```
[C#]

public interface IScheduleDataProvider
{
    /// Provides a list of schedule items for a particular day.
    IScheduleAppointmentList GetScheduleForDay(DateTime day);

    /// Provides a list of schedule items for a range of dates.
    IScheduleAppointmentList GetSchedule(DateTime startDate, DateTime
endDate);

    /// Provides a list of schedule items for a particular day and
owner.
    IScheduleAppointmentList GetScheduleForDay(DateTime day, int owner);

    /// Provides a list of schedule items for a range of dates and a
specified owner.
    IScheduleAppointmentList GetSchedule(DateTime startDate, DateTime
endDate, int owner);

    /// Called when the ScheduleControl needs to save modifications to
the schedule items back to the data store.
    void CommitChanges();

    /// Determines whether CommitChanges is called when the top-level
Form holding the ScheduleControl is closed.
    SaveOnCloseBehavior SaveOnCloseBehaviorAction

    /// Gets or sets whether the data store is modified.
    bool IsDirty
}
```

```
/// Returns an instance of a new schedule item.
IScheduleAppointment NewScheduleAppointment();

/// Adds a schedule item to this list.
void AddItem(IScheduleAppointment item);

/// Removes a schedule item from this list.
void RemoveItem(IScheduleAppointment item);

/// Initializes the contents of the ILookupObjectList lists.
void InitLists();

/// Returns a list holding the possible values for the <see
cref="IScheduleAppointment.LocationValue"/> property.
ILookUpObjectList GetLocations();

/// Returns a list holding the possible values for the <see
cref="IScheduleAppointment.MarkerValue"/> property.
ILookUpObjectList GetMarkers();

/// Returns a list holding the possible values for the <see
cref="IScheduleAppointment.LabelValue"/> property.
ILookUpObjectList GetLabels();

/// Returns a list holding the possible values for the <see
cref="IScheduleAppointment.ReminderValue"/> property.
/// </summary>
ILookUpObjectList GetReminders();

/// Returns a list holding the possible values for the <see
cref="IScheduleAppointment.Owner"/> property.
/// </summary>
ILookUpObjectList GetOwners();
}
```

3.4.2.2 The DropLists

Two of the five data interfaces work directly with the **DropList** data. Here are the two interfaces.

- **ILookUpObject Interface**

ILookUpObject is part of the data support to provide the DropList data. This interface defines the object that may appear in a droplist.

[C#]

```
/// Defines items that can be included in a ILookupObjectList.
/// Choice lists within the ScheduleControl are used to provide
possible
/// schedule item information like location or a reminder.
ILookUpObject
/// allows such list items to have a ValueMember / DisplayMember
associated with
/// the choices as well as a color that will be used in drop-downs
showing these
/// lists. Value members are normally the values serialized to data
stores.
public interface ILookupObject
{
    /// The value member associated with this item.
    int ValueMember {get; set;}

    /// The display member associated with this item.
    string DisplayMember {get; set;}

    /// A color associated with this item.
    Color ColorMember {get; set;}
}
```

- **ILookUpObjectList Interface**

ILookUpObjectList is the wrapper for this list of objects that may appear in a droplist.

[C#]

```
/// Collection of <see cref="ILookUpObject"/> items.
public interface ILookupObjectList
{
    /// Indexer that returns a ILookupObject object.
    ILookupObject this[int i] {get; set;}
}
```

4 Concepts and Features

The features of the Schedule control are illustrated with use case scenarios, code examples and screen shots under this section.

4.1 ScheduleControl

It is a User Control that provides the basic scheduling functionality.

Properties

Name	Description
Appearance	Gets / sets the ScheduleAppearance object that controls the visual aspects of the ScheduleControl.
Calendar	Gets the navigation calendar.
CaptionPanel	Gets the caption panel that holds the caption above the calendar.
DataSource	Gets / sets the data source for the ScheduleControl.
EnableAlerts	Indicates whether alerts should be raised as the appointment time approaches.
NavigationPanel	Gets the navigation panel.
ScheduleType	Gets / sets whether a daily, weekly or monthly schedule is displayed.

Methods

Name	Description
AddControlToNavigationPanel	Adds the specified control to the navigation panel underneath the navigation calendar.
AddSpanAppointment	Adds a multiday span appointment to a data

	provider.
PerformNewItemClick	Displays a dialog box allowing you to add an item.
PerformDeleteItemClick	Displays a dialog box allowing you to delete an item.
PerformEditItemClick	Displays a dialog box allowing you to edit an item.
PerformSwitchToScheduleViewTypeClick	Switches the display to the specified ScheduleView type.

Events

Name	Description
ItemChanged	Notifies when an appointment is modified.
ScheduleAppointmentClick	Occurs when an item is clicked / double-clicked.
ScheduleGridCreated	Lets you either use a derived ScheduleGridControl or subscribe to the events on the ScheduleGridControl.

4.1.1 Caption Panel

This panel displays a caption at the top of the ScheduleControl. The two buttons on this panel will navigate the schedule forward and backward.

4.1.2 Navigation Panel

It is a panel where you can place additional controls and make them appear adjacent to the ScheduleControl.

4.1.3 Navigation Calendar

A GridControl-derived object that displays multiple calendars lets you select particular dates or data ranges to be displayed in the ScheduleControl.

Properties

Name	Description
------	-------------

CalendarGrid	Gets the grid control that is used to display the calendars.
DateValue	Gets / sets the date value for the navigation calendar.
SelectedDates	Gets the dates selected in the navigation calendar.
ShowWeekNumbers	Indicates whether the week numbers should be displayed in the calendars.
Today	Gets / sets the DateTime value for the current day.

Methods

Name	Description
FirstDayOfMonth	Returns the date of the first day of the month of the passed-in date.
MondayBeforeDate	Returns the Monday before the given date.
SundayAfterDate	Returns the Sunday after the given date.

Event

Name	Description
DateValueChanged	Occurs when NavigationCalendar.DateValue property is changed.

4.2 Customizing Appearance

The appearance of any region of the ScheduleControl can be customized by using the **ScheduleControl.Appearance** property.

This property gains access to the **ScheduleAppearance** object which controls the various appearance attributes of different scheduler regions.

The following table describes the appearance options available in the ScheduleControl.

Name	Description
------	-------------

Border	
ClickItemBorderColor	Gets or sets the border color of a clicked item.
DragColor	Gets or sets the color of the item that is dragged.
SolidBorderColor	Gets or sets the color of the solid lines in the calendar.
Caption	
CaptionBackColor	Gets or sets the color of the caption area above the calendar.
ShowCaption	Specifies whether the caption panel above the calendar is visible.
ShowCaptionButtons	Specifies whether the navigation buttons are shown on the caption panel.
DisplayItemFormat	
AllDayItemFormat	Gets or sets the display format of an allday item.
DateFormat	Gets or sets the format string used when formatting any token from DisplayItemFormatStrings that represents a date only value.
DateTimeFormat	Gets or sets the format string used when formatting any token from DisplayItemFormatStrings that represents the combined date and time values.
DayItemFormat	Gets or sets the display format of a schedule item displayed in a day or workweek view.
FullWeekHeaderFormat	Specifies the display format of the header of a day in a week view.
LongHeaderFormat	Specifies the display format of the header in a day view.
SpanItemFormatLeftText	Specifies the display format of text displayed on the interior left side of a multiday span.

SpanItemFormatMiddleText	Specifies the display format of text displayed in the middle of a multiday span.
SpanItemFormatRightText	Specifies the display format of text displayed on the interior right side of a multiday span.
SpanItemFormatTerminalLeftText	Specifies the display format of text displayed on the open left side of a multiday span.
SpanItemFormatTerminalRightText	Specifies the display format of text displayed on the open right side of a multiday span.
TimeFormat	Gets or sets the format string used when formatting any token from DisplayItemFormatStrings that represents a time only value.
WeekHeaderFormat	Specifies the display format of the header label in a workweek view.
WeekMonthItemFormat	Specifies the display format of a schedule item shown in a week or month view.
WorkWeekHeaderFormat	Determines the display format of the header of a day in a workweek view.
Header	
AllDayBackColor	Gets or sets the back color of the allday row in the calendar.
MonthWeekHeaderBackColor	Specifies the back color of the header cells in a month or week view.
MonthWeekHeaderForeColor	Specifies the fore color of the header cells in a month or week view.
WorkWeekHeaderBackColor	Specifies the back color of header cells in a workweek view.
WorkWeekHeaderForeColor	Specifies the fore color of header cells in a workweek view.
Navigation Calendar	

NavigationCalendarArrowColor	Specifies the color of arrows in the navigation calendar.
NavigationCalendarBackColor	Specifies the back color of the navigation calendar.
NavigationCalendarDisabledTextColor	Specifies the color of disabled text in the navigation calendar.
NavigationCalendarHeaderColor	Gets or sets the color of the header in the navigation calendar.
NavigationCalendarSelectionColor	Gets or sets the selection color in the navigation calendar.
NavigationCalendarStartDayOfWeek	Specifies the DayOfWeek that is shown in the left-most column of the navigation calendar.
NavigationCalendarTextColor	Specifies the text color of the navigation calendar.
NavigationCalendarTodayColor	Specifies the color of today's text in the navigation calendar.
NavigationCalendarWeekNumberColor	Specifies the color of week numbers in the navigation calendar.
Prime Time	
NonPrimeTimeCellColor	Gets or sets the color of non-prime time cells in the calendar.
PrimeTimeCellColor	Gets or sets the color of prime time cells in the calendar.
PrimeTimeEnd	Specifies the time when the prime time color stops being used in the display.
PrimeTimeStart	Specifies the time when the prime time color starts being used in the display.
Time Column	
Hours24	Determines whether the time column is displayed using a 24-hour format.
MarkColumnColor	Gets or sets the color of the thick solid line next to

	the time column in a day view.
ShowTime	Indicates whether the time column should appear.
TimeBackColor	Specifies the back color of the time column.
TimeBigFontSize	Determines the size of larger font used in the time column.
TimeLittleFontSize	Determines the size of smaller font used in the time column.
TimeTextColor	Specifies the color of text in the time column.
Visual Style	
VisualStyle	Specifies the visual style for the ScheduleControl.
Miscellaneous	
DayMonthCutOff	Gets or sets the maximum number of days that can appear side-by-side in a day style calendar.
DivisionsPerHour	Gets or sets the number of time divisions that appear in a day, custom or workweek view.
MonthCalendarStartDayOfWeek	Gets or sets the DayOfWeek that is shown in the left-most column of the month calendar.
MonthShowFullWeek	Specifies whether the month view shows 7 columns or 6 columns with saturday or sunday stacked.
ScheduleAppointmentTipFormat	Defines the text that is displayed for schedule item tips.
ScheduleAppointmentTipsEnabled	Determines whether to show item tips.
SplitterBackColor	Specifies the back color of the two splitters in the ScheduleControl.
TextColor	Specifies the color of basic text shown in the calendar.
ThemesEnabled	Specifies whether the themes are enabled.

WeekCalendarStartDayOfWeek	Specifies the DayOfWeek that is shown in the first column of the week calendar.
----------------------------	---

4.3 Metro Theme for Essential Schedule

This feature enables you to apply new Metro styles to the Schedule control.

Use Case Scenario

The Metro theme support is useful for commercial applications in order to attract end users with inspiring UI look and feel.

Properties

Property	Description
VisualStyle	This is an enumeration type property. It is used to get or set the visual styles (skins) such as Office2010, Office2007, Office2003, Metro, etc.

Events

Event	Parameters	Description
ThemeChanged	Object sender, EventArgs	Occurs when the ThemesEnabled property is changed.

4.3.1 Applying Metro Theme to the Schedule Control

You can apply the Metro theme to the Schedule control by setting the **GridVisualStyles** property as **Metro**. The following code example illustrates this.

[C#]

```
this.scheduleControl1.GetScheduleHost().Schedule.Appearance.VisualStyle = Syncfusion.Windows.Forms.GridVisualStyles.Metro;
```

[VB]

```
Me.scheduleControl1.GetScheduleHost().Schedule.Appearance.VisualStyle = Syncfusion.Windows.Forms.GridVisualStyles.Metro
```

The following screenshot is a sample output for the previous code.

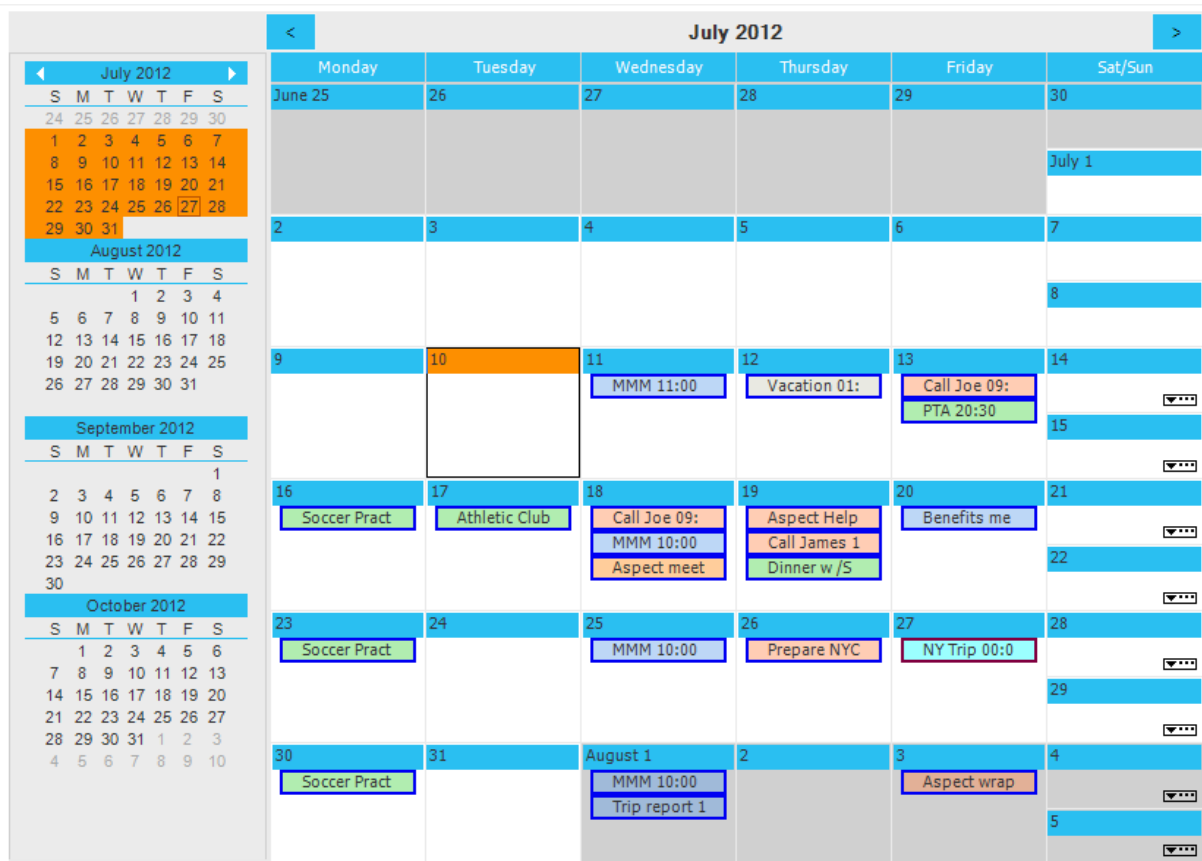


Figure 28: Schedule Control Applied with Metro Theme

4.4 Item Dragging Context in the ItemChanging event

This feature provides support to detect the dragging context when an item is dropped in schedule part or calendar part. It also enables to cancel the needed items through ItemChanging event.

Use Case Scenario

In ItemChanging event, through the ItemDragHitContext enumeration, you can detect the dragging context (Schedule or Calendar) and cancel the needed items.

Properties

Property	Description	Data Type
ItemDragHitContext	Specifies where the mouse is during an appointment drag in a	enum

	Week or Month view.	
--	---------------------	--

Events

Event	Parameters	Description
ItemChanging	object sender, ScheduleAppointmentCancelEventArgs e	Occurs after an IScheduleAppointment is modified

Sample Link

You can get the schedule sample from the following online location,

<http://asp.syncfusion.com/sfwinrepsamplebrowser/8.4.0.10/Windows/Schedule.Windows/Samples/2.0/Schedule%20Samples/Scheduler%20With%20Recurrence%20Demo/Sample.aspx?args=0>

Adding this support to an Application

The following steps help you to get the target part in the Schedule control while dragging:

1. Create a schedule control enabled sample application
2. Add appointments in that schedule grid
3. Hook the '**ItemChanging**' event

[C#]

```
this.scheduleControl1.ItemChanging += new  
ScheduleAppointmentChangingEventHandler(scheduleControl1.ItemChanging);
```

[VB]

```
AddHandler scheduleControl1.ItemChanging, AddressOf  
scheduleControl1.ItemChanging
```

Get the drag hit context with the below code.

[C#]

```
void scheduleControl1_ItemChanging(object sender,  
ScheduleAppointmentCancelEventArgs e)
```



```
{  
    if (e.Action == ItemAction.ItemDrag)  
    {  
        Console.WriteLine("Dropped Area :" + e.ItemDragHitContext);  
    }  
}
```

[VB]

```
Private Sub scheduleControl1 ItemChanging(ByVal sender As Object, ByVal  
e As ScheduleAppointmentCancelEventArgs)  
    If e.Action = ItemAction.ItemDrag Then  
        Console.WriteLine("Dropped Area :" +  
e.ItemDragHitContext.ToString())  
    End If  
End Sub
```

You can cancel the dropped item using the ItemDragHitContext property as shown below in the following code.

[C#]

```
void scheduleControl1 ItemChanging(object sender,  
ScheduleAppointmentCancelEventArgs e)  
{  
    if (e.ItemDragHitContext == ItemDragHitContext.Calendar)  
        e.Cancel = true;  
}
```

[VB]

```
Private Sub scheduleControl1 ItemChanging(ByVal sender As Object, ByVal  
e As ScheduleAppointmentCancelEventArgs)  
    If e.ItemDragHitContext = ItemDragHitContext.Calendar Then  
        e.Cancel = True  
    End If  
End Sub
```

4.5 Time Interval

This topic illustrates the time interval format options for scheduling appointments.

4.5.1 Setting the Time Interval in Seconds Format

The Schedule control, by default, allows you to set the time interval for scheduling appointments only in hours and minutes format. Now, you can also include seconds in the time interval by enabling the **AllowSecondsInAppointment** property.

[C#]

```
this.scheduleControl1.AllowSecondsInAppointment = true;
```

[VB]

```
Me.scheduleControl1.AllowSecondsInAppointment = True
```

The screenshot displays the Windows Forms Schedule control. At the top, there are two rows for 'Start Time' and 'End Time'. Each row consists of a date dropdown menu (showing '7/ 2/2013') and a time input field (showing '8:00:01 AM' and '8:00:02 AM' respectively). Below these fields is a calendar grid. The grid has two columns. The left column shows dates 13, 20, and 21. The right column shows dates 14 and 21. Appointments are shown as colored rectangles with their start and end times. For example, on 08/13/2013, there are appointments from 01:00:00 to 07:00:00. On 08/21/2013, there is an appointment from 08:00:01 to 08:00:02. A small dropdown arrow is visible in the center of the grid.

5 Frequently Asked Questions

This section comprises the following topics:

5.1 How to disable the drag behavior of schedule appointments in the ScheduleControl

You can do this by invoking the **ItemChanging** event of the ScheduleControl, and canceling the **ItemDrag** action, as shown in the below code snippet.

[C#]

```
// Handle the ItemChanging event.
this.scheduleControll1.ItemChanging += new
ScheduleAppointmentChangingEventHandler(scheduleControll1_ItemChanging);

private void scheduleControll1_ItemChanging(object sender,
Syncfusion.Schedule.ScheduleAppointmentCancelEventArgs e)
{
    // Cancel the ItemDrag action.
    if (e.Action == ItemAction.ItemDrag)
    {
        e.Cancel = true;
    }
}
```

[VB.NET]

```
' Handle the ItemChanging event.
AddHandler Me.scheduleControll1.ItemChanging, AddressOf
scheduleControll1_ItemChanging

Private Sub scheduleControll1_ItemChanging(ByVal sender As Object, ByVal
e As Syncfusion.Schedule.ScheduleAppointmentCancelEventArgs)
    ' Cancel the ItemDrag action.
    If e.Action = ItemAction.ItemDrag Then
        e.Cancel = True
    End If
End Sub
```

5.2 How to set the schedule appointments as read-only

You can achieve this by canceling the **ScheduleAppointmentClick** event of the **ScheduleControl**. Please refer the below code snippet which illustrates this.

[C#]

```
// Handle the ScheduleAppointmentClick event.
this.scheduleControl1.ScheduleAppointmentClick += new
ScheduleAppointmentClickEventHandler(scheduleControl1_ScheduleAppointmentClick);

private void scheduleControl1_ScheduleAppointmentClick(object sender,
ScheduleAppointmentClickEventArgs e)
{
    // Cancel the event.
    e.Cancel = true;
}
```

[VB.NET]

```
' Handle the ScheduleAppointmentClick event.
AddHandler ScheduleControl1.ScheduleAppointmentClick, AddressOf
ScheduleControl1_ScheduleAppointmentClick

Private Sub scheduleControl1_ScheduleAppointmentClick(ByVal sender As
Object, ByVal e As ScheduleAppointmentClickEventArgs)
    ' Cancel the event.
    e.Cancel = True
End Sub
```

5.3 How to change the current display to a desired Schedule View Type

The current display of a **ScheduleControl** can be changed to a desired view type by invoking the **PerformSwitchToScheduleViewTypeClick** method and passing the desired view as parameter.

[C#]

```
// Switches the display to Month View.
```

```
scheduleControl1.PerformSwitchToScheduleViewTypeClick(ScheduleViewType.  
Month);
```

[VB.NET]

```
' Switches the display to Month View.  
scheduleControl1.PerformSwitchToScheduleViewTypeClick(ScheduleViewType.  
Month)
```

5.4 How to obtain the date that is clicked in the ScheduleControl

Clicked DateTime value of a Schedule Control can be obtained by handling the **ScheduleAppointmentClick** event as shown in the following code snippet.

[C#]

```
// Subscribe to item click event.  
this.scheduleControl1.ScheduleAppointmentClick += new  
ScheduleAppointmentClickEventHandler(scheduleControl1_ScheduleAppointme  
ntClick);  
  
// Sample event handler to catch clicks on the schedule control.  
private void scheduleControl1_ScheduleAppointmentClick(object sender,  
ScheduleAppointmentClickEventArgs e)  
{  
    Console.WriteLine("scheduleControl1_ScheduleAppointmentClick: {0}  
{1}", e.ClickType, e.ClickDateTime);  
}
```

[VB.NET]

```
' Subscribe to item click event  
AddHandler scheduleControl1.ScheduleAppointmentClick, AddressOf  
scheduleControl1_ScheduleAppointmentClick  
  
' Sample event handler to catch clicks on the schedule control.  
Private Sub scheduleControl1_ScheduleAppointmentClick(ByVal sender As  
Object, ByVal e As ScheduleAppointmentClickEventArgs)  
    Console.WriteLine("scheduleControl1_ScheduleAppointmentClick: {0}  
{1}", e.ClickType, e.ClickDateTime)  
End Sub
```

5.5 How to prevent the switching of schedule view type

The schedule type can be changed at any time by using the **ScheduleViewType** property. Switching of view style can be prevented by making the view type as constant in the Schedule control by disabling the SwitchViewStyle property. By default this property value is true.

Properties

Property	Description	Data Type
SwitchViewStyle	Gets/sets whether the view style should be changed in the Schedule control. The default value is <i>true</i> .	bool

The following codes are used to disable the SwitchViewStyle property.

```
[C#]
this.ScheduleControl1.ScheduleType = ScheduleViewType.Month;
this.ScheduleControl1.SwitchViewStyle = false;
```

```
[VB]
Me.ScheduleControl1.ScheduleType = ScheduleViewType.Month;
Me.ScheduleControl1.SwitchViewStyle = False;
```

5.6 How to show the start and end time with scheduled appointments

To show or hide the time of an appointment in the schedule window, the ShowTime property can be set to true or false. The time shown in the appointment can also be formatted by setting the time format through the WeekMonthItemFormat property.

Properties

Table 1: Properties Table

Property	Description	Type	Data Type	Reference links
ShowTime	Indicates whether the time column should appear or	ShowTime	Boolean	

	not.			
WeekMonthItemFormat	Specifies the display format of a schedule item shown in a week or month view.	WeekMonthItemFormat	string	
ScheduleAppointmentTipFormat	Defines the text that is displayed for schedule item tips.	ScheduleAppointmentTipFormat	string	
ScheduleAppointmentTipsEnabled	Determines whether to show item tips.	ScheduleAppointmentTipsEnabled	Boolean	

TimeFormat	t			
WeekHeaderFormat	MMMM dd			
WeekMonthFullFormat	dddd, dd MMMM yyyy			
WeekMonthItemFormat	[subject] [starttime][endtime]			
WeekMonthNewMonth	MMMM d			

[C#]

```
// Show the appointment time.
this.scheduleControl1.Appearance.ShowTime = true;

// To show the appointment time based on this format.
this.ScheduleControl1.Appearance.WeekMonthItemFormat="[subject]
[starttime][endtime]" ;

// Tooltip format.
this.ScheduleControl1.Appearance.ScheduleAppointmentTipFormat =
"[subject] [starttime][endtime]";

// Enable the appointment tooltip.
this.ScheduleControl1.Appearance.ScheduleAppointmentTipsEnabled = true;
```

```
[VB.NET]

' Show the appointment time.
Me.ScheduleControl1.Appearance.ShowTime=True

// To show the appointment time based on this format.
Me.ScheduleControl1.Appearance.WeekMonthItemFormat="[subject]
[starttime][endtime]"

// Tooltip format.
Me.ScheduleControl1.Appearance.ScheduleAppointmentTipFormat = "[subject]
[starttime][endtime]"

// Enable the appointment tooltip.
Me.ScheduleControl1.Appearance.ScheduleAppointmentTipsEnabled = True
```

5.7 How to set the Transparency Level of a Span Appointment

The **EnableTransparencySpan** property is used to enable transparency in Span appointments. Once you have enabled this property, you can set the transparency level (from 0 through 255) by using the **SpanTransparencyLevel** property.

```
[C#]

this.scheduleControl.Appearance.EnableTransparentSpan = true;
this.scheduleControl.Appearance.SpanTransparencyLevel = 220;
```

```
[VB]

Me.scheduleControl.Appearance.EnableTransparentSpan = True
Me.scheduleControl.Appearance.SpanTransparencyLevel = 220
```

2	3	4	5	6	7	8
---	---	---	---	---	---	---

Index

1

1 Overview 4

1.1 Introduction to Essential Schedule 4

1.2 Prerequisites and Compatibility 7

1.3 Documentation 8

2

2 Installation and Deployment 10

2.1 Installation 10

2.2 Sample and Location 13

2.3 Deployment Requirements 16

3

3 Getting Started 17

3.1 Control Structure 17

3.2 Class Diagram 23

3.3 Tutorial 24

3.3.1 Lesson 1: Using ScheduleControl and SimpleScheduleDataProvider 24

3.3.1.1 Lesson: Using ScheduleControl 25

3.4 Data used by ScheduleControl 39

3.4.1 ScheduleData Base Classes 41

3.4.1.1 The Appointments Data 41

3.4.1.2 The DropLists 47

3.4.2 IScheduleData Interfaces 48

3.4.2.1 The Appointments Data 49

3.4.2.2 The DropLists 53

4

4 Concepts and Features 55

4.1 ScheduleControl 55

4.1.1 Caption Panel 56

4.1.2 Navigation Panel 56

4.1.3 Navigation Calendar 56

4.2 Customizing Appearance 57

4.3 Metro Theme for Essential Schedule 62

4.3.1 Applying Metro Theme to the Schedule Control 62

4.4 Item Dragging Context in the ItemChanging event 63

4.5 Time Interval 66

4.5.1 Setting the Time Interval in Seconds Format 66

5

5 Frequently Asked Questions 67

5.1 How to disable the drag behavior of schedule appointments in the ScheduleControl 67

5.2 How to set the schedule appointments as read-only 68

5.3 How to change the current display to a desired Schedule View Type 68

5.4 How to obtain the date that is clicked in the ScheduleControl 69

5.5 How to prevent the switching of schedule view type 70

5.6 How to show the start and end time with scheduled appointments 70

5.7 How to set the Transparency Level of a Span Appointment 72