Andrew Brust

Crie relatórios profissionais com o Crystal Report e o Visual Studio .NET

O Crystal Reports for Visual Studio .NET oferece uma solução abrangente para criação de relatórios para desenvolvedores .NET totalmente integrada ao Visual Studio .NET IDE e ao .NET Framework. O Crystal Reports oferece suporte ao ADO.NET, XML Web Services, ASP.NET server controls e caching. Ele também é totalmente integrado ao Visual Studio .NET Server Explorer, a Toolbox e ao ambiente de design. Além disso, apresenta um modelo de programação completo e opções flexíveis para customização e implantação de relatórios. Esses recursos principais e outros mencionados aqui põem um fim ao trabalho maçante da representação de dados em seus aplicativos.

Os desenvolvedores que utilizam tecnologias Microsoft® têm um relacionamento próximo e longo com o Crystal Reports, desde o remoto surgimento do Visual Basic® 1.0. O Crystal Reports for Visual Studio .NET foi desenvolvido juntamente com engenheiros da Microsoft para integração completa com o .NET Framework e o Visual Studio .NET. Ele apresenta um designer integrado para criação de novos relatórios, a capacidade de importar qualquer relatório Crystal Report existente (ou relatórios de dados do Visual Basic 6.0) para o formato .rpt e muitos outros recursos que estarão descritos em detalhes neste artigo.

Aqui, examinaremos a criação de um aplicativo Windows® Forms, acrescentaremos um ASP.NET Web Service e uma ASP.NET Web Application. Durante as etapas, você verá alguns truques úteis de codificação para tarefas como a geração de documentos do Acrobat (PDF) a partir dos seus relatórios, a criação de relatórios parametrizados em Stored Procedures no SQL Server™, e o envio de relatórios usando as credenciais de segurança de banco de dados do usuário atual. O código disponível para download inclui os aplicativos Windows Forms, ASP.NET Web Forms e ASP.NET Web Services.

Um relatório simples

Para começar com alguma coisa simples, criarei um relatório a partir da tabela Customers (Clientes) do banco de dados SQL Server Northwind e a exibiremos em um aplicativo Windows Forms. Começarei criando um novo aplicativo para Windows no Visual Studio .NET. (Usarei o Visual Basic, mas os conceitos são compatíveis com todas as linguagens .NET.) Quando o aplicativo estiver criado, clique com o botão direito do mouse no nome do projeto no Solution Explorer, selecione Add | Add New Item nos menus de contexto. Depois, na caixa de diálogo Add New Item, selecione Crystal Report na lista de Templates, nomeie o relatório como CustomersBasic.rpt e clique em Open (veja a **Figura 1**).

Quando a janela Crystal Report Gallery for exibida, você pode aceitar as configurações padrão (Using the Report Expert e Standard) e clique em OK. Quando o Report Expert

for exibido, percorra a ramificação do OLE DB do controle do modo de exibição do treeview na guia Data. Quando a caixa de diálogo OLE DB (ADO) for exibida, selecione Microsoft OLE DB Provider for SQL Server, especifique o servidor e as informações de login do SQL Server, selecione o banco de dados Northwind e, depois, clique em Finish. De volta ao Standard Report Expert, percorra o nó do Northwind no modo de exibição do treeview até o nó Tables e, depois, clique duas vezes no nó Customers. Na guia Fields, clique no botão Add All para adicionar todos os campos da tabela Customers ao relatório. Depois, verifique se atribuiu ao relatório um título e um estilo na guia Style e clique no botão Finish.

O designer do Crystal Reports for Visual Studio .NET é ativado e o relatório apresentado. Observe que a janela Field Explorer é exibida e está ancorada às janelas Toolbox e Server Explorer. No Toolbox a paleta do Crystal Reports está disponível. Além disso, a barra de ferramentas Crystal Reports agora é apresentada na parte superior esquerda da área da barra de ferramentas do Visual Studio .NET.

Usarei esse relatório de várias maneiras ao percorrer cada exemplo no artigo. O primeiro exemplo envolve a exibição desse relatório em uma janela, sem modificações. Isso é fácil. No designer, arraste um CrystalReportViewer da paleta Windows Forms do Toolbox para o formulário padrão do projeto, defina a propriedade name como cvwMain, o Dock como Fill e a ReportSource para o meu relatório (usando a opção Browse) e, depois, execute o aplicativo. Caso eu tivesse criado o relatório usando um log in de banco de dados com uma senha em branco, o relatório seria exibido imediatamente, caso contrário, seria necessário fazer login antes de exibi-lo. Mostrarei como escrever código resumidamente para automatizar esse login, para que usuários já autenticados não tenham que fazer login uma segunda vez apenas para executar relatórios.

Relatórios fortemente tipados

Agora que já abordamos os princípios básicos, vamos passar para recursos mais sofisticados. Da mesma forma que é possível criar DataSets fortemente tipados com seus próprios métodos e propriedades, também é possível criar relatórios fortemente tipados. Na verdade, todos os relatórios criados com Crystal Reports e Visual Studio .NET resultam em uma classe de relatórios fortemente tipados como um subproduto do próprio processo de design. Para ver isso, clique no botão Show All Files no Solution Explorer, percorra o CustomersBasic.rpt e você verá que um arquivo de classe foi criado pelo Visual Studio .NET. Eu posso criar uma instância dessa classe em código e atribuir a ela a propriedade ReportSource do objeto CrystalReportViewer em meu formulário. Para fazer isso, eu acrescento a seguinte linha de código ao evento Load do formulário. (Primeiro, eu redefino a propriedade ReportSource de cvwMain como None na janela Properties).

```
cvwMain.ReportSource = New CustomersBasic()
```

Mas, da mesma maneira que o designer de componente DataSet pode ser usado com DataSets fortemente tipados,

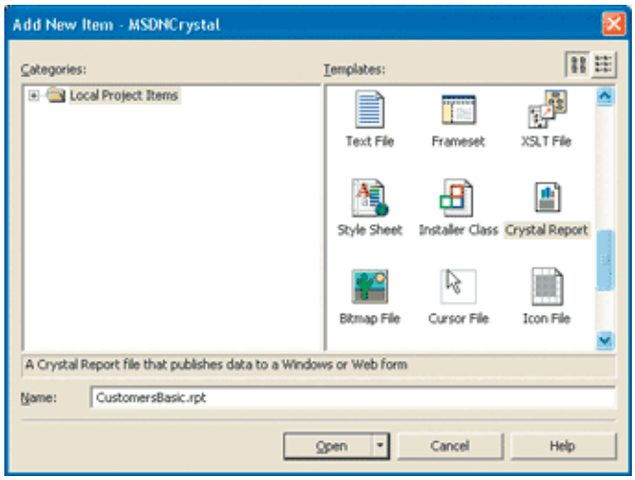

Figura 1. Criação de um relatório simples

relatórios fortemente tipados também oferecem um designer de componente. Ao clicar na paleta Components da janela Toolbox no Visual Studio .NET, você verá um objeto chamado ReportDocument. Se arrastar um deles para o seu formulário, você verá o nome da classe do seu relatório listado no dropdown Name, na caixa de diálogo Choose aReportDocument. Selecione-o e clique em OK, e terá uma instância disponível em tempo de execução da sua classe de relatório. Renomeie o objeto como cbsMain (cbs é o prefixo de CustomersBasic) e substitua a linha a seguir do código no evento Load do formulário:

```
cvwMain.ReportSource = cbsMain
```

Agora, rode o aplicativo novamente e você verá que o relatório é executado como antes. A diferença nessas duas execuções é que, em vez de fazer referência a um arquivo específico em um determinado local físico, seu ReportViewer faz referência a um objeto que poderia ser instanciado de uma classe no assembly do aplicativo, um assembly externo, ou como você verá depois, até mesmo em um XML Web Service.

Conforme mencionei, se você tiver fornecido informações de conexão que incluem uma ID de usuário com uma senha em branco na criação do relatório, ele abrirá imediatamente. Em outros casos, os relatórios do Crystal Reports for Visual Studio .NET solicitarão o login. Isso ocorre porque todas as informações de conexão, exceto a senha, são salvas com o relatório. Se você desejar fornecer uma senha em tempo de execução, ou uma definição diferente de informações de conexão da que foi fornecida na criação do relatório (essa situação é freqüente), use o código na **Listagem 1** antes de definir cvwMain.ReportSource.

Na **Listagem 1**, eu arbitrariamente defino o servidor para localhost, ID de usuário e senha como, ReportUser e msdn, respectivamente. Na maioria dos aplicativos, você provavelmente deseja usar variáveis globais ou propriedades de uma classe, ou talvez variáveis Session (de sessão) no caso de um aplicativo ASP.NET, para fornecer o nome do banco de dados e as credenciais fornecidas pelo usuário do seu aplicativo

Listagem 1. Logging Crystal Reports no Banco de Dados

```
Dim tbCurrent As CrystalDecisions.CrystalReports.Engine.Table
Dim tliCurrent As CrystalDecisions.Shared.TableLogOnInfo
For Each tbCurrent In cbsMain.Database.Tables
    tliCurrent = tbCurrent.LogOnInfo
    With tliCurrent.ConnectionInfo
        .ServerName = "localhost"
        .UserID = "ReportUser"
        .Password = "msdn"
        .DatabaseName = "Northwind"
    End With
    tbCurrent.ApplyLogOnInfo(tliCurrent)
Next tbCurrent
```

durante o login original. Observe o loop For Each no código. Meu relatório tem apenas uma tabela, o que torna o loop um tanto desnecessário (eu poderia apenas ter referenciado cbsMain.Database.Tables(0)), mas esse código funcionará com qualquer relatório Crystal Reports for Visual Studio .NET e eu prefiro manter essa flexibilidade.

O Server Explorer

Agora, vamos examinar outras maneiras de configurar um controle CrystalReportViewer. Outra opção em tempo de design é usar o Server Explorer. Devido a uma alteração de última hora no .NET Framework, esse recurso do Crystal Reports for Visual Studio .NET não é corretamente ativado pelo instalador do Visual Studio. Para corrigir isso, antes de prosseguir com as etapas deste artigo, tenha a certeza de definir as permissões na pasta Crystal Reports for Visual Studio .NET, para controle completo (Full control) do usuário ASP.NET. (A propósito, localização padrão da pasta é C:\Arquivos de Programas\Microsoft Visual Studio .NET\Crystal Reports). Faça isso na guia Security da propriedade da pasta. Para que a guia Security seja exibida, os usuários do Windows XP precisam primeiro selecionar o item de menu Tools (Ferramentas) | Folder Options (Opções da pasta) em uma janela do Explorer, clicar na guia View e limpar a opção "Use simple file sharing", que é o último item da lista de configurações Advanced. Quando essas etapas estiverem concluídas, as etapas a seguir deverão funcionar corretamente.

Sob o nó Servers, clique no nó que contém o nome da sua máquina local, depois, no nó Crystal Services abaixo dele e, por fim, no nó Server Files. Você verá que aparecem dois nós de pasta abaixo desse nó Server Files, e que abaixo de cada um deles aparecem diversos arquivos de relatório (veja a **Figura 2**). Cada um desses arquivos .rpt pode ser arrastado para o form designer, o que resultará na criação de um objeto do designer de componente ServerFileReport que aponta para o arquivo .rpt correto. A interface deste recurso com seu próprio relatório é a próxima etapa.

Parece que as pastas e arquivos de relatório exibidos nesta seção do Server Explorer estão todos localizados em um determinado diretório no disco rígido do servidor. Por padrão, essa pasta está localizada no subdiretório Samples\Reports da pasta Crystal Reports padrão mencionada anteriormente. As duas pastas mencionadas anteriormente são subdiretórios localizados aqui e os arquivos individuais .rpt estão fisicamente contidos nessas pastas.

Para fazer com que o seu relatório apareça no Server Explorer, copie-o para a pasta pai, uma das pastas filho ou para sua própria pasta criada sob a pasta pai. Por exemplo, você pode criar uma pasta chamada MSDN na pasta Samples\Reports e copiar o CustomersBasic.rpt para lá. Depois, se você clicar com o botão direito do mouse no nó Server Files em Crystal Services no Server Explorer e selecionar Refresh no menu de contexto, a pasta MSDN® deverá ser exibida, e sob ela um nó do CustomersBasic.rpt. Arraste-o para o formulário e renomeie o objeto resultante como sfrCustomersBasic. Na janela Properties, defina a propriedade ReportSource de cvwMain para apontar para esse objeto. Comente qualquer código no evento Load do formulário, execute o aplicativo e você verá o relatório ser exibido perfeitamente.

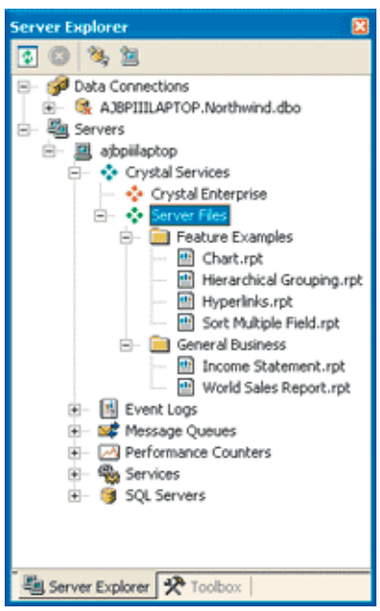

Figura 2. Server Explorer

Criando relatórios a partir de DataSets

Discutirei os recursos mais avançados do designer de componente ServerFileReport mais adiante. Por enquanto, a próxima etapa será modificar o relatório para que obtenha seus dados de um ADO.NET DataSet em vez de um banco de dados físico através de um provedor OLE DB. Em primeiro lugar, criaremos um DataSet fortemente tipado com base na tabela Customers do Northwind. A maneira mais fácil de fazer isso é adicionar um novo DataSet vazio ao projeto e nomear esse DataSet como dsNorthwind.xsd usando a caixa de diálogo Add New Item (veja a **Figura 3**).

Em seguida, abra o designer do DataSet clicando duas vezes em seu nó no Solution Explorer. Depois, no Server Explorer, arraste a tabela Customers para a superfície do design do DataSet e salve o arquivo. Em seguida, abra o relatório e clique com o botão direito do mouse na superfície do design para exibir o menu de contexto e, depois, selecione Database e Set Location. Na caixa de diálogo Set Location, expanda o nó Project Data, o nó ADO.NET DataSets, o nó do seu DataSet e selecione a tabela Customers. Por último, na combobox Current Data Source, selecione Customers e, depois, clique no botão Replace. Seu relatório procurará os metadados em seu DataSet fortemente tipado e será executado quando vinculado a uma instância dele. Clique no botão Close para continuar.

Agora que modifiquei o relatório para ser executado com uma instância do dsNorthwind, preciso criar uma em meu formulário e definir o ReportSource do CrystalReportViewer para ele. Para fazer

isso, arraste a tabela Customers do Server Explorer para a superfície do designer do formulário. Isso acrescentará objetos SQLConnection e SQLDataAdapter ao formulário que aponta para o banco de dados Northwind e sua tabela Customers, respectivamente. Renomeie esses objetos como scnNorthwind e sdaCustomers, modifique a propriedade ConnectionString do scnNorthwind para incluir uma ID de usuário e senha válidas e, depois, clique com o botão direito do mouse em sdaCustomers e selecione Generate Dataset. Na seção "Choose a dataset", selecione o botão de opções Existing e o nome de classe do seu DataSet fortemente tipado na combobox adjacente (de qualquer maneira, essas seleções provavelmente serão as configurações padrão). A **Figura 4** ilustra isso claramente.

Clique no botão OK, renomeie o novo objeto DataSet como "ndsMain" e altere o código de evento Load para o seguinte:

```
sdaCustomers.Fill(ndsMain)
cbsMain.SetDataSource(ndsMain)
cvwMain.ReportSource = cbsMain
```

Observe que esse código preenche primeiro a ndsMain, uma instância do DataSet fortemente tipada do dsNorthwind que eu criei. Em seguida, o código define a propriedade DataSource da cbsMain, a instância da classe de relatório fortemente tipada, CustomersBasic, como ndsMain. Por último, como nos exemplos anteriores, aponto o ReportSource do cvwMain, meu CrystalReportViewer para cbsMain.

Inicie o aplicativo e você verá o relatório ser executado e exibido exatamente como foi feito antes. A diferença neste exemplo é que os dados não vêm diretamente de um banco de dados, mas de um objeto DataSet desconectado. No meu caso, o DataSet foi preenchido por uma consulta de seleção direta com a tabela Customers. (Examine a propriedade CommandText da propriedade do objeto SelectCommand de sdaCustomer para ver o texto da consulta real.) Eu poderia ter criado esse DataSet a partir de qualquer fonte ou fontes de dados que tivesse escolhido, desde que o dataset construído fosse compatível com a estrutura do ndsNorthwind.

Criação de relatórios a partir de Stored Procedures

Para ilustrar esse último ponto, criaremos uma stored procedure parametrizada para ser executada com a tabela Customers chamada spCustomers que usa uma string parametrizada a ser usada em uma cláusula WHERE para limitar o número de registros retornados. Depois, no código, eu posso criar um DataSet com essa stored procedure e vincular o relatório a esse dataset. Esse é o código do stored procedure.

```
CREATE PROCEDURE dbo.spCustomers
(
    @CustPattern nVarChar(40)
)
AS
select * from Customers Where CompanyName Like @CustPattern + '%'
RETURN
```

Você pode criar esse stored procedure através de vários meios com os quais está acostumado; fazer isso através do Visual Studio .NET é bem conveniente, conforme mostrado na **Figura 5**. Para criar um relatório a partir desta stored procedure (usando o valor arbitrário de "A" para @CustPattern), eu modifico o código do evento Load do formulário, como você pode ver no exemplo mostrado na **Listagem 2**.

Nesse código, não estou usando nenhum dos objetos que criei em design time exceto o scnNorthwind. E o mais importante, não estou usando uma instância do meu DataSet fortemente tipado, dsNorthwind, como a fonte de dados do relatório, mas estou usando um objeto ADO.NET DataSet não-tipado chamado dsReport. Isso é extremamente importante — mesmo que eu tenha criado o relatório com um DataSet fortemente tipado, em run time, eu defino sua fonte de dados para um objeto DataSet genérico, não-tipado, cuja estrutura

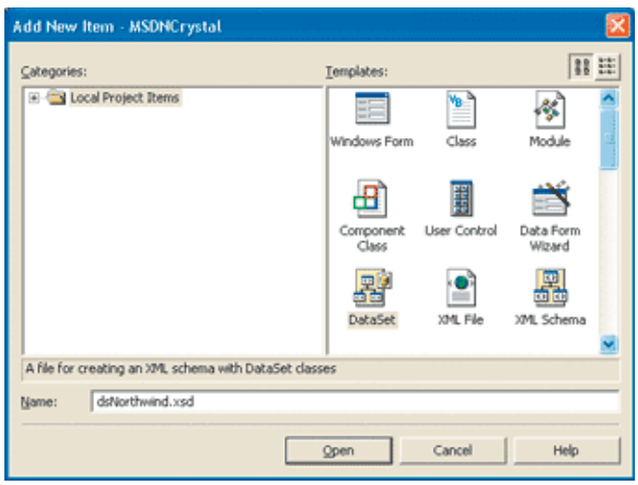

Figura 3. Adicionando um DataSet

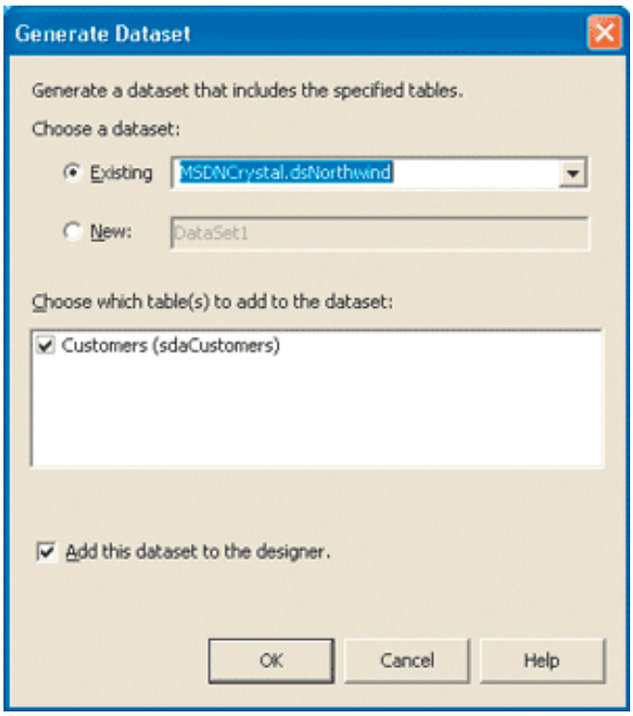

Figura 4. Geração do DataSet

Listagem 2. Evento OnLoad Modificado

```
Private Sub frmViewReport_Load(ByVal sender As Object,_
ByVal e As System.EventArgs) Handles MyBase.Load

    Dim scmCustomersSP As New _
        SqlClient.SqlCommand("spCustomers", scnNorthwind)
    scmCustomersSP.CommandType = CommandType.StoredProcedure
    scmCustomersSP.Parameters.Add("@CustPattern", "A")

    Dim sdaCustomersSP As New _
        SqlClient.SqlDataAdapter(scmCustomersSP)
    Dim dsReport As New DataSet()
    sdaCustomersSP.Fill(dsReport, "Customers")

    cbsMain.SetDataSource(dsReport)
    cvwMain.ReportSource = cbsMain
End Sub
```

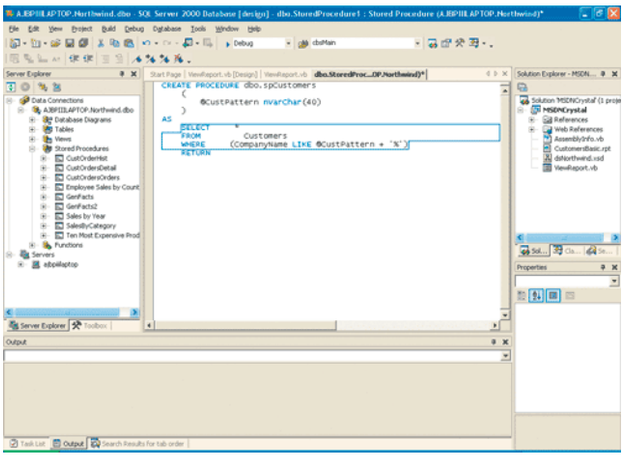

Figura 5. Criando a Stored Procedure

apenas corresponde àquela do dsNorthwind. Quando eu chamo o método Fill do objeto DataAdapter, faço isso usando a versão sobrecarregada que aceita um nome de tabela, e forneço Customers como nome dessa tabela para garantir que dsReport corresponda estruturalmente a dsNorthwind. Se eu não fizer isso, o relatório teria sido apresentado sem dados.

Também é possível criar um relatório diretamente com uma stored procedure, da mesma maneira que meu primeiro exemplo foi criado diretamente com uma tabela. Definir o valor do parâmetro no código requer algum comando de controle dos objetos, propriedades e métodos nos diversos namespaces do Crystal Reports for Visual Studio .NET. Vejamos.

Primeiro, você precisa configurar o Crystal Reports for Visual Studio .NET para criar stored procedures selecionáveis como fontes de dados para relatórios. Para fazer isso, clique com o botão direito do mouse em uma área em branco da superfície do designer do relatório e escolha Designer | Default Settings nos menus de contexto. Na caixa de diálogo Default Settings, clique na guia Database, verifique se a caixa de seleção Stored Procedures na seção Show está marcada e, depois, clique em OK.

A seguir, clique com o botão direito do mouse na superfície do designer e escolha Database | Set Location nos menus de contexto. Na treeview Replace With, percorra o nó OLE DB (ADO) e mais uma vez, na caixa de diálogo OLE DB (ADO), forneça todas as informações necessárias para conexão à sua cópia do banco de dados SQL Server Northwind. Depois, de volta à caixa de diálogo Set Location, percorra o nó Northwind e, em seguida, o nó Stored Procedures e selecione o nó spCustomers;1. Na combobox Current Data Source, selecione Customers, clique no botão Replace e, por último, no botão Close.

Você melhorou este relatório — originalmente criado com a tabela Customers no Northwind Database e, depois, convertido para usar um dataset fortemente tipado como sua fonte de dados — para usar a stored procedure spCustomers para obter seus dados. Como essa é uma stored procedure parametrizada, o Crystal Reports for Visual Studio .NET acrescenta automaticamente ao seu relatório um campo de parâmetros (Parameter Field), com o mesmo nome que o parâmetro da stored procedure. Para vê-lo, procure na janela Field Explorer e analise o nó Parameter Fields para revelar o campo de parâmetros @CustPattern logo abaixo dele (a **Figura 6** mostra a localização).

Agora, você pode definir novamente o cvwMain.ReportSource, no momento da criação ou no código, para apontar para o arquivo rpt, ou para cbsMain, a instância do designer de componente ReportDocument. Se você executar o aplicativo, verá que o Crystal Reports for Visual Studio .NET automaticamente invoca uma caixa de diálogo que solicita um valor para o parâmetro @CustPattern. Digite qualquer caracter que desejar e ele retornará apenas os clientes cujo valor de campo CompanyName inicia-se com o caracter digitado.

Ao mesmo tempo que é legal o fato de o Crystal Reports for Visual Studio .NET poder solicitar valores de parâmetros e executar o relatório sem ter que escrever um código, é melhor para os aplicativos solicitar e passar esses valores para o relatório programaticamente, a **Listagem 3** mostra o código necessário para atribuição programática de valores de parâmetro através do modelo de objeto do Crystal Reports. Nesse caso, mais uma vez, atribuindo o valor arbitrariamente de "A" como parâmetro de pdvCustPattern.

Faz-se necessária alguma explicação desse código. A parte confusa é que apesar de ser possível atribuir apenas um único valor a parâmetros T-SQL, é possível atribuir uma coleção de valores aos campos de parâmetro do Crystal Reports for Visual Studio .NET. Portanto, tenho que prosseguir com essa etapa e criar uma coleção que tenha apenas um membro, atribuindo "A" como valor. Conseqüentemente, o código cria pvCustPattern, uma instância do objeto CrystalDecisions.Shared.ParameterValues que recebe vários valores, e pdvCustPattern, uma instância da classe CrystalDecisions.Shared.ParameterDiscreteValue que recebe um valor único. A string "A" é atribuída à propriedade Value de pdvCustPattern e o objeto é carregado no pvCustPattern através do método Add. Por último, pvCustPattern é atribuído ao para-

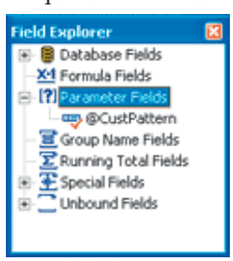

Figura 6. Parâmetros

Listagem 3. Atribuindo Valores

```
Private Sub frmViewReport_Load(ByVal sender As Object,_
ByVal e As System.EventArgs) Handles MyBase.Load

    Dim scmCustomersSP As New _
        SqlClient.SqlCommand("spCustomers", scnNorthwind)
    scmCustomersSP.CommandType = CommandType.StoredProcedure
    scmCustomersSP.Parameters.Add("@CustPattern", "A")

    Dim sdaCustomersSP As New _
        SqlClient.SqlDataAdapter(scmCustomersSP)
    Dim dsReport As New DataSet()
    sdaCustomersSP.Fill(dsReport, "Customers")

    cbsMain.SetDataSource(dsReport)
    cvwMain.ReportSource = cbsMain
End Sub
```

meter field por meio do método ApplyCurrentValues e a instância de CustomersBasic é atribuída à propriedade ReportSource do visualizador de relatórios. Apesar do código parecer um pouco enrolado, a funcionalidade produzida o torna vantajoso.

XML Report Web Services

Investigaremos o suporte para XML Web Services no Crystal Reports for Visual Studio .NET. A boa notícia é que, apesar dos recursos serem sofisticados, usufruir deles é fácil. Os relatórios podem ser publicados como Web Services e, depois, usados como o ReportSource para um controle CrystalReportViewer.

Para criar um XML Report Web Service a partir do CustomersBasic.rpt, abra o Visual Studio .NET e crie um novo projeto ASP.NET Web Service denominado MSDNCrystalWebService. Exclua o arquivo Service1.asmx que é adicionado ao projeto por padrão, e adicione o CustomersBasic.rpt ao projeto. A seguir, clique com o botão direito do mouse em CustomersBasic.rpt, no Solution Explorer, e escolha Publish as Web Service no menu de contexto (veja a **Figura 7**). Ao fazer isso você está adicionando um arquivo Web Service, CustomersBasicService.asmx, ao projeto. Agora, compile o projeto. Depois, de volta ao aplicativo Windows Forms, defina cvwMain.ReportSource para a URL do Report Web Service, como na seguinte linha do código:

```
cvwMain.ReportSource = "http://localhost/MSDNCrystalWebService/" _
  & "CustomersBasicService.asmx"
```

Como alternativa, você pode adicionar uma referência da Web desse arquivo asmx ao projeto e, depois, apontar cvwMain.ReportSource para uma instância da classe Web Service.

```
cvwMain.ReportSource = New localhost.CustomersBasicService()
```

Como pode ser tão fácil criar uma interface XML Web Services para Crystal Reports? Examine bem o código no arquivo da classe sob o arquivo CustomersBasic.rpt (clique no botão Show All Files no Solution Explorer, se necessário, para exibir o arquivo). Existem duas classes no código, neste arquivo: CustomersBasic e CachedCustomersBasic.

A CustomerBasic é a classe necessária para implementar o relatório fortemente tipado. Se você examinar as propriedades, verá que ele expõe antes de tudo o relatório e suas diversas seções, herdando de ReportClass e empacotando (wrapping) diversos membros da coleção ReportDefinition.Sections. A CachedCustomersBasic, por sua vez, implementa a interface ICachedReports para criar um relatório armazenado em cache, e em seu método CreateReport, ela cria uma instância da CustomersBasic (a primeira classe). Usando uma interface semelhante a esta que utiliza WebMethods, é possível expor o relatório como um XML Web Service.

Agora, examine o código no arquivo da classe sob CustomersBasicService.asmx. Ele também contém duas classes, apesar de uma estar aninhada na outra desta vez. A CustomersBasicService é herdada do ReportServiceBase, que por sua vez expõem os WebMethods necessários para publicar o relatório como um XML Web Service. A CustomersBasicService contém a classe CachedWebCustomersBasic, que tem essencialmente a mesma implementação que a classe CachedCustomersBasic recém-comentada. Uma diferença pode ser encontrada em seu construtor:

```
Public Sub New(ByVal webServiceParam As CustomersBasicService)
    Me.webService = webServiceParam
End Sub
```

O construtor é chamado a partir do próprio CustomersBasicService, para definir a propriedade ReportSource do relatório armazenado em cache.

```
Public Sub New()
    Me.ReportSource = New CachedWebCustomersBasic(Me)
End Sub
```

Recapitulando: A ReportClass do Crystal Report torna possíveis relatórios fortemente tipados. A interface ICachedReports torna possível criar instâncias armazenadas em cache da ReportClass. E a ReportServiceBase cria efetivamente uma interface de Web Services para relatórios fortemente tipados, que é compatível com a interface IcachedReports.

Mais uma coisa. Lembra-se do componente ServerFileReport que eu criei arrastando e soltando do Server Explorer? Ele também utiliza a tecnologia Web Services. Se você retornar ao projeto Windows Form, observará que sfrCustomersBasic tem uma propriedade WebServiceURL, que aponta para um arquivo genérico .asmx. O URL se parece com localhost/crystalreportwebformviewer/ServerFileReportService.asmx. Apesar dessa URL ser genérica, a ReportPathProperty de sfrCustomersBasic, que aponta para \MSDN\CustomersBasic.rpt, especifica o nome e o local relativo do relatório que eu realmente quero executar.

Agora pode parecer estranho que o componente ServerFileReport solicite que você execute o relatório por uma conexão HTTP/SOAP para sua máquina local, mas isso é, na verdade, o que ele está fazendo. Entretanto, isso significa que você poderia facilmente clicar com o botão direito do mouse no nó Servers no Server Explorer, escolher Add Server, conectar-se a qualquer outro servidor da sua LAN/WAN e executar relatórios Crystal Reports for Visual Studio .NET existentes naquela máquina remota. Isso facilita muito

a implantação de relatórios por toda a empresa: basta instalar o Visual Studio .NET em um servidor, copiar os arquivos rpt para a pasta apropriada, e eles serão disponibilizados imediatamente na organização inteira (via SOAP) sem a necessidade de nenhuma codificação especial. Mesmo o processo relativamente simples de clicar com o botão direito do mouse em um relatório de um projeto ASP .NET e selecionar "Publish as Web Service" não é necessário. Tudo que você precisa fazer é copiar o arquivo para a pasta correta.

A desvantagem de usar o recurso arrastar e soltar no Server Explorer é que podem ser pesquisados apenas objetos em servidores residentes na própria LAN/WAN do desenvolvedor, e não aqueles protegidos por um firewall. Mas existe uma saída: contanto que a URL e o ReportPath de determinado relatório sejam conhecidos, e o servidor (com ou sem firewall) esteja conectado à Internet, você poderá criar uma nova instância do CrystalDecisions.ReportSource.ServerFileReport e definir essas propriedades dele em código.

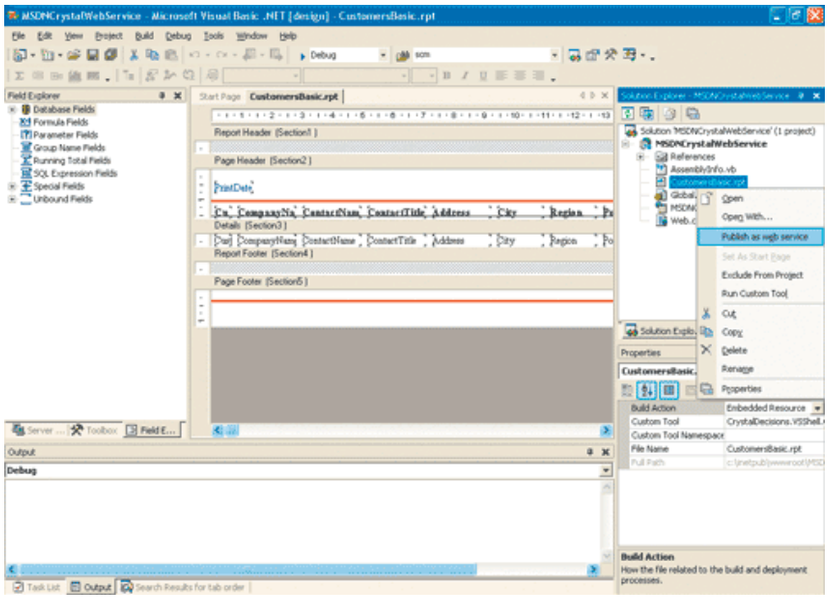

Figura 7. MTS

O nó Crystal Enterprise sob Crystal Services no Server Explorer pode ser usado para pesquisar relatórios que residem em qualquer Crystal Enterprise Server na LAN/WAN, e usá-los como ReportSources válidos para um controle do CrystalReportViewer, usando, outra vez, o SOAP. Agora os relatórios Crystal Enterprise estão disponíveis para qualquer desenvolvedor do Crystal Reports for Visual Studio .NET conectado à Internet munido com a URL e a ID de objeto corretas.

Construção de relatórios no ASP.NET

Até agora, tudo que eu fiz no cliente tem sido em um aplicativo Windows Forms, mas você pode querer apresentar relatórios em aplicativos ASP.NET. Felizmente, o Crystal Reports for Visual Studio .NET fornece suporte para ASP.NET. Agora eu vou mostrar como apresentar relatórios usando o controle CrystalReportViewer para ASP.NET Web Forms, como documentos Acrobat (PDF) e HTML. Também examinarei o armazenamento em cache e a utilização de dados em cache.

Posso abordar os princípios básicos bem rapidamente porque o Web Forms CrystalReportViewer funciona de maneira bem semelhante em relação ao Windows Forms. Para começar, crie um novo ASP.NET Web Application no Visual Studio .NET, e utilize o recurso Add Existing Item para adicionar o CustomersBasic.rpt ao projeto. Defina o data location do relatório de volta à tabela Customers, e para fins de teste, utilize um login com uma senha em branco. Depois, renomeie o WebForm default como ViewReport.aspx, e adicione um controle Web Forms CrystalReportViewer a ele, definindo sua propriedade ID como cvwMain. Como o controle Web Forms CrystalReportViewer não tem uma propriedade ReportSource no momento da criação (design time), você terá que definir sua propriedade ReportSource no evento Page_Load do ViewReport.aspx, da seguinte maneira:

```
cvwMain.ReportSource = Server.MapPath("CustomersBasic.rpt")
```

Execute o aplicativo e você verá que ele funciona bem. O relatório é apresentado como HTML puro, mas a pesquisa, a navegação, o zoom e outros recursos de controle do visualizador estão suportados (exceto a impressão), apesar de ser feito através do uso de postbacks. Você pode tentar todos os outros modos de carregamento de um relatório que eu já discuti: criação de relatórios a partir de um DataSet fortemente tipado ou não-tipado, de uma stored procedure, usando um ReportDocument ou um designer de componente ServerFileReport e outros. O código que eu usei anteriormente para configuração das informações de segurança do relatório e dos valores de parâmetro, e para manipulação de objetos ADO.NET, pode ser copiado para o evento Page_Load do ViewReport.aspx.

ReportDocuments armazenados em cache

Exploraremos os relatórios armazenados em cache do Crystal Reports for Visual Studio .NET. Arraste um componente ReportDocument para a superfície do designer de ViewReport.aspx, e selecione ProjectName.CustomersBasic na combobox "Name" na caixa de diálogo "Choose a ReportDocument", mas não clique no botão OK ainda. Observe a checkbox "Generate cached strongly typed report" (veja a **Figura 8**).

Marque este checkbox. Clique em OK e renomeie o componente gerado para ccbMain. Depois, vincule o controle do visualizador (viewer control) a esse objeto no evento Page_Load, com esta linha de código:

```
cvwMain.ReportSource = ccbMain
```

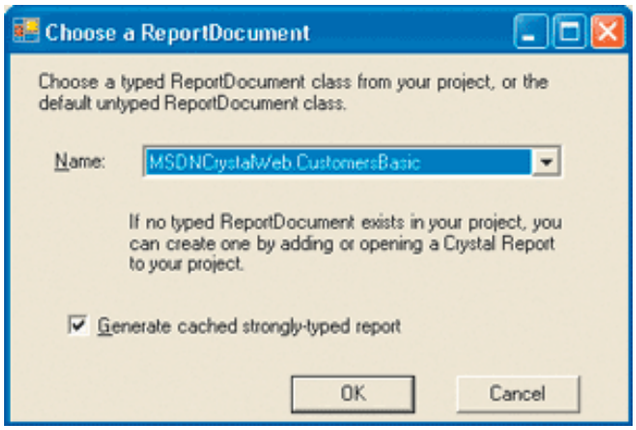

Figura 8. Recurso de relatórios armazenados em cache

Em seguida, arraste os controles Data Time e Print Time para a seção do cabeçalho da página (page header) do relatório. Esses controles podem ser encontrados na janela "Field Explorer", sob o nó "Special Fields". Agora execute o aplicativo e exiba o relatório. Na primeira vez em que você executá-lo, Data Time e Print Time devem ser os mesmos (ou muito próximos). Mas, se você atualizar a página, verá que Print Time continua a ser atualizado e Data Time permanece constante. Agora, copie a URL atual do navegador para a área de transferência e feche o navegador. Abra uma nova janela do navegador e cole a URL para exibir o relatório em uma nova sessão (sem ter que recriar o assembly no Visual Studio .NET). Você verá que a data e hora original continua a ser exibido para Data Time.

Agora, feche o navegador e volte ao Visual Studio .NET. Mais uma vez, arraste um componente ReportDocument para a superfície do designer de ViewReport.aspx, e selecione ProjectName.CustomersBasic, verificando se a caixa de seleção (checkbox) "Generate cached strongly typed report" está desmarcada. Clique em OK e renomeie o componente gerado para "cbsMain". Após concluir essa tarefa, vincule o controle do visualizador (viewer control) a esse objeto no evento Page_Load, fazendo uma pequena alteração no código para refletir o nome do novo componente, desta maneira:

```
cvwMain.ReportSource = cbsMain
```

Execute o aplicativo novamente para exibir o relatório e atualizar a página algumas vezes. Você verá que o controle Data Time é alterado a cada atualização. Isso demonstra a diferença básica entre um relatório armazenado em cache e outro não-armazenado: o armazenamento em cache evita retornos desnecessários ao banco de dados, quando você já tem os dados necessários.

Apesar de ser possível usar as propriedades DataDefinition ou Database de cbsMain (uma instância do meu ReportDocument não-armazenado em cache) para definir valores de parâmetros ou propriedades de logon de tabela, você observará que ccbMain (uma instância do ReportDocument armazenado em cache) não tem essas propriedades. Então, como você usa relatórios armazenados em cache com tabelas de segurança e stored procedures parametrizadas? A resposta reside no código por trás (code behind) do relatório fortemente tipado. Mais uma vez, clique no botão Show All Files no Solution Explorer e percorra CustomersBasic.rpt para exibir o arquivo de código por trás dele. Você já esteve aqui antes, mas desta vez examine o código da classe CachedCustomersBasic neste arquivo.

Observe que o código do método CreateReport cria, na verdade, uma nova instância da classe não-armazenada em cache, CustomersBasic. Isto se deve ao fato de que CachedCustomersBasic realmente envolve (wraps) CustomersBasic. Desta forma, eu posso escrever um código personalizado neste método público para acessar as propriedades Database e DataDefinition da instância criada de CustomersBasic, e fazer interface com objetos de bancos de dados seguros e stored procedures parametrizadas. Se eu adotar essa abordagem, quando a nova instância do relatório armazenado em cache for criada, ela então criará uma instância apropriadamente configurada do relatório fortemente tipado não-armazenado em cache.

Agora, percorra mais adiante e você verá o código, com a maior parte comentada, de outro método importante, GetCustomizedCacheKey. Esse método gera uma chave exclusiva que determina se uma cópia de um relatório armazenado em cache poderá ou não ser utilizada para atender uma solicitação. Basicamente, se a chave corresponder àquela do relatório armazenado em cache, a cópia armazenada em cache será utilizada. Se a chave não corresponder, os dados serão recuperados de novo.

Se você descomentar o código, uma função especial chamada BuildCompleteCacheKey gerará uma chave que será exclusiva para um determinado relatório. Não comentar o código, mas deixá-lo sem modificação, é funcionalmente equivalente a deixar o código comentado. Se a chave não for modificada, o Crystal Reports implicitamente utilizará o algoritmo BuildCompleteCacheKey para gerar uma chave para você. Para diversos casos, isso basta. Mas o que aconteceria se você desejasse modificar a chave ligeiramente para seus próprios objetivos? Suponha que você soubesse, por exemplo, que os dados mudaram significativamente, em determinado momento. Você poderia anexar sua própria string de caracteres ao final da chave do cache para ter certeza de que ela mudou naquele momento, ou depois. Veja a seguir um exemplo trivial: descomente o código em GetCustomizedCacheKey e insira a seguinte linha antes da instrução Return:

```
key &= DatePart(DateInterval.Minute, Now())
```

Isso assegurará que sempre que a hora atual mudar para um novo minuto, será gerada uma nova chave do cache. No evento Page_Load do ViewReport.aspx, altere cvwMain.ReportSource de volta para apontar para ccbMain, execute o aplicativo e atualize a página diversas vezes. Você deverá ver o Data Time mudar apenas quando a hora atual mudar; isto é, no ponto exato no tempo quando o lugar dos minutos é incrementado. Uma técnica semelhante pode ser usada para acrescentar valores de parâmetro e outras es-

Listagem 4. Implementando ICachedReport no Component Class

```
Imports CrystalDecisions.CrystalReports.Engine
Imports CrystalDecisions.ReportSource
Imports CrystalDecisions.Shared
Imports System
Imports System.ComponentModel

Public Class cMSDNCachedReport
    Inherits Component
    Implements ICachedReport

    'Required by the Component Designer
    Private components As System.ComponentModel.Container

    'NOTE: The following procedure is required by
    'the Component Designer
    'It can be modified using the Component Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        components = New System.ComponentModel.Container()
    End Sub

    Public ASPRequest As Web.HttpRequest

    Public Sub New()
        MyBase.New()
    End Sub

    Public Overridable Property IsCacheable() As [Boolean] _
      Implements _
      CrystalDecisions.ReportSource.ICachedReport.IsCacheable
        Get
            Return True
        End Get
        Set(ByVal Value As [Boolean])
            '
        End Set
    End Property

    Public Overridable Property ShareDBLogonInfo() As [Boolean] _
      Implements _
      CrystalDecisions.ReportSource.ICachedReport.ShareDBLogonInfo
        Get
            Return False
        End Get
        Set(ByVal Value As [Boolean])
            '
        End Set
    End Property

    Public Overridable Property CacheTimeOut() As TimeSpan _
      Implements _
      CrystalDecisions.ReportSource.ICachedReport.CacheTimeOut
        Get
            Return CachedReportConstants.DEFAULT_TIMEOUT
        End Get
        Set(ByVal Value As TimeSpan)
            '
        End Set
    End Property

    Public Overridable Function CreateReport() As  _
      CrystalDecisions.CrystalReports.Engine.ReportDocument
      Implements _
      CrystalDecisions.ReportSource.ICachedReport.CreateReport
        Dim rpt As CustomersBasic = New CustomersBasic()
        Dim tbCurrent As _
          CrystalDecisions.CrystalReports.Engine.Table
        Dim pvCustPattern As New _
          CrystalDecisions.Shared.ParameterValues()
        Dim pdvCustPattern As New _
          CrystalDecisions.Shared.ParameterDiscreteValue()

        pdvCustPattern.Value = _
          IIf(ASPRequest("ParamValue") Is Nothing, _
          "", ASPRequest("ParamValue"))
        pvCustPattern.Add(pdvCustPattern)
        rpt.DataDefinition.ParameterFields _
          ("@CustPattern").ApplyCurrentValues(pvCustPattern)

        For Each tbCurrent In rpt.Database.Tables
            Dim oLIInfo As CrystalDecisions.Shared.TableLogOnInfo
            oLIInfo = tbCurrent.LogOnInfo
            With oLIInfo.ConnectionInfo
                .ServerName = "localhost"
                .UserID = "ReportUser"
                .Password = "msdn"
                .DatabaseName = "Northwind"
            End With
            tbCurrent.ApplyLogOnInfo(oLIInfo)
        Next tbCurrent

        rpt.Site = Me.Site
        Return rpt

    End Function

    Public Overridable _
      Function GetCustomizedCacheKey(ByVal request _
      As RequestContext) As [String] Implements _
      CrystalDecisions.ReportSource.ICachedReport.
GetCustomizedCacheKey
        Dim key As [String] = Nothing
        ' Este código gera uma chave padrão para o
        ' caching report no ASP.NET Cache.
        ' Sinta-se livre para alterar este código
        ' conforme a sua necessidade
        ' Retornando key == null provoca uma chave
        ' default cache a ser gerada

        key = RequestContext.BuildCompleteCacheKey( _
            request, _
            Nothing, _
            Me.GetType(), _
            Me.ShareDBLogonInfo)
        key &= IIf(ASPRequest("ParamValue") Is Nothing, _
          "", ASPRequest("ParamValue"))
        Return key
    End Function
End Class
```

pecificações relativas a dados à chave do cache, para evitar que execuções distintas do mesmo relatório sejam manipuladas pelos mesmos dados armazenados em cache. Em vez de meramente acrescentar algo à chave do cache padrão, você poderia criar sua própria chave inteiramente do zero. Contudo, eu não recomendaria essa estratégia, já que isso é reinventar a roda.

Ao mesmo tempo que essas alterações na classe de relatório fortemente tipado são legais, fica claro que você está enfrentando algum perigo aqui. Se você atualizar o relatório mesmo assim, o Visual Studio .NET tentará sobrescrever o arquivo da classe do relatório; se você não tiver cuidado, suas alterações poderão se perder. Uma maneira de evitar isso é criar sua própria classe que implementa ICachedReport, que basicamente contém o código da classe CachedCustomersBasic. Em seguida, no evento Page_Load do ViewReport.aspx, você poderá criar uma instância dessa classe e definir cvwMain.ReportSource para apontar para ela. A **Listagem 4** tem o código da classe (incluindo o código que utiliza o objeto ASP.NET Request para definir o valor de campo de parâmetros [parameter field] do relatório e para acrescentar esse valor à chave do cache).

E o código a seguir mostrará o código de evento Page_Load necessário para usá-lo.

```
Private Sub Page_Load(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles MyBase.Load

    Dim mcrMain As New cMSDNCachedReport()
    mcrMain.ASPRequest = Request
    cvwMain.ReportSource = mcrMain
End Sub
```

Exportação para PDF

Como o controle Web Forms CrystalReportViewer fornece relatórios como HTML puro, não significa que não exista um problema. Ao mesmo tempo em que possibilita a compatibilidade máxima entre Crystal Reports for Visual Studio .NET e navegadores cliente, o HTML não permite uma paginação precisa na saída impressa.

Uma maneira de contornar problemas de impressão no lado do cliente é exportar seu relatório para o formato Adobe Acrobat (PDF). É possível, e razoavelmente simples, escrever o código ASP.NET que exporta programaticamente um relatório para PDF e, depois, redirecionar o navegador para aquele mesmo arquivo. Desta forma, o usuário pode solicitar um relatório através de um evento de postback, que será imediatamente roteado para uma versão em PDF do relatório, que pode ser impressa. A **Listagem 5** mostra o código necessário para fazer isso. Para maior simplicidade, eu uso o objeto cbsMain de ReportDocument não armazenado em cache para gerar o relatório.

Listagem 5. Exportando o relatório para PDF

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As _
  System.EventArgs) Handles MyBase.Load

  Dim dfdoCustomers As New _
    CrystalDecisions.Shared.DiskFileDestinationOptions()
  Dim szFileName As String = "c:\windows\temp\myreport2.pdf"
  dfdoCustomers.DiskFileName = szFileName
  With cbsMain
      .ExportOptions.ExportDestinationType = _
        CrystalDecisions.Shared.ExportDestinationType.DiskFile
      .ExportOptions.ExportFormatType = _
        CrystalDecisions.Shared.ExportFormatType.PortableDocFormat
      .ExportOptions.DestinationOptions = dfdoCustomers
      .Export()
  End With
  Response.Redirect(szFileName)

End Sub
```

Se você decidir usar PDFs para apresentação do relatório, deverá estar ciente das deficiências dessa abordagem. Primeiro, para acomodar vários usuários, os arquivos PDF que você gerar deverão ter nomes exclusivos; a **Listagem 5** utiliza uma seqüência de caracteres constante e literal e serve apenas para demonstrar a técnica, isoladamente. Além disso, os arquivos PDF devem ser limpos em algum momento e, dependendo do volume de relatórios feitos no seu aplicativo, todas as I/O de arquivo relativas à geração de PDF e limpeza poderiam afetar gravemente a escalabilidade se não fossem gerenciadas e configuradas de maneira adequada. Se você estiver criando aplicativos ASP.NET usando Crystal Reports for Visual Studio .NET, reserve alguns minutos para mapear seu visualizador no lado do cliente e para a estratégia de impressão. Cada opção tem seu próprio conjunto de prós e contras significativos.

Conclusão

Eu forneci aqui uma visão bem detalhada dos recursos voltados para o desenvolvedor do Crystal Reports for Visual Studio .NET, e ainda resta muito mais a ser investigado. No mínimo, esse material deverá dar a você uma visão da riqueza desses recursos e de como eles são bem integrados ao .NET Framework e ao Visual Studio .NET.

Download de códigos em: www.neoficio.com.br/msdn

Andrew J. Brust é presidente da Progressive Systems Consulting Inc., baseada em New York City, empresa especializada em desenvolver aplicações customizadas com .NET e outras tecnologias Microsoft. Andrew é MSDN Regional Director em New York City e Vice Presidente do New York Software Industry Association.