Chapter 25: Web Services in Visual Basic

Internet & World Wide Web
How to Program, 5/e

Note: This chapter is a copy of Chapter 23 of our book *Visual Basic 2010 How to Program*. For that reason, we simply copied the PowerPoint slides for this chapter and *did not* re-numb er them

OBJECTIVES

In this chapter you'll learn:

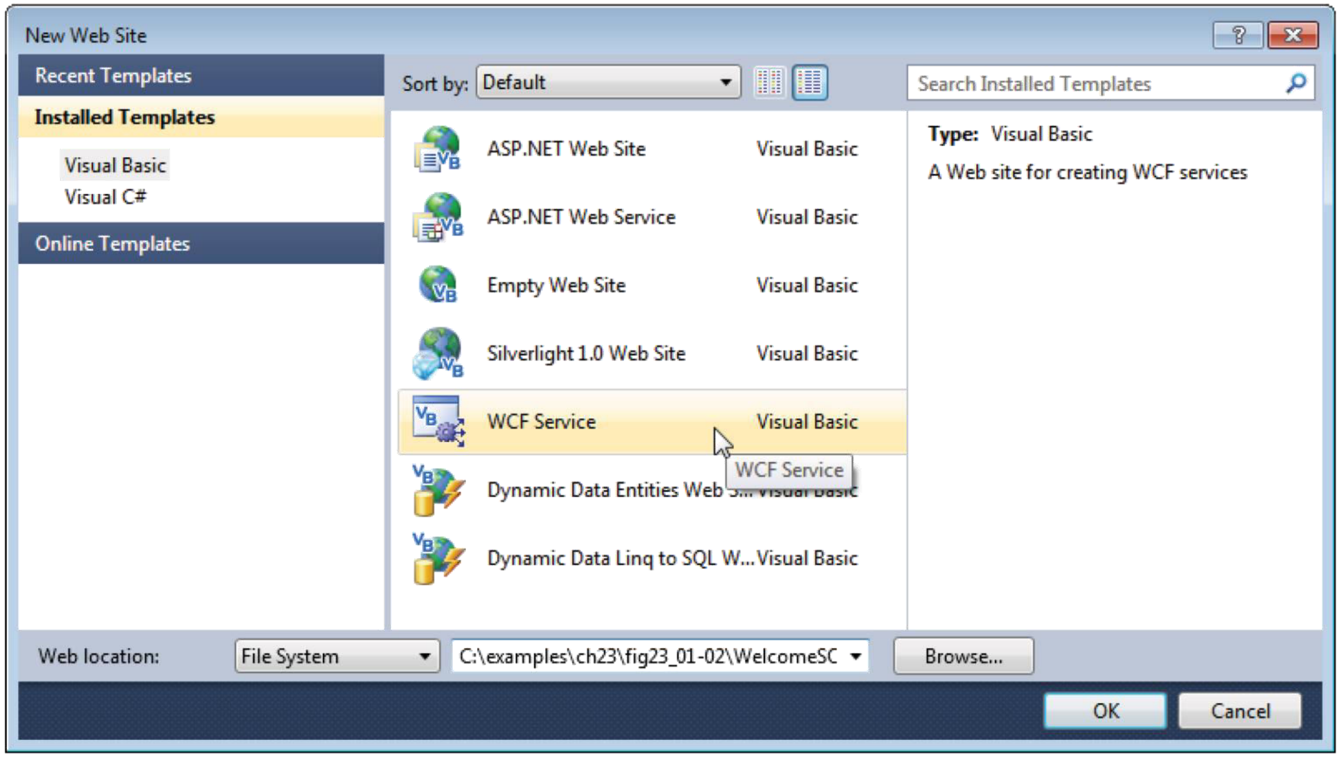
- How to create WCF web services.
- How XML, JSON, XML-Based Simple Object Access Protocol (SOAP) and Representational State Transfer Architecture (REST) enable WCF web services.
- The elements that comprise WCF web services, such as service references, service endpoints, service contracts and service bindings.
- How to create a client that consumes a WCF web service.
- How to use WCF web services with Windows and web applications.
- How to use session tracking in WCF web services to maintain state information for the client.
- How to pass user-defined types to a WCF web service.

```
' Fig. 23.1: IWelcomeSOAPXMLService.vb
' WCF web service interface that returns a welcome message through SOAP
' protocol and XML format.
<ServiceContract()>
Public Interface IWelcomeSOAPXMLService
   ' returns a welcome message
   <OperationContract()>
   Function Welcome(ByVal yourName As String) As String
End Interface ' IWelcomeSOAPXMLService
```

Fig. 23.1 | WCF web service interface that returns a welcome message through SOAP protocol and XML format.

```vb
' Fig. 23.2: WelcomeSOAPXMLService.vb
' WCF web service that returns a welcome message through SOAP protocol and
' XML format.
Public Class WelcomeSOAPXMLService
   Implements IWelcomeSOAPXMLService

   ' returns a welcome message
   Public Function Welcome(ByVal yourName As String) As String _
      Implements IWelcomeSOAPXMLService.Welcome

      Return "Welcome to WCF Web Services with SOAP and XML, " &
         yourName & "!"
   End Function ' Welcome
End Class ' WelcomeSOAPXMLService
```

Fig. 23.2 | WCF web service that returns a welcome message through the SOAP protocol and XML format.



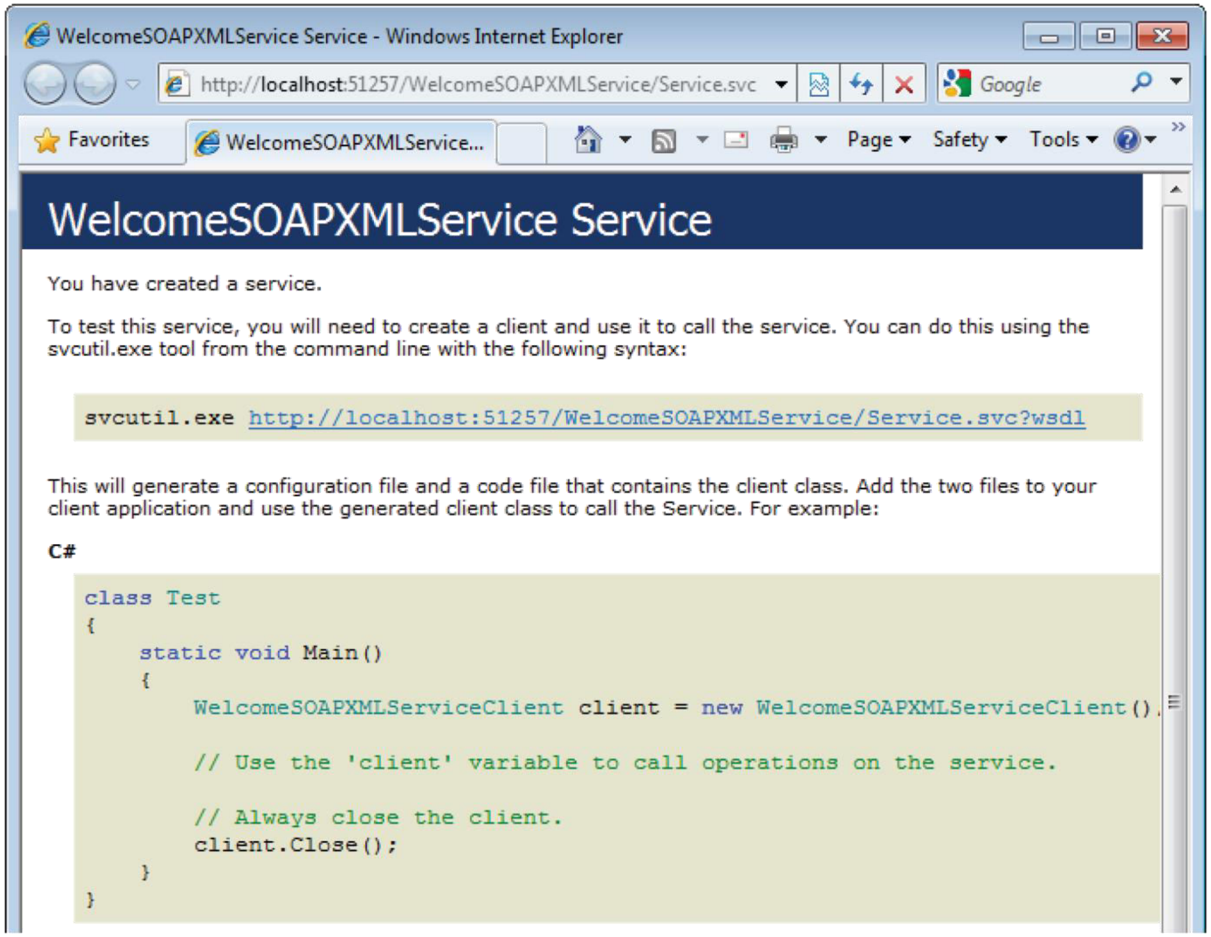
ASP.NET Web Site	Visual Basic
ASP.NET Web Service	Visual Basic
Empty Web Site	Visual Basic
Silverlight 1.0 Web Site	Visual Basic
WCF Service	Visual Basic
Dynamic Data Entities Web S...	Visual Basic
Dynamic Data Linq to SQL W...	Visual Basic

Type: Visual Basic

A Web site for creating WCF services

Web location: File System C:\examples\ch23\fig23_01-02\WelcomeSC

Fig. 23.3 | Creating a WCF Service in Visual Web Developer.


WelcomeSOAPXMLService Service - Windows Internet Explorer
http://localhost:51257/WelcomeSOAPXMLService/Service.svc
Google
Favorites
WelcomeSOAPXMLService...
Page
Safety
Tools

WelcomeSOAPXMLService Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

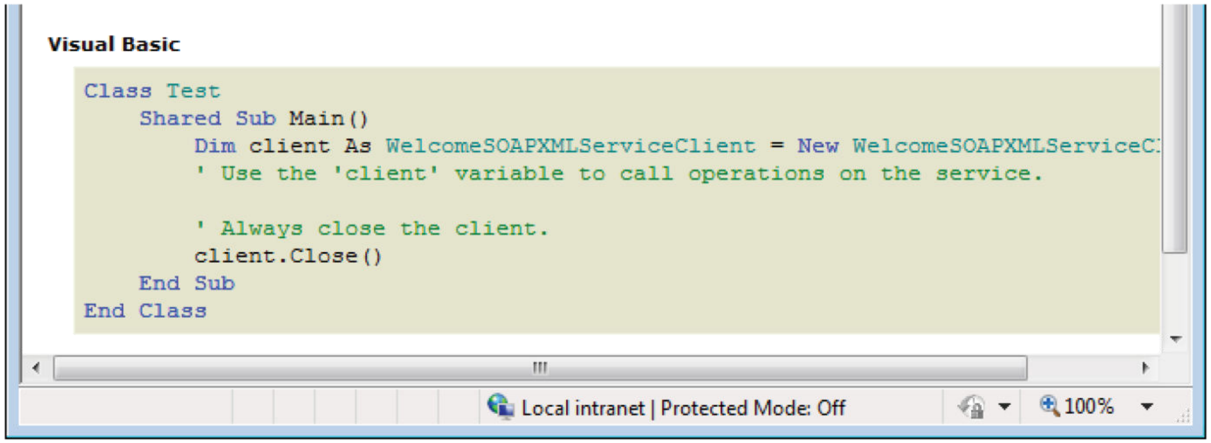
```
svcutil.exe http://localhost:51257/WelcomeSOAPXMLService/Service.svc?wsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

```
class Test
{
    static void Main()
    {
        WelcomeSOAPXMLServiceClient client = new WelcomeSOAPXMLServiceClient().

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

Fig. 23.4 | SVC file rendered in a web browser. (Part 1 of 2.)



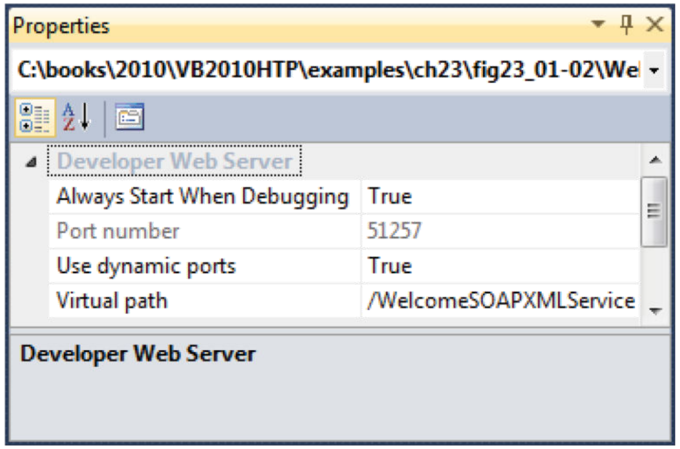
Visual Basic

```
Class Test
    Shared Sub Main()
        Dim client As WelcomeSOAPXMLServiceClient = New WelcomeSOAPXMLServiceC
        ' Use the 'client' variable to call operations on the service.

        ' Always close the client.
        client.Close()
    End Sub
End Class
```

Local intranet | Protected Mode: Off

100%

Fig. 23.4 | SVC file rendered in a web browser. (Part 2 of 2.)


Properties

C:\books\2010\VB2010HTP\examples\ch23\fig23_01-02\We

Developer Web Server	
Always Start When Debugging	True
Port number	51257
Use dynamic ports	True
Virtual path	/WelcomeSOAPXMLService

Developer Web Server

Fig. 23.5 | WCF web service **Properties** window.

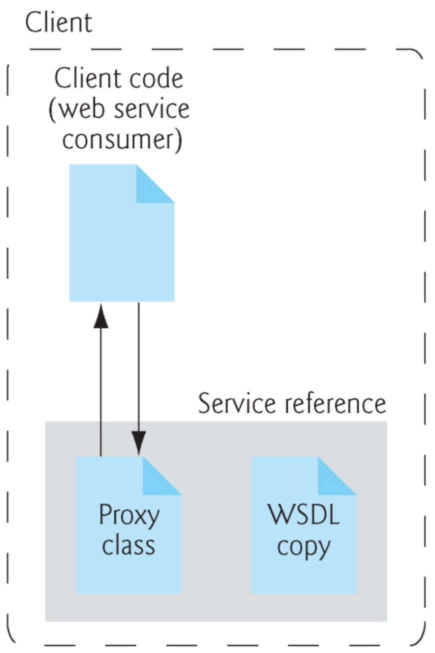


Fig. 23.6 | .NET WCF web service client after a web-service reference has been added.

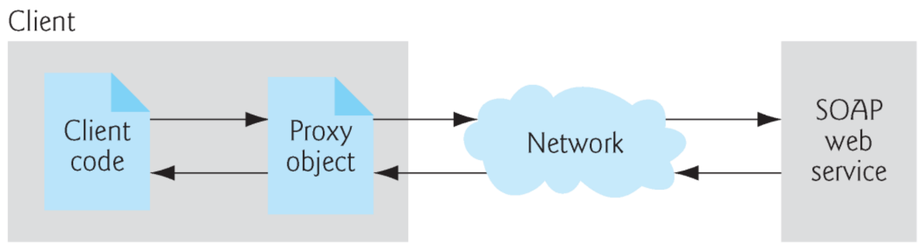
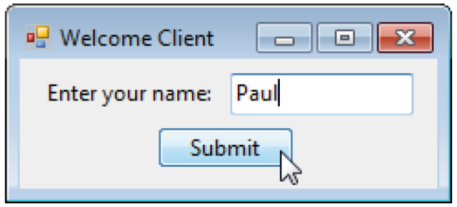


Fig. 23.7 | Interaction between a web-service client and a SOAP web service.

```
' Fig. 23.8: WelcomeSOAPXML.vb
' Client that consumes WelcomeSOAPXMLService.
Public Class WelcomeSOAPXML
   ' reference to web service
   Private client As New ServiceReference.WelcomeSOAPXMLServiceClient()

   ' creates welcome message from text input and web service
   Private Sub submitButton_Click(ByVal sender As System.Object,
      ByVal e As System.EventArgs) Handles submitButton.Click

      MessageBox.Show(client.Welcome(textBox.Text))
   End Sub ' submitButton_Click
End Class ' WelcomeSOAPXML
```

a) User inputs name



Enter your name: Paul

Fig. 23.8 | Client that consumes `WelcomeSOAPXMLService`. (Part 1 of 2.)

b) Message sent from **WelcomeSOAPXML-Service**

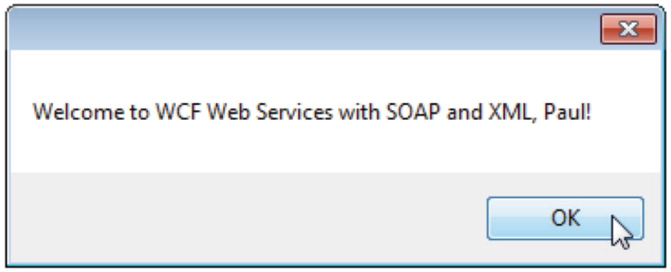

Welcome to WCF Web Services with SOAP and XML, Paul!

Fig. 23.8 | Client that consumes `WelcomeSOAPXMLService`. (Part 2 of 2.)

```
' Fig. 23.9: IWelcomeRESTXMLService.vb
' WCF web-service interface. A class that implements this interface
' returns a welcome message through REST architecture and XML data format.
Imports System.ServiceModel.Web

<ServiceContract()>
Public Interface IWelcomeRESTXMLService
   ' returns a welcome message
   <OperationContract()>
   <WebGet(UriTemplate:="welcome/{yourName}")>
   Function Welcome(ByVal yourName As String) As String
End Interface ' IWelcomeRESTXMLService
```

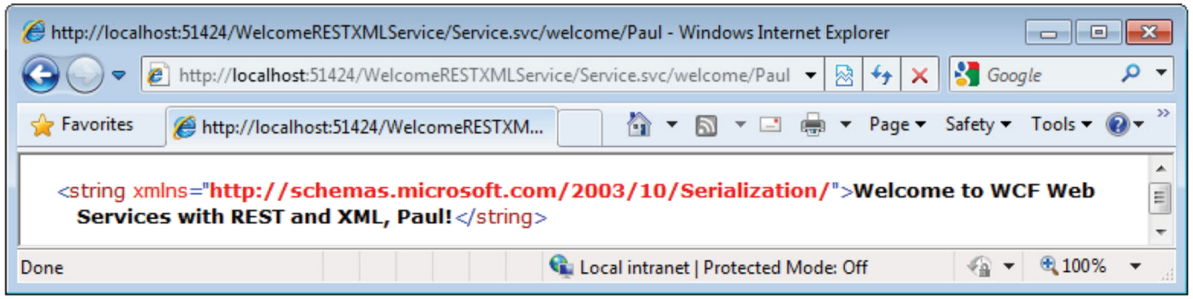
Fig. 23.9 | WCF web-service interface. A class that implements this interface returns a welcome message through REST architecture and XML data format.

```
' Fig. 23.10: WelcomeRESTXMLService.vb
' WCF web service that returns a welcome message using REST architecture
' and XML data format.
Public Class WelcomeRESTXMLService
   Implements IWelcomeRESTXMLService

   ' returns a welcome message
   Public Function Welcome(ByVal yourName As String) _
      As String Implements IWelcomeRESTXMLService.Welcome

      Return "Welcome to WCF Web Services with REST and XML, " &
         yourName & "!"
   End Function ' Welcome
End Class ' WelcomeRESTXMLService
```

Fig. 23.10 | WCF web service that returns a welcome message using REST architecture and XML data format.

```
<system.serviceModel>
   <behaviors>
      <serviceBehaviors>
         <behavior>
            <!-- To avoid disclosing metadata information, set the
               value below to false and remove the metadata
               endpoint above before deployment -->
            <serviceMetadata httpGetEnabled="true"/>
            <!-- To receive exception details in faults for debugging
               purposes, set the value below to true.  Set to false
               before deployment to avoid disclosing exception
               information -->
            <serviceDebug includeExceptionDetailInFaults="false"/>
         </behavior>
      </serviceBehaviors>
      <endpointBehaviors>
         <behavior>
            <webHttp/>
         </behavior>
      </endpointBehaviors>
   </behaviors>
   <protocolMapping>
      <add scheme="http" binding="webHttpBinding"/>
   </protocolMapping>
</system.serviceModel>
```

Fig. 23.11 | WelcomeRESTXMLService Web.config file.


http://localhost:51424/WelcomeRESTXMLService/Service.svc/welcome/Paul - Windows Internet Explorer
http://localhost:51424/WelcomeRESTXMLService/Service.svc/welcome/Paul
Google
Favorites
http://localhost:51424/WelcomeRESTXM...
Page
Safety
Tools
Done
Local intranet | Protected Mode: Off
100%

```
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">Welcome to WCF Web
  Services with REST and XML, Paul!</string>
```

Fig. 23.12 | Response from `WelcomeRESTXMLService` in XML data format.

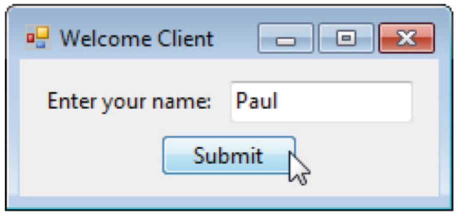
```
' Fig. 23.13: WelcomeRESTXML.vb
' Client that consumes the WelcomeRESTXMLService.
Imports System.Net
Imports System.Xml.Linq
Imports <xmlns="http://schemas.microsoft.com/2003/10/Serialization/">

Public Class WelcomeRESTXML
   ' object to invoke the WelcomeRESTXMLService
   Private WithEvents service As New WebClient()

   ' get user input and pass it to the web service
   Private Sub submitButton_Click(ByVal sender As System.Object,
      ByVal e As System.EventArgs) Handles submitButton.Click

      ' send request to WelcomeRESTXMLService
      service.DownloadStringAsync(New Uri(
         "http://localhost:51424/WelcomeRESTXMLService/Service.svc/" &
         "welcome/" & textBox.Text))
   End Sub ' submitButton_Click

```

Fig. 23.13 | Client that consumes the `WelcomeRESTXMLService`. (Part 1 of 3.)

```
    ' process web-service response
    Private Sub service_DownloadStringCompleted(ByVal sender As Object,
       ByVal e As System.Net.DownloadStringCompletedEventArgs) _
       Handles service.DownloadStringCompleted

       ' check if any errors occurred in retrieving service data
       If e.Error Is Nothing Then
          ' parse the returned XML string (e.Result)
          Dim xmlResponse = XDocument.Parse(e.Result)

          ' use XML axis property to access the <string> element's value
          MessageBox.Show(xmlResponse.<string>.Value)
       End If
    End Sub ' service_DownloadStringCompleted
 End Class ' WelcomeRESTXML
```

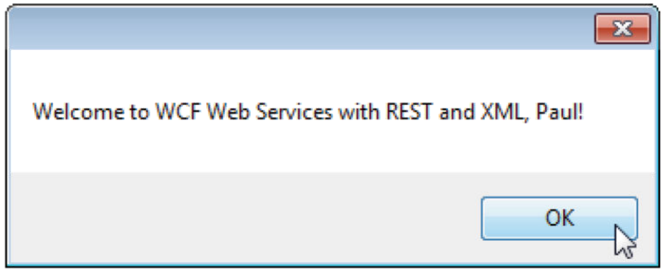
Fig. 23.13 | Client that consumes the `WelcomeRESTXMLService`. (Part 2 of 3.)

a) User inputs name.



Enter your name: Paul

b) Message sent from **WelcomeRESTXMLService**.



Welcome to WCF Web Services with REST and XML, Paul!

Fig. 23.13 | Client that consumes the `WelcomeRESTXMLService`. (Part 3 of 3.)

```
' Fig. 23.14: IWelcomeRESTJSONService.vb
' WCF web-service interface that returns a welcome message through REST
' architecture and JSON format.
Imports System.ServiceModel.Web

<ServiceContract()>
Public Interface IWelcomeRESTJSONService
```

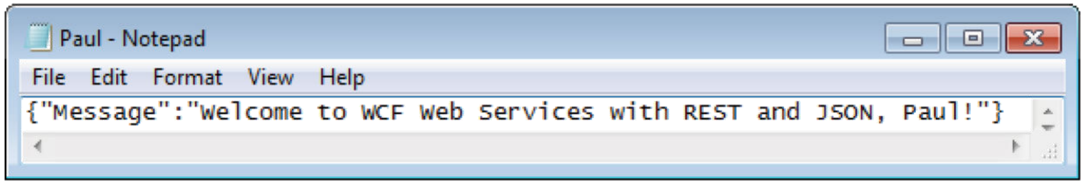
Fig. 23.14 | WCF web-service interface that returns a welcome message through REST architecture and JSON format. (Part 1 of 2.)

```
   ' returns a welcome message
   <OperationContract()>
   <WebGet(ResponseFormat:=WebMessageFormat.Json,
      UriTemplate:="welcome/{yourName}")>
   Function Welcome(ByVal yourName As String) As TextMessage
End Interface ' IWelcomeRESTJSONService

' class to encapsulate a String to send in JSON format
<DataContract()>
Public Class TextMessage
   Public messageValue As String

   ' property Message
   <DataMember()>
   Public Property Message() As String
      Get
         Return messageValue
      End Get
      Set(ByVal value As String)
         messageValue = value
      End Set
   End Property ' Message
End Class ' TextMessage
```

Fig. 23.14 | WCF web-service interface that returns a welcome message through REST architecture and JSON format. (Part 2 of 2.)

```vb
' Fig. 23.15: WelcomeRESTJSONService.vb
' WCF web service that returns a welcome message through REST architecture
' and JSON format.
Public Class WelcomeRESTJSONService
   Implements IWelcomeRESTJSONService

   ' returns a welcome message
   Public Function Welcome(ByVal yourName As String)
      As TextMessage Implements IWelcomeRESTJSONService.Welcome
      ' add welcome message to field of TextMessage object
      Dim welcomeString As New TextMessage
      welcomeString.Message = "Welcome to WCF Web Services with REST " &
         "and JSON, " & yourName & "!"
      Return welcomeString
   End Function ' Welcome
End Class ' WelcomeRESTJSONService
```

Fig. 23.15 | WCF web service that returns a welcome message through REST architecture and JSON format.



```
{"Message":"Welcome to WCF Web Services with REST and JSON, Paul!"}
```

Fig. 23.16 | Response from `WelcomeRESTJSONService` in JSON data format.

```vb
' Fig. 23.17: WelcomeRESTJSON.vb
' Client that consumes WelcomeRESTJSONService.
Imports System.IO
Imports System.Net
Imports System.Runtime.Serialization.Json
Imports System.Text

Public Class WelcomeRESTJSON
   ' object to invoke the WelcomeRESTJSONService
   Private WithEvents service As New WebClient()

   ' creates welcome message from text input and web service
   Private Sub submitButton_Click(ByVal sender As System.Object,
      ByVal e As System.EventArgs) Handles submitButton.Click

      ' send request to WelcomeRESTJSONService
      service.DownloadStringAsync(New Uri(
         "http://localhost:49745/WelcomeRESTJSONService/Service.svc/" &
       "welcome/" & textBox.Text))
   End Sub ' submitButton

```

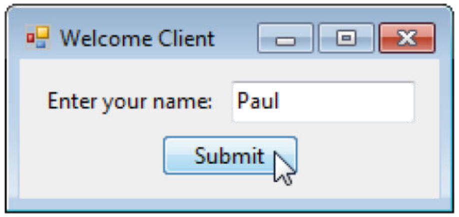
Fig. 23.17 | Client that consumes WelcomeRESTJSONService. (Part 1 of 3.)

```
      ' process web-service response
      Private Sub service_DownloadStringCompleted(ByVal sender As Object,
         ByVal e As System.Net.DownloadStringCompletedEventArgs) _
         Handles service.DownloadStringCompleted

         ' check if any errors occurred in retrieving service data
         If e.Error Is Nothing Then
            ' deserialize response into a TextMessage object
            Dim JSONSerializer _
               As New DataContractJsonSerializer(GetType(TextMessage))
            Dim welcomeString = JSONSerializer.ReadObject(
               New MemoryStream(Encoding.Unicode.GetBytes(e.Result)))

            ' display Message text
            MessageBox.Show(CType(welcomeString, TextMessage).Message)
         End If
      End Sub ' service_DownloadStringCompleted
   End Class ' WelcomeRESTJSON

```

Fig. 23.17 | Client that consumes `WelcomeRESTJSONService`. (Part 2 of 3.)

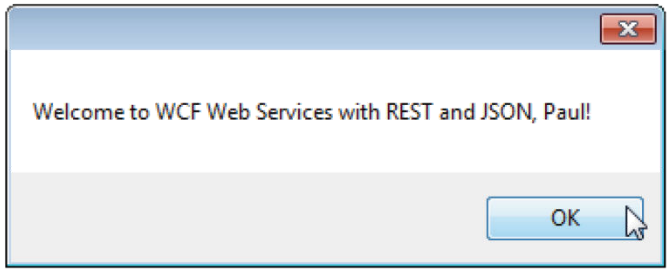
```
' TextMessage class representing a JSON object
<Serializable()>
Public Class TextMessage
   Public Message As String
End Class ' TextMessage
```

a) User inputs name.



Enter your name: Paul

b) Message sent from `WelcomeRESTJSONService`.



Welcome to WCF Web Services with REST and JSON, Paul!

Fig. 23.17 | Client that consumes `WelcomeRESTJSONService`. (Part 3 of 3.)

```
' Fig. 23.18: IBlackjackService.vb
' Blackjack game WCF web-service interface.
<ServiceContract(SessionMode:=SessionMode.Required)> _
Public Interface IBlackjackService
   ' deals a card that has not been dealt
   <OperationContract()>
   Function DealCard() As String

   ' creates and shuffles the deck
   <OperationContract()>
   Sub Shuffle()

   ' calculates value of a hand
   <OperationContract()>
   Function GetHandValue(ByVal dealt As String) As Integer
End Interface ' IBlackjackService
```

Fig. 23.18 | Blackjack game WCF web-service interface.

```
' Fig. 23.19: BlackjackService.vb
' Blackjack game WCF web service.
Imports System.Collections.Generic

<ServiceBehavior(InstanceContextMode:=InstanceContextMode.PerSession)>
Public Class BlackjackService
   Implements IBlackjackService
   ' create persistent session deck-of-cards object
   Dim deck As New List(Of String)

   ' deals card that has not yet been dealt
   Public Function DealCard() As String _
      Implements IBlackjackService.DealCard

      Dim card As String = Convert.ToString(deck(0)) ' get first card
      deck.RemoveAt(0) ' remove card from deck
      Return card
   End Function ' DealCard

```

Fig. 23.19 | Blackjack game WCF web service. (Part 1 of 4.)

```
    ' creates and shuffles a deck of cards
    Public Sub Shuffle() Implements IBlackjackService.Shuffle
       Dim randomObject As New Random() ' generates random numbers

       deck.Clear() ' clears deck for new game

       ' generate all possible cards
       For face = 1 To 13 ' loop through face values
          For suit As Integer = 0 To 3 ' loop through suits
             deck.Add(face & " " & suit) ' add card (string) to deck
          Next suit
       Next face

       ' shuffles deck by swapping each card with another card randomly
       For i = 0 To deck.Count - 1
          ' get random index
          Dim newIndex = randomObject.Next(deck.Count - 1)
          Dim temporary = deck(i) ' save current card in temporary variable
          deck(i) = deck(newIndex) ' copy randomly selected card
          deck(newIndex) = temporary ' copy current card back into deck
       Next
    End Sub ' Shuffle

```

Fig. 23.19 | Blackjack game WCF web service. (Part 2 of 4.)

```vb
    ' computes value of hand
    Public Function GetHandValue(ByVal dealt As String) As Integer _
       Implements IBlackjackService.GetHandValue
       ' split string containing all cards
       Dim tab As Char = Convert.ToChar(vbTab)
       Dim cards As String() = dealt.Split(tab) ' get array of cards
       Dim total As Integer = 0 ' total value of cards in hand
       Dim face As Integer ' face of the current card
       Dim aceCount As Integer = 0 ' number of aces in hand

       ' loop through the cards in the hand
       For Each card In cards
          ' get face of card
          face = Convert.ToInt32(card.Substring(0, card.IndexOf(" ")))

          Select Case face
             Case 1 ' if ace, increment aceCount
                aceCount += 1
             Case 11 To 13 ' if jack, queen or king add 10
                total += 10
             Case Else ' otherwise, add value of face
                total += face
          End Select
       Next

```

Fig. 23.19 | Blackjack game WCF web service. (Part 3 of 4.)

```
      ' if there are any aces, calculate optimum total
      If aceCount > 0 Then
         ' if it is possible to count one ace as 11, and the rest
         ' as 1 each, do so; otherwise, count all aces as 1 each
         If (total + 11 + aceCount - 1 <= 21) Then
            total += 11 + aceCount - 1
         Else
            total += aceCount
         End If
      End If

      Return total
   End Function ' GetHandValue
End Class ' BlackjackService
```

Fig. 23.19 | Blackjack game WCF web service. (Part 4 of 4.)

```
' Fig. 23.20: Blackjack.vb
' Blackjack game that uses the BlackjackService web service.
Imports System.Net

Public Class Blackjack
   ' reference to web service
   Private dealer As ServiceReference.BlackJackServiceClient

   ' string representing the dealer's cards
   Private dealersCards As String

   ' string representing the player's cards
   Private playersCards As String
   Private cardBoxes As List(Of PictureBox) ' list of card images
   Private currentPlayerCard As Integer ' player's current card number
   Private currentDealerCard As Integer ' dealer's current card number

```

Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 1 of 18.)

```
    ' enum representing the possible game outcomes
    Public Enum GameStatus
       PUSH ' game ends in a tie
       LOSE ' player loses
       WIN ' player wins
       BLACKJACK ' player has blackjack
    End Enum ' GameStatus

    ' sets up the game
    Private Sub Blackjack_Load(ByVal sender As Object,
       ByVal e As System.EventArgs) Handles Me.Load
       ' instantiate object allowing communication with web service
       dealer = New ServiceReference.BlackJackServiceClient()

       cardBoxes = New List(Of PictureBox)

       ' put PictureBoxes into cardBoxes List
       cardBoxes.Add(pictureBox1)
       cardBoxes.Add(pictureBox2)
       cardBoxes.Add(pictureBox3)
       cardBoxes.Add(pictureBox4)
       cardBoxes.Add(pictureBox5)
       cardBoxes.Add(pictureBox6)
```

Fig. 23.20 | Blackjack game that uses the BlackjackService web service. (Part 2 of 18.)

```
            cardBoxes.Add(pictureBox7)
            cardBoxes.Add(pictureBox8)
            cardBoxes.Add(pictureBox9)
            cardBoxes.Add(pictureBox10)
            cardBoxes.Add(pictureBox11)
            cardBoxes.Add(pictureBox12)
            cardBoxes.Add(pictureBox13)
            cardBoxes.Add(pictureBox14)
            cardBoxes.Add(pictureBox15)
            cardBoxes.Add(pictureBox16)
            cardBoxes.Add(pictureBox17)
            cardBoxes.Add(pictureBox18)
            cardBoxes.Add(pictureBox19)
            cardBoxes.Add(pictureBox20)
            cardBoxes.Add(pictureBox21)
            cardBoxes.Add(pictureBox22)
        End Sub ' Blackjack_Load

```

Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 3 of 18.)

```
   ' deals cards to dealer while dealer's total is less than 17,
   ' then computes value of each hand and determines winner
   Private Sub DealerPlay()
      ' reveal dealer's second card
      Dim tab As Char = Convert.ToChar(vbTab)
      Dim cards As String() = dealersCards.Split(tab)
      DisplayCard(1, cards(1))

      Dim nextCard As String

      ' while value of dealer's hand is below 17,
      ' dealer must take cards
      While dealer.GetHandValue(dealersCards) < 17
         nextCard = dealer.DealCard() ' deal new card
         dealersCards &= vbTab & nextCard

         ' update GUI to show new card
         MessageBox.Show("Dealer takes a card")
         DisplayCard(currentDealerCard, nextCard)
         currentDealerCard += 1
      End While

```

Fig. 23.20 | Blackjack game that uses the BlackjackService web service. (Part 4 of 18.)

```
         Dim dealerTotal As Integer = dealer.GetHandValue(dealersCards)
         Dim playerTotal As Integer = dealer.GetHandValue(playersCards)

         ' if dealer busted, player wins
         If dealerTotal > 21 Then
            GameOver(GameStatus.WIN)
         Else
            ' if dealer and player have not exceeded 21,
            ' higher score wins; equal scores is a push.
            If dealerTotal > playerTotal Then ' player loses game
               GameOver(GameStatus.LOSE)
            ElseIf playerTotal > dealerTotal Then ' player wins game
               GameOver(GameStatus.WIN)
            Else ' player and dealer tie
               GameOver(GameStatus.PUSH)
            End If
         End If
      End Sub ' DealerPlay

```

Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 5 of 18.)

```
   ' displays card represented by cardValue in specified PictureBox
   Public Sub DisplayCard(
      ByVal card As Integer, ByVal cardValue As String)
      ' retrieve appropriate PictureBox
      Dim displayBox As PictureBox = cardBoxes(card)

      ' if string representing card is empty,
      ' set displayBox to display back of card
      If String.IsNullOrEmpty(cardValue) Then
         displayBox.Image =
            Image.FromFile("blackjack_images/cardback.png")
         Return
      End If

      ' retrieve face value of card from cardValue
      Dim face As String =
         cardValue.Substring(0, cardValue.IndexOf(" "))

      ' retrieve the suit of the card from cardValue
      Dim suit As String =
         cardValue.Substring(cardValue.IndexOf(" ") + 1)

      Dim suitLetter As Char ' suit letter used to form image file name
```

Fig. 23.20 | Blackjack game that uses the BlackjackService web service. (Part 6 of 18.)

```
      ' determine the suit letter of the card
      Select Case Convert.ToInt32(suit)
         Case 0 ' clubs
            suitLetter = "c"c
         Case 1 ' diamonds
            suitLetter = "d"c
         Case 2 ' hearts
            suitLetter = "h"c
         Case Else ' spades
            suitLetter = "s"c
      End Select

      ' set displayBox to display appropriate image
      displayBox.Image = Image.FromFile(
         "blackjack_images/" & face & suitLetter & ".png")
   End Sub ' DisplayCard

```

Fig. 23.20 | Blackjack game that uses the BlackjackService web service. (Part 7 of 18.)

```
   ' displays all player cards and shows
   ' appropriate game status message
   Public Sub GameOver(ByVal winner As GameStatus)
      ' display appropriate status image
      If winner = GameStatus.PUSH Then ' push
         statusPictureBox.Image =
            Image.FromFile("blackjack_images/tie.png")
      ElseIf winner = GameStatus.LOSE Then ' player loses
         statusPictureBox.Image =
            Image.FromFile("blackjack_images/lose.png")
      ElseIf winner = GameStatus.BLACKJACK Then
         ' player has blackjack
         statusPictureBox.Image =
            Image.FromFile("blackjack_images/blackjack.png")
      Else ' player wins
         statusPictureBox.Image =
            Image.FromFile("blackjack_images/win.png")
      End If

      ' display final totals for dealer and player
      dealerTotalLabel.Text =
         "Dealer: " & dealer.GetHandValue(dealersCards)
```

Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 8 of 18.)

```
         playerTotalLabel.Text =
            "Player: " & dealer.GetHandValue(playersCards)

         ' reset controls for new game
         stayButton.Enabled = False
         hitButton.Enabled = False
         dealButton.Enabled = True
      End Sub ' GameOver

      ' deal two cards each to dealer and player
      Private Sub dealButton_Click(ByVal sender As System.Object,
         ByVal e As System.EventArgs) Handles dealButton.Click
         Dim card As String ' stores a card temporarily until added to a hand

         ' clear card images
         For Each cardImage As PictureBox In cardBoxes
            cardImage.Image = Nothing
         Next

         statusPictureBox.Image = Nothing ' clear status image
         dealerTotalLabel.Text = String.Empty ' clear final total for dealer
         playerTotalLabel.Text = String.Empty ' clear final total for player

```

Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 9 of 18.)

```
        ' create a new, shuffled deck on the web service host
        dealer.Shuffle()

        ' deal two cards to player
        playersCards = dealer.DealCard() ' deal a card to player's hand

        ' update GUI to display new card
        DisplayCard(11, playersCards)
        card = dealer.DealCard() ' deal a second card
        DisplayCard(12, card) ' update GUI to display new card
        playersCards &= vbTab & card ' add second card to player's hand

        ' deal two cards to dealer, only display face of first card
        dealersCards = dealer.DealCard() ' deal a card to dealer's hand
        DisplayCard(0, dealersCards) ' update GUI to display new card
        card = dealer.DealCard() ' deal a second card
        DisplayCard(1, String.Empty) ' update GUI to show face-down card
        dealersCards &= vbTab & card ' add second card to dealer's hand

        stayButton.Enabled = True ' allow player to stay
        hitButton.Enabled = True ' allow player to hit
        dealButton.Enabled = False ' disable Deal Button

```

Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 10 of 18.)

```
        ' determine the value of the two hands
        Dim dealerTotal As Integer = dealer.GetHandValue(dealersCards)
        Dim playerTotal As Integer = dealer.GetHandValue(playersCards)

        ' if hands equal 21, it is a push
        If dealerTotal = playerTotal And dealerTotal = 21 Then
           GameOver(GameStatus.PUSH)
        ElseIf dealerTotal = 21 Then ' if dealer has 21, dealer wins
           GameOver(GameStatus.LOSE)
        ElseIf playerTotal = 21 Then ' player has blackjack
           GameOver(GameStatus.BLACKJACK)
        End If

        currentDealerCard = 2 ' next dealer card has index 2 in cardBoxes
        currentPlayerCard = 13 ' next player card has index 13 in cardBoxes
     End Sub ' dealButton_Click

```

Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 11 of 18.)

```
   ' deal another card to player
   Private Sub hitButton_Click(ByVal sender As System.Object,
      ByVal e As System.EventArgs) Handles hitButton.Click
      ' get player another card
      Dim card As String = dealer.DealCard() ' deal new card
      playersCards &= vbTab & card ' add new card to player's hand

      ' update GUI to show new card
      DisplayCard(currentPlayerCard, card)
      currentPlayerCard += 1

      ' determine the value of the player's hand
      Dim total As Integer = dealer.GetHandValue(playersCards)

      ' if player exceeds 21, house wins
      If total > 21 Then
         GameOver(GameStatus.LOSE)
      End If

      ' if player has 21,
      ' they cannot take more cards, and dealer plays
      If total = 21 Then
         hitButton.Enabled = False
```

Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 12 of 18.)

```
            DealerPlay()
         End If
      End Sub ' hitButton_Click

      ' play the dealer's hand after the play chooses to stay
      Private Sub stayButton_Click(ByVal sender As System.Object,
         ByVal e As System.EventArgs) Handles stayButton.Click
         stayButton.Enabled = False ' disable Stay Button
         hitButton.Enabled = False ' disable Hit Button
         dealButton.Enabled = True ' re-enable Deal Button
         DealerPlay() ' player chose to stay, so play the dealer's hand
      End Sub ' stayButton_Click
   End Class ' Blackjack
```

Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 13 of 18.)

a) Initial cards dealt to the player and the dealer when the user presses the **Deal** button.

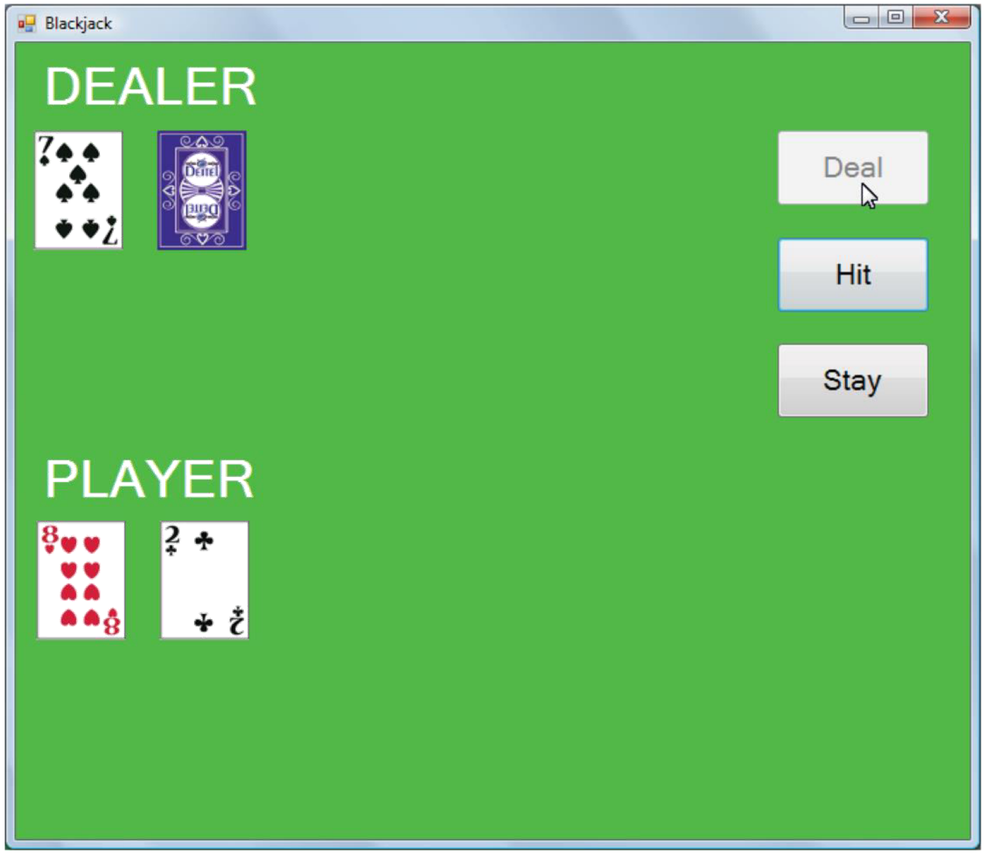


Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 14 of 18.)

b) Cards after the player presses the **Hit** button once, then the **Stay** button. In this case, the player wins the game with a higher total than the dealer.

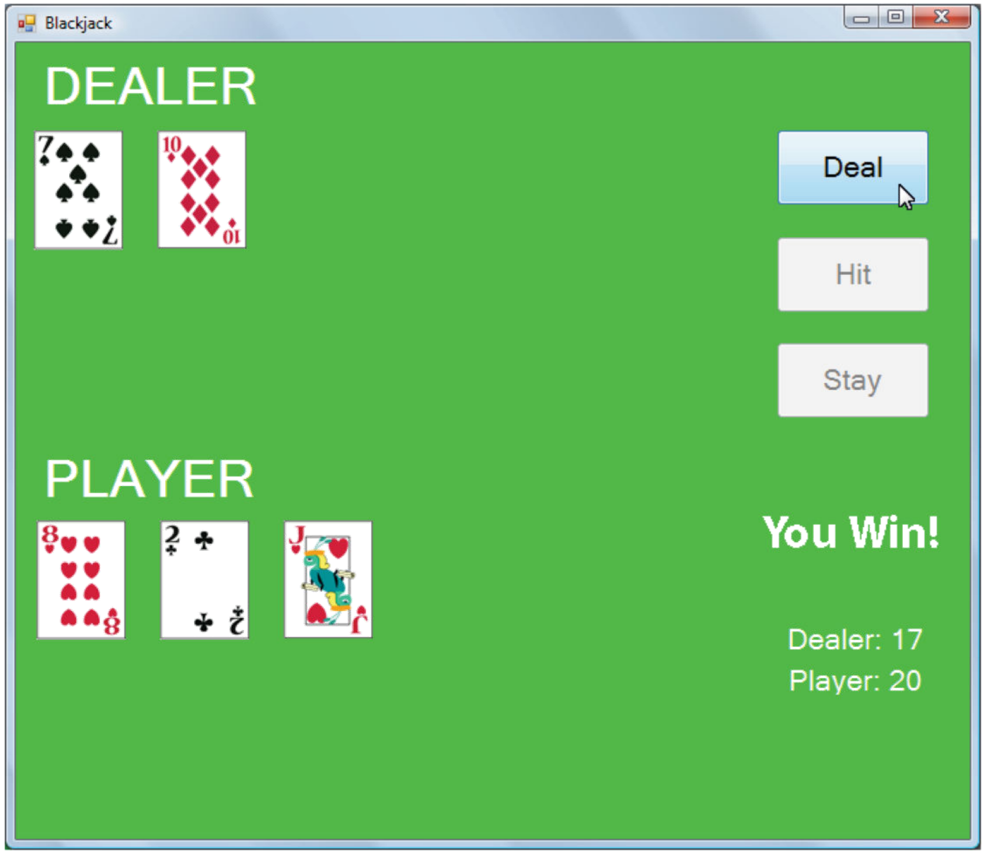
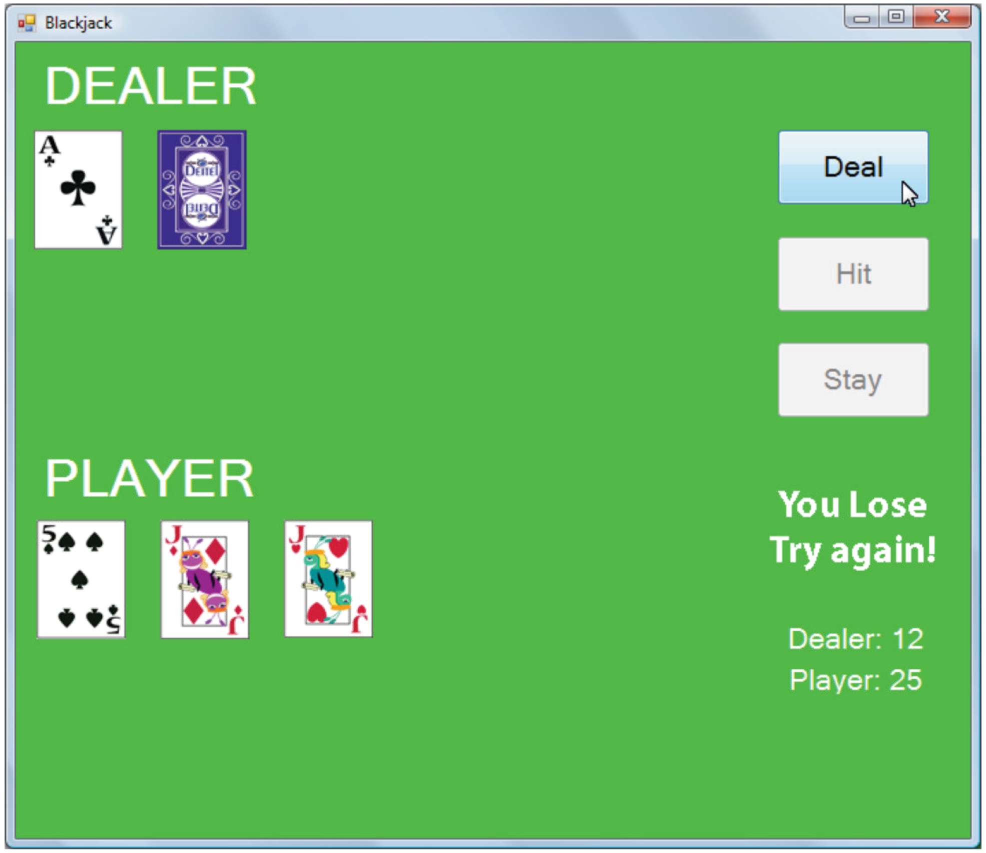


Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 15 of 18.)

c) Cards after the player presses the **Hit** button once, then the **Stay** button. In this case, the player busts (exceeds 21) and the dealer wins the game.

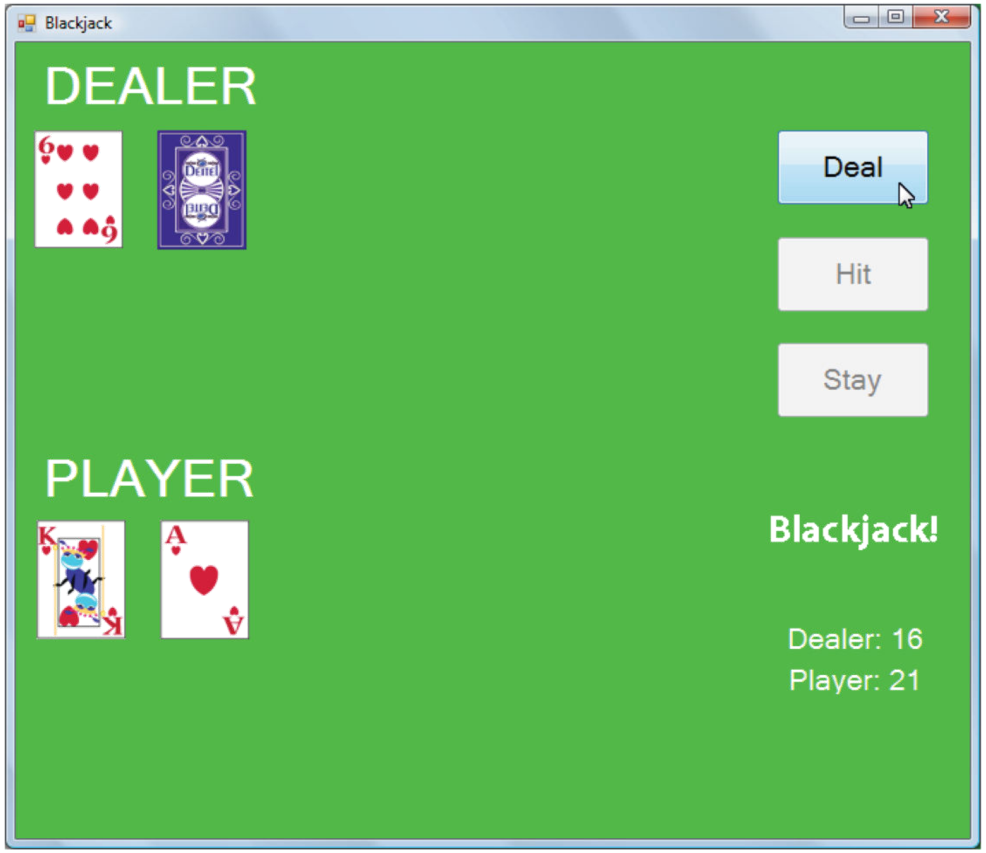


You Lose
Try again!

Dealer: 12
Player: 25

Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 16 of 18.)

d) Cards after the player presses the **Deal** button. In this case, the player wins with Blackjack because the first two cards are an ace and a card with a value of 10 (a jack in this case).

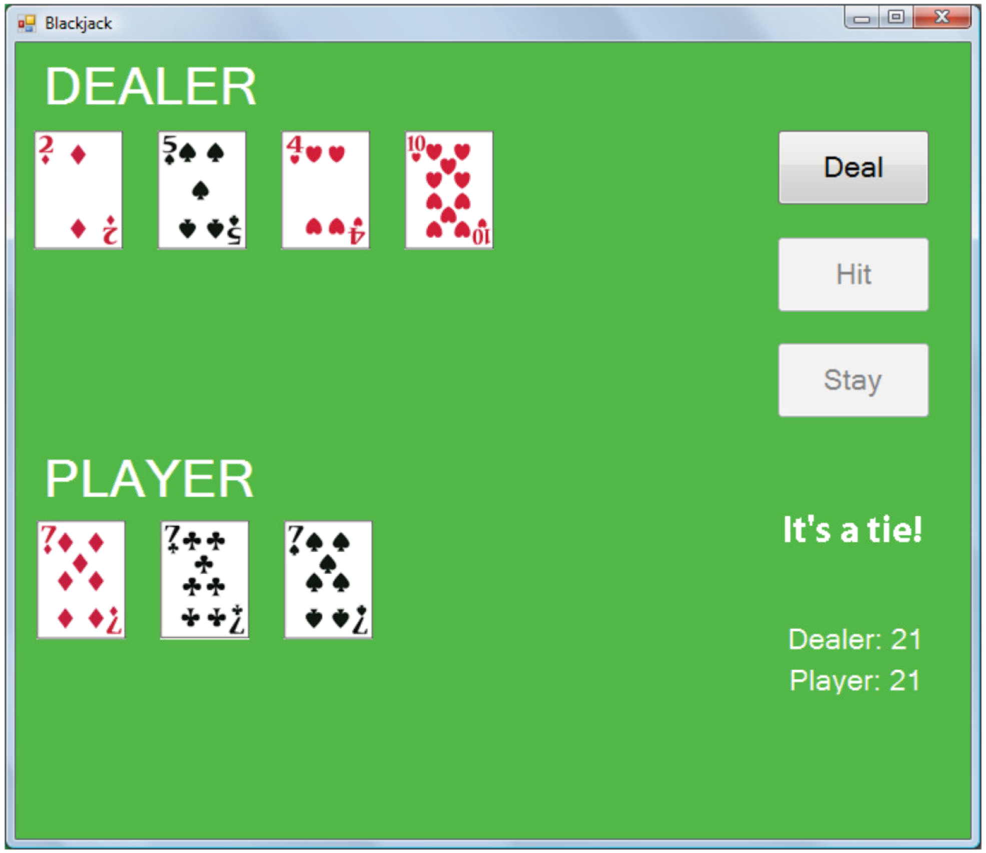


Blackjack!

Dealer: 16
Player: 21

Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 17 of 18.)

e) Cards after the player presses the **Stay** button. In this case, the player and dealer push—they have the same card total.



It's a tie!

Dealer: 21
Player: 21

Fig. 23.20 | Blackjack game that uses the `BlackjackService` web service. (Part 18 of 18.)

```
' Fig. 23.21: IReservationService.vb
' Airline reservation WCF web-service interface.
<ServiceContract()>
Public Interface IReservationService
   ' reserves a seat
   <OperationContract()>
   Function Reserve(ByVal seatType As String,
      ByVal classType As String) As Boolean
End Interface ' IReservationService
```

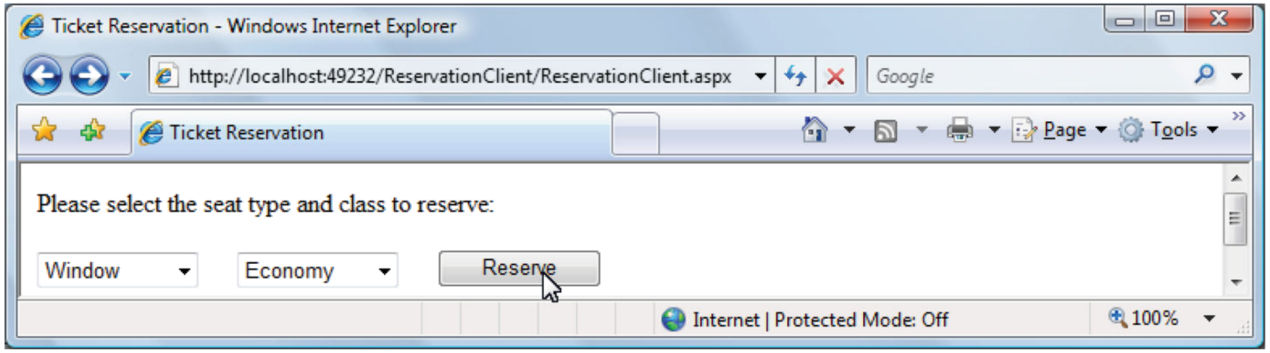
Fig. 23.21 | Airline reservation WCF web-service interface.

```
' Fig. 23.22: ReservationService.vb
' Airline reservation WCF web service.
Public Class ReservationService
   Implements IReservationService

   ' create ticketsDB object to access Tickets database
   Private ticketsDB As New TicketsDataContext()

   ' checks database to determine whether matching seat is available
   Public Function Reserve(ByVal seatType As String,
      ByVal classType As String) As Boolean _
      Implements IReservationService.Reserve

      ' LINQ query to find seats matching the parameters
      Dim result =
         From seat In ticketsDB.Seats
         Where (seat.Taken = 0) And (seat.SeatType = seatType)
            And (seat.SeatClass = classType)

```

Fig. 23.22 | Airline reservation WCF web service. (Part 1 of 2.)

```
        ' if the number of seats returned is nonzero,
        ' obtain the first matching seat number and mark it as taken
        If result.Count() <> 0 Then
           ' get first available seat
           Dim firstAvailableSeat As Seat = result.First()
           firstAvailableSeat.Taken = 1 ' mark the seat as taken
           ticketsDB.SubmitChanges() ' update
           Return True ' seat was reserved
        End If

        Return False ' no seat was reserved
     End Function ' Reserve
  End Class ' ReservationService
```

Fig. 23.22 | Airline reservation WCF web service. (Part 2 of 2.)


Ticket Reservation - Windows Internet Explorer
http://localhost:49232/ReservationClient/ReservationClient.aspx
Google
Ticket Reservation
Page
Tools
Internet | Protected Mode: Off
100%

Please select the seat type and class to reserve:

Window | Economy | Reserve

Fig. 23.23 | ASPX file that takes reservation information.

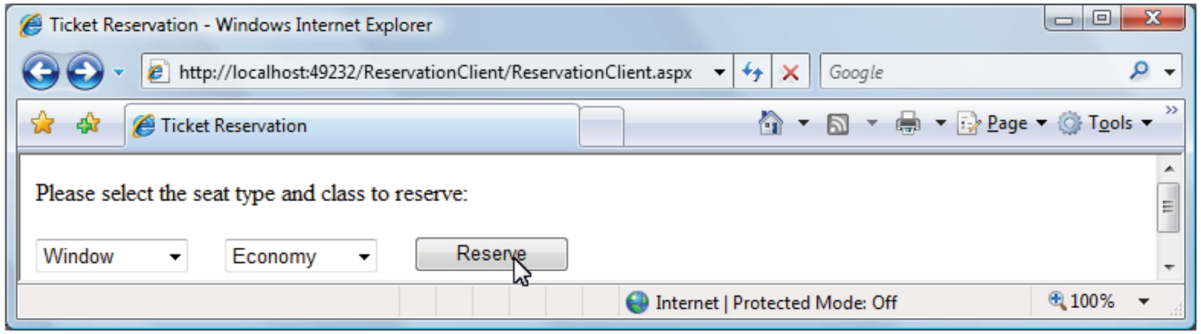
```vb
' Fig. 23.24: ReservationClient.aspx.vb
' ReservationClient code-behind file.
Partial Class ReservationClient
   Inherits System.Web.UI.Page
   ' object of proxy type used to connect to Reservation service
   Private ticketAgent As New ServiceReference.ReservationServiceClient()

   Protected Sub reserveButton_Click(ByVal sender As Object,
      ByVal e As System.EventArgs) Handles reserveButton.Click

      ' if the ticket is reserved
      If ticketAgent.Reserve(seatList.SelectedItem.Text,
         classList.SelectedItem.Text.ToString()) Then

         ' hide other controls
         instructionsLabel.Visible = False
         seatList.Visible = False
         classList.Visible = False
         reserveButton.Visible = False
         errorLabel.Visible = False

         ' display message indicating success
         Response.Write("Your reservation has been made. Thank you.")
```

Fig. 23.24 | ReservationClient code-behind file. (Part 1 of 2.)

```
        Else ' service method returned false, so signal failure
           ' display message in the initially blank errorLabel
           errorLabel.Text = "This type of seat is not available. " &
              "Please modify your request and try again."
        End If
     End Sub ' reserveButton_Click
  End Class ' ReservationClient
```

Fig. 23.24 | ReservationClient code-behind file. (Part 2 of 2.)

a) Selecting a seat.

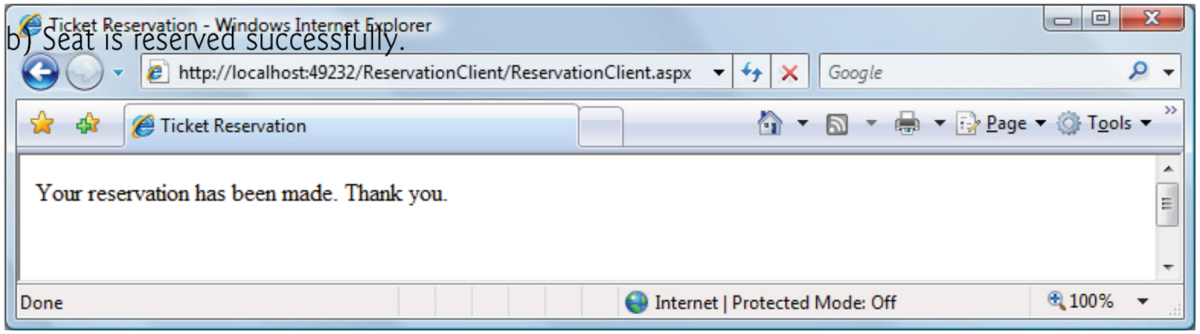

Ticket Reservation - Windows Internet Explorer
http://localhost:49232/ReservationClient/ReservationClient.aspx
Google
Ticket Reservation
Page
Tools

Please select the seat type and class to reserve:

Window | Economy | Reserve

Internet | Protected Mode: Off
100%

b) Seat is reserved successfully.

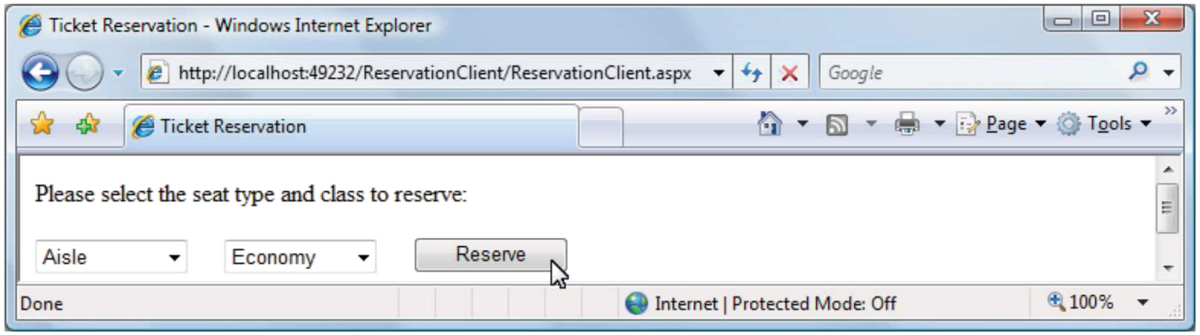

Ticket Reservation - Windows Internet Explorer
http://localhost:49232/ReservationClient/ReservationClient.aspx
Google
Ticket Reservation
Page
Tools

Your reservation has been made. Thank you.

Done
Internet | Protected Mode: Off
100%

Fig. 23.25 | Ticket reservation web-application sample execution. (Part 1 of 2.)

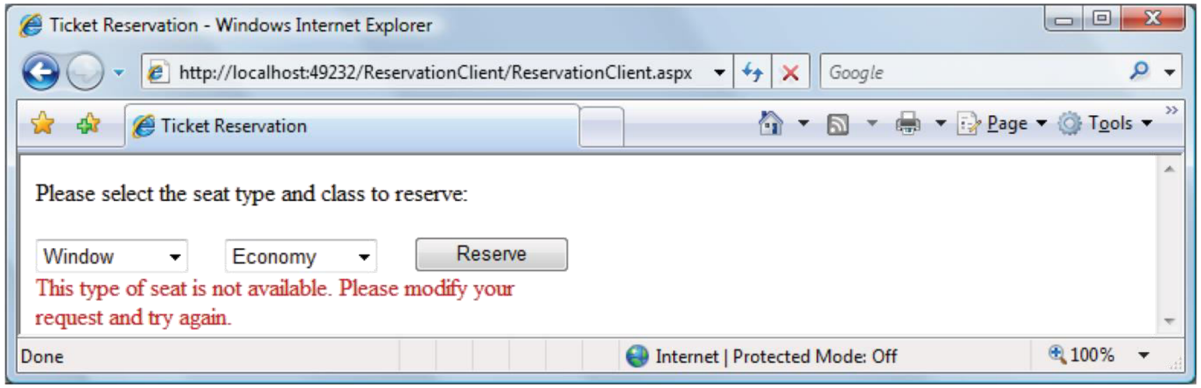
c) Attempting to reserve another seat.


Ticket Reservation - Windows Internet Explorer
http://localhost:49232/ReservationClient/ReservationClient.aspx
Ticket Reservation
Done
Internet | Protected Mode: Off
100%

Please select the seat type and class to reserve:

Aisle | Economy | Reserve

d) No seats match the requested type and class.


Ticket Reservation - Windows Internet Explorer
http://localhost:49232/ReservationClient/ReservationClient.aspx
Ticket Reservation
Done
Internet | Protected Mode: Off
100%

Please select the seat type and class to reserve:

Window | Economy | Reserve

This type of seat is not available. Please modify your request and try again.

Fig. 23.25 | Ticket reservation web-application sample execution. (Part 2 of 2.)

```
' Fig. 23.26: Equation.vb
' Class Equation that contains information about an equation.
<DataContract()>
Public Class Equation
   Private leftOperand As Integer ' number to the left of the operator
   Private rightOperand As Integer ' number to the right of the operator
   Private resultValue As Integer ' result of the operation
   Private operationType As String ' type of the operation

   ' required default constructor
   Public Sub New()
      MyClass.New(0, 0, "add")
   End Sub ' parameterless New

```

Fig. 23.26 | Class Equation that contains information about an equation. (Part 1 of 6.)

```
   ' three-argument constructor for class Equation
   Public Sub New(ByVal leftValue As Integer,
      ByVal rightValue As Integer, ByVal type As String)

      Left = leftValue
      Right = rightValue

      Select Case type ' perform appropriate operation
         Case "add" ' addition
            Result = leftOperand + rightOperand
            operationType = "+"
         Case "subtract" ' subtraction
            Result = leftOperand - rightOperand
            operationType = "-"
         Case "multiply" ' multiplication
            Result = leftOperand * rightOperand
            operationType = "*"
      End Select
   End Sub ' three-parameter New

```

Fig. 23.26 | Class Equation that contains information about an equation. (Part 2 of 6.)

```
    ' return string representation of the Equation object
    Public Overrides Function ToString() As String
       Return leftOperand.ToString() & " " & operationType & " " &
          rightOperand.ToString() & " = " & resultValue.ToString()
    End Function ' ToString

    ' property that returns a string representing left-hand side
    <DataMember()>
    Public Property LeftHandSide() As String
       Get
          Return leftOperand.ToString() & " " & operationType & " " &
             rightOperand.ToString()
       End Get

       Set(ByVal value As String) ' required set accessor
          ' empty body
       End Set
    End Property ' LeftHandSide

```

Fig. 23.26 | Class Equation that contains information about an equation. (Part 3 of 6.)

```
   ' property that returns a string representing right-hand side
   <DataMember()>
   Public Property RightHandSide() As String
      Get
         Return resultValue.ToString()
      End Get

      Set(ByVal value As String) ' required set accessor
         ' empty body
      End Set
   End Property ' RightHandSide

   ' property to access the left operand
   <DataMember()>
   Public Property Left() As Integer
      Get
         Return leftOperand
      End Get

      Set(ByVal value As Integer)
         leftOperand = value
      End Set
   End Property ' Left

```

Fig. 23.26 | Class Equation that contains information about an equation. (Part 4 of 6.)

```
    ' property to access the right operand
    <DataMember()>
    Public Property Right() As Integer
       Get
          Return rightOperand
       End Get

       Set(ByVal value As Integer)
          rightOperand = value
       End Set
    End Property ' Right

    ' property to access the result of applying
    ' an operation to the left and right operands
    <DataMember()>
    Public Property Result() As Integer
       Get
          Return resultValue
       End Get

       Set(ByVal value As Integer)
          resultValue = value
       End Set
    End Property ' Result
```

Fig. 23.26 | Class Equation that contains information about an equation. (Part 5 of 6.)

```

   ' property to access the operation
   <DataMember()>
   Public Property Operation() As String
      Get
         Return operationType
      End Get

      Set(ByVal value As String)
         operationType = value
      End Set
   End Property ' Operation
End Class ' Equation
```

Fig. 23.26 | Class `Equation` that contains information about an equation. (Part 6 of 6.)

```
' Fig. 23.27: IEquationGeneratorService.vb
' WCF REST service interface to create random equations based on a
' specified operation and difficulty level.
Imports System.ServiceModel.Web

<ServiceContract()>
Public Interface IEquationGeneratorService
   ' method to generate a math equation
   <OperationContract()>
   <WebGet(UriTemplate:="equation/{operation}/{level}")>
   Function GenerateEquation(ByVal operation As String,
      ByVal level As String) As Equation
End Interface ' IEquationGeneratorService
```

Fig. 23.27 | WCF REST service interface to create random equations based on a specified operation and difficulty level.

```
' Fig. 23.28: EquationGeneratorService.vb
' WCF REST service to create random equations based on a
' specified operation and difficulty level.
Public Class EquationGeneratorService
   Implements IEquationGeneratorService
   ' method to generate a math equation
   Public Function GenerateEquation(ByVal operation As String,
      ByVal level As String) As Equation _
      Implements IEquationGeneratorService.GenerateEquation

      ' convert level from String to Integer
      Dim digits = Convert.ToInt32(level)

      ' calculate maximum and minimum number to be used
      Dim maximum As Integer = Convert.ToInt32(Math.Pow(10, digits))
      Dim minimum As Integer = Convert.ToInt32(Math.Pow(10, digits - 1))

      Dim randomObject As New Random() ' used to generate random numbers

```

Fig. 23.28 | WCF REST service to create random equations based on a specified operation and difficulty level. (Part 1 of 2.)

```
        ' create Equation consisting of two random
        ' numbers in the range minimum to maximum
        Dim newEquation As New Equation(
           randomObject.Next(minimum, maximum),
           randomObject.Next(minimum, maximum), operation)

        Return newEquation
     End Function ' GenerateEquation
  End Class ' EquationGeneratorService
```

Fig. 23.28 | WCF REST service to create random equations based on a specified operation and difficulty level. (Part 2 of 2.)

```
' Fig. 23.29: MathTutor.vb
' Math tutor using EquationGeneratorService to create equations.
Imports System.Net
Imports System.Xml.Linq
Imports <xmlns="http://schemas.datacontract.org/2004/07/">

Public Class MathTutor
   Private operation As String = "add" ' the default operation
   Private level As Integer = 1 ' the default difficulty level
   Private leftHandSide As String ' the left side of the equation
   Private result As Integer ' the answer

   Private WithEvents service As New WebClient() ' used to invoke service

   ' generates a new equation when user clicks button
   Private Sub generateButton_Click(ByVal sender As System.Object,
      ByVal e As System.EventArgs) Handles generateButton.Click

      ' send request to EquationGeneratorServiceXML
      service.DownloadStringAsync(New Uri(
         "http://localhost:49593/EquationGeneratorServiceXML/" &
         "Service.svc/equation/" & operation & "/" & level))
   End Sub ' generateButton_Click
```

Fig. 23.29 | Math tutor using XML version of EquationGeneratorService to create equations. (Part 1 of 7.)

```
      ' process web-service response
      Private Sub service_DownloadStringCompleted(ByVal sender As Object,
         ByVal e As System.Net.DownloadStringCompletedEventArgs) _
         Handles service.DownloadStringCompleted

         ' check if any errors occurred in retrieving service data
         If e.Error Is Nothing Then
            ' parse response and get LeftHandSide and Result values
            Dim xmlResponse = XDocument.Parse(e.Result)
            leftHandSide = xmlResponse.<Equation>.<LeftHandSide>.Value
            result = Convert.ToInt32(xmlResponse.<Equation>.<Result>.Value)

            questionLabel.Text = leftHandSide ' display left side of equation
            okButton.Enabled = True ' enable okButton
            answerTextBox.Enabled = True ' enable answerTextBox
         End If
      End Sub ' service_DownloadStringCompleted

```

Fig. 23.29 | Math tutor using XML version of EquationGeneratorService to create equations. (Part 2 of 7.)

```
   ' check user's answer
   Private Sub okButton_Click(ByVal sender As System.Object,
      ByVal e As System.EventArgs) Handles okButton.Click

      If Not String.IsNullOrEmpty(answerTextBox.Text) Then
         ' get user's answer
         Dim userAnswer As Integer = Convert.ToInt32(answerTextBox.Text)

         ' determine whether user's answer is correct
         If result = userAnswer Then
            questionLabel.Text = String.Empty ' clear question
            answerTextBox.Clear() ' clear answer
            okButton.Enabled = False ' disable OK button
            MessageBox.Show("Correct! Good job!")
         Else
            MessageBox.Show("Incorrect. Try again.")
         End If
      End If
   End Sub ' okButton_Click

```

Fig. 23.29 | Math tutor using XML version of EquationGeneratorService to create equations. (Part 3 of 7.)

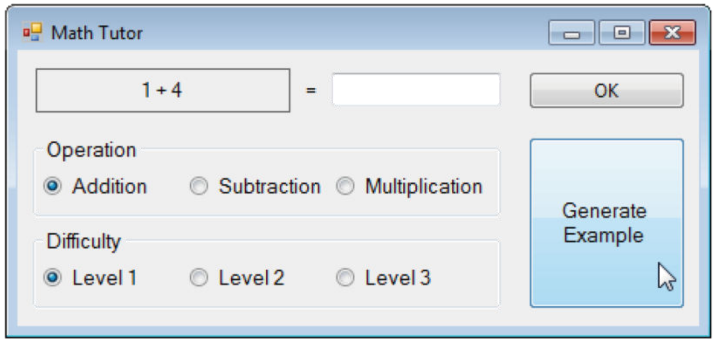
```
   ' set the operation to addition
   Private Sub additionRadioButton_CheckedChanged(
      ByVal sender As System.Object, ByVal e As System.EventArgs) _
      Handles additionRadioButton.CheckedChanged
      operation = "add"
   End Sub ' additionRadioButton_CheckedChanged

   ' set the operation to subtraction
   Private Sub subtractionRadioButton_CheckedChanged(
      ByVal sender As System.Object, ByVal e As System.EventArgs) _
      Handles subtractionRadioButton.CheckedChanged
      operation = "subtract"
   End Sub ' subtractionRadioButton_CheckedChanged

   ' set the operation to multiplication
   Private Sub multiplicationRadioButton_CheckedChanged(
      ByVal sender As System.Object, ByVal e As System.EventArgs) _
      Handles multiplicationRadioButton.CheckedChanged
      operation = "multiply"
   End Sub ' multiplicationRadioButton_CheckedChanged

```

Fig. 23.29 | Math tutor using XML version of EquationGeneratorService to create equations. (Part 4 of 7.)

```
      ' set difficulty level to 1
      Private Sub levelOneRadioButton_CheckedChanged(
         ByVal sender As System.Object, ByVal e As System.EventArgs) _
         Handles levelOneRadioButton.CheckedChanged
         level = 1
      End Sub ' levelOneRadioButton_CheckedChanged

      ' set difficulty level to 2
      Private Sub levelTwoRadioButton_CheckedChanged(
         ByVal sender As System.Object, ByVal e As System.EventArgs) _
         Handles levelTwoRadioButton.CheckedChanged
         level = 2
      End Sub ' levelTwoRadioButton_CheckedChanged

      ' set difficulty level to 3
      Private Sub levelThreeRadioButton_CheckedChanged(
         ByVal sender As System.Object, ByVal e As System.EventArgs) _
         Handles levelThreeRadioButton.CheckedChanged
         level = 3
      End Sub ' levelThreeRadioButton_CheckedChanged
   End Class ' MathTutor
```

Fig. 23.29 | Math tutor using XML version of `EquationGeneratorService` to create equations. (Part 5 of 7.)

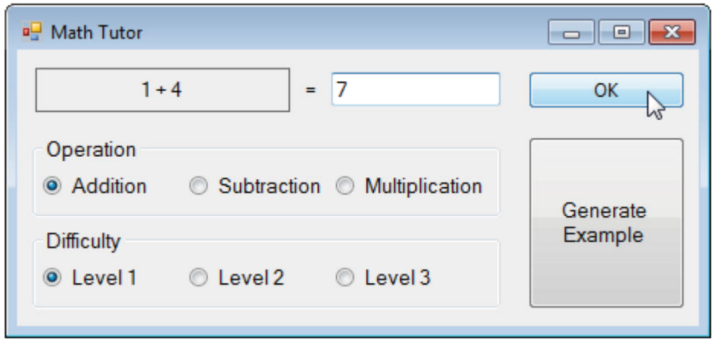
a) Generating a level 1 addition equation.



Operation: [✓ Addition] Subtraction Multiplication

Difficulty: [✓ Level 1] Level 2 Level 3

b) Answering the question incorrectly.



1 + 4 = 7

Operation: [✓ Addition] Subtraction Multiplication

Difficulty: [✓ Level 1] Level 2 Level 3

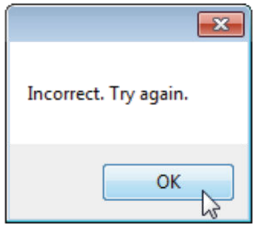
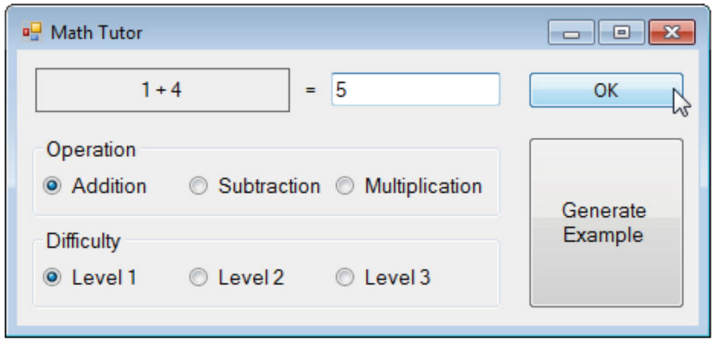

Incorrect. Try again.

Fig. 23.29 | Math tutor using XML version of `EquationGeneratorService` to create equations. (Part 6 of 7.)

c) Answering the question correctly.



5

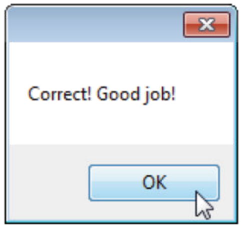

Correct! Good job!

Fig. 23.29 | Math tutor using XML version of `EquationGeneratorService` to create equations. (Part 7 of 7.)

```vb
' Fig. 23.30: IEquationGeneratorService.vb
' WCF REST service interface to create random equations based on a
' specified operation and difficulty level.
Imports System.ServiceModel.Web

<ServiceContract()>
Public Interface IEquationGeneratorService
   ' method to generate a math equation
   <OperationContract()>
   <WebGet(ResponseFormat:=WebMessageFormat.Json,
      UriTemplate:="equation/{operation}/{level}")>
   Function GenerateEquation(ByVal operation As String,
      ByVal level As String) As Equation
End Interface ' IEquationGeneratorService
```

Fig. 23.30 | WCF REST service interface to create random equations based on a specified operation and difficulty level.

```
' Fig. 23.31: MathTutor.vb
' Math tutor using EquationGeneratorServiceJSON to create equations.
Imports System.Net
Imports System.IO
Imports System.Text
Imports System.Runtime.Serialization.Json

Public Class MathTutor
   Private operation As String = "add" ' the default operation
   Private level As Integer = 1 ' the default difficulty level
   Private currentEquation As Equation ' represents the Equation
   Private WithEvents service As New WebClient() ' used to invoke service

   ' generates a new equation when user clicks button
   Private Sub generateButton_Click(ByVal sender As System.Object,
      ByVal e As System.EventArgs) Handles generateButton.Click

      ' send request to EquationGeneratorServiceJSON
      service.DownloadStringAsync(New Uri(
         "http://localhost:49817/EquationGeneratorServiceJSON/" &
         "Service.svc/equation/" & operation & "/" & level))
   End Sub ' generateButton_Click
```

Fig. 23.31 | Math tutor using JSON version of EquationGeneratorServiceJSON. (Part 1 of 7.)

```
   ' process web-service response
   Private Sub service_DownloadStringCompleted(ByVal sender As Object,
      ByVal e As System.Net.DownloadStringCompletedEventArgs) _
      Handles service.DownloadStringCompleted

      ' check if any errors occurred in retrieving service data
      If e.Error Is Nothing Then
         ' deserialize response into an equation object
         Dim JSONSerializer As New
            DataContractJsonSerializer(GetType(Equation))
         currentEquation = CType(JSONSerializer.ReadObject(New
            MemoryStream(Encoding.Unicode.GetBytes(e.Result))), Equation)

         ' display left side of equation
         questionLabel.Text = currentEquation.LeftHandSide
         okButton.Enabled = True ' enable okButton
         answerTextBox.Enabled = True ' enable answerTextBox
      End If
   End Sub ' service_DownloadStringCompleted

```

Fig. 23.31 | Math tutor using JSON version of EquationGeneratorServiceJSON. (Part 2 of 7.)

```
   ' check user's answer
   Private Sub okButton_Click(ByVal sender As System.Object,
      ByVal e As System.EventArgs) Handles okButton.Click

      ' check if answer field is filled
      If Not String.IsNullOrEmpty(answerTextBox.Text) Then
         ' determine whether user's answer is correct
         If currentEquation.Result =
            Convert.ToInt32(answerTextBox.Text) Then

            questionLabel.Text = String.Empty ' clear question
            answerTextBox.Clear() ' clear answer
            okButton.Enabled = False ' disable OK button
            MessageBox.Show("Correct! Good job!")
         Else
            MessageBox.Show("Incorrect. Try again.")
         End If
      End If
   End Sub ' okButton_Click

```

Fig. 23.31 | Math tutor using JSON version of EquationGeneratorServiceJSON. (Part 3 of 7.)

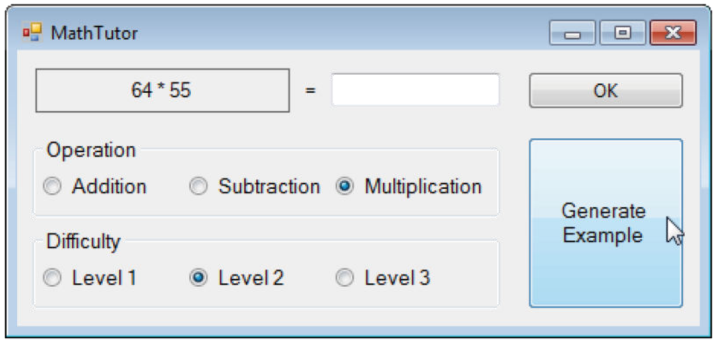
```
      ' set the operation to addition
      Private Sub additionRadioButton_CheckedChanged(
         ByVal sender As System.Object, ByVal e As System.EventArgs) _
         Handles additionRadioButton.CheckedChanged
         operation = "add"
      End Sub ' additionRadioButton_CheckedChanged

      ' set the operation to subtraction
      Private Sub subtractionRadioButton_CheckedChanged(
         ByVal sender As System.Object, ByVal e As System.EventArgs) _
         Handles subtractionRadioButton.CheckedChanged

         operation = "subtract"
      End Sub ' subtractionRadioButton_CheckedChanged

      ' set the operation to multiplication
      Private Sub multiplicationRadioButton_CheckedChanged(
         ByVal sender As System.Object, ByVal e As System.EventArgs) _
         Handles multiplicationRadioButton.CheckedChanged
         operation = "multiply"
      End Sub ' multiplicationRadioButton_CheckedChanged

```

Fig. 23.31 | Math tutor using JSON version of EquationGeneratorServiceJSON. (Part 4 of 7.)

```
   ' set difficulty level to 1
   Private Sub levelOneRadioButton_CheckedChanged(
      ByVal sender As System.Object, ByVal e As System.EventArgs) _
      Handles levelOneRadioButton.CheckedChanged
      level = 1
   End Sub ' levelOneRadioButton_CheckedChanged

   ' set difficulty level to 2
   Private Sub levelTwoRadioButton_CheckedChanged(
      ByVal sender As System.Object, ByVal e As System.EventArgs) _
      Handles levelTwoRadioButton.CheckedChanged
      level = 2
   End Sub ' levelTwoRadioButton_CheckedChanged

   ' set difficulty level to 3
   Private Sub levelThreeRadioButton_CheckedChanged(
      ByVal sender As System.Object, ByVal e As System.EventArgs) _
      Handles levelThreeRadioButton.CheckedChanged
      level = 3
   End Sub ' levelThreeRadioButton_CheckedChanged
End Class ' MathTutor
```

Fig. 23.31 | Math tutor using JSON version of EquationGeneratorServiceJSON. (Part 5 of 7.)

a) Generating a level 2 multiplication equation.

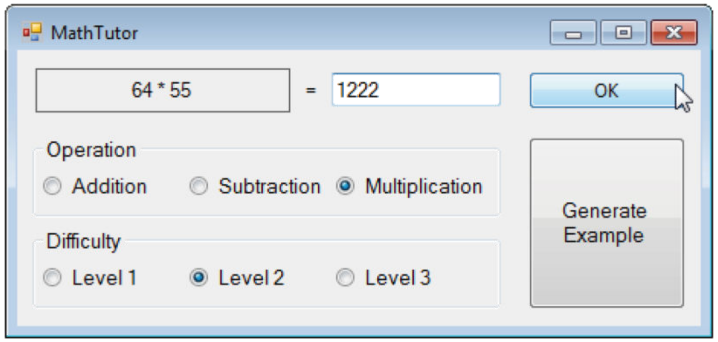


64 * 55 =

Operation: ○ Addition ○ Subtraction ● Multiplication

Difficulty: ○ Level 1 ● Level 2 ○ Level 3

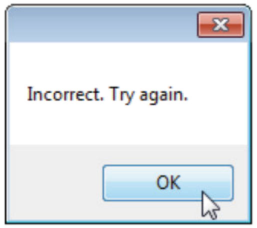
b) Answering the question incorrectly.



64 * 55 = 1222

Operation: ○ Addition ○ Subtraction ● Multiplication

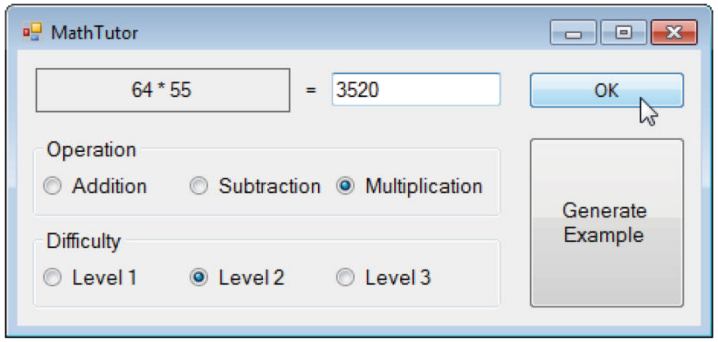
Difficulty: ○ Level 1 ● Level 2 ○ Level 3



Incorrect. Try again.

Fig. 23.31 | Math tutor using JSON version of `EquationGeneratorServiceJSON`. (Part 6 of 7.)

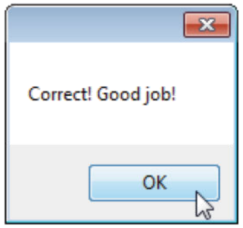
c) Answering the question correctly.



64 * 55 = 3520

Operation: Addition, Subtraction, [✓ Multiplication]

Difficulty: Level 1, [✓ Level 2], Level 3

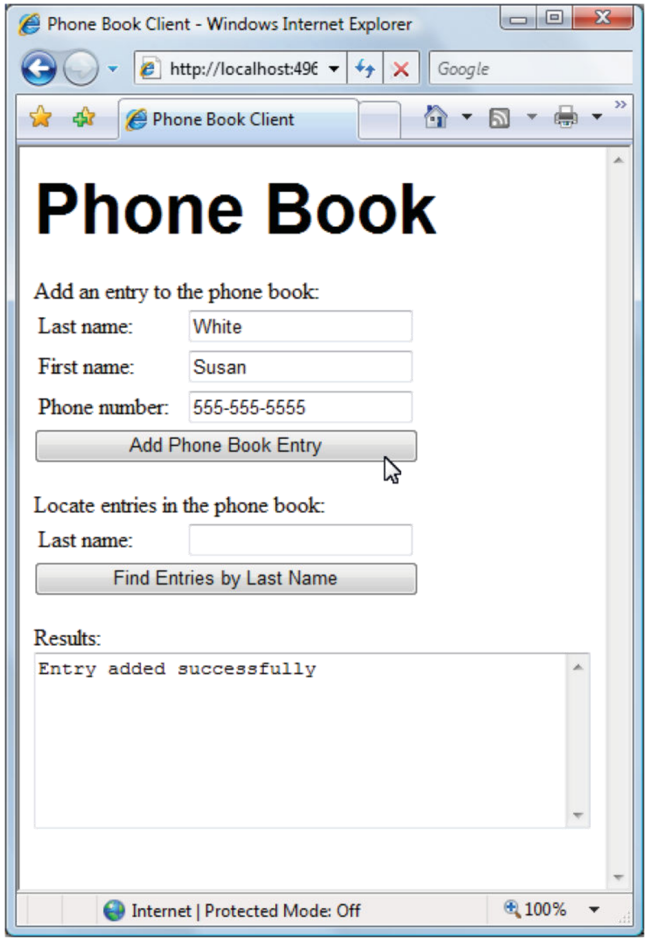


Correct! Good job!

Fig. 23.31 | Math tutor using JSON version of `EquationGeneratorServiceJSON`. (Part 7 of 7.)

```
' Fig. 23.32: Equation.vb
' Equation class representing a JSON object.
<Serializable()>
Public Class Equation
   Public Left As Integer
   Public LeftHandSide As String
   Public Operation As String
   Public Result As Integer
   Public Right As Integer
   Public RightHandSide As String
End Class ' Equation
```

Fig. 23.32 | Equation class representing a JSON object.


Phone Book Client - Windows Internet Explorer
http://localhost:49€
Google
Phone Book Client

Phone Book

Add an entry to the phone book:

Last name:	White
First name:	Susan
Phone number:	555-555-5555

Locate entries in the phone book:

Last name:	

Find Entries by Last Name

Results:

```
Entry added successfully
```

Internet | Protected Mode: Off
100%

Fig. 23.33 | Template web form for phone book client.