使用Windows NT集成安全性

Windows NT的优势在于能够提供处理安全问题的方式。 当用户登录时，系统会根据系统管理员建立的权限管理制度自动为用户分配权限。 然后，每个应用程序或服务都可查询Windows NT来了解他能访问的资源。
流行的数据库提供器都会将该安全器作为一种可选的安全管理选项。
在这种情形下，你只需要确定你想与哪些用户分享你的数据，并为他们分配适当的读写权限。
这种情况下，在报告定义文件中的ConnectionString不需要设置任何密码。 授权用户就能看到数据，而未授权的用户就无法读取。

使用用户提供的密码创建连接字符串（ConnectionString）

!MISSING PHRASE 'Show All'!
!MISSING PHRASE 'Hide All'!
使用用户提供的密码创建连接字符串是一种非常简单的保护数据的方法。
例如在渲染报表之前（或当控件提示"无法连接"的错误时），可以提示用户输入密码，并把该密码插入到连接字符串：
Visual Basic

```
Visual Basic
```

```
Dim strConn strConn = "Provider=Microsoft.Jet.OLEDB.4.0;" & _ "Data Source=C:\SecureData\People.mdb;" & _
"Password=THEPASSWORD;"
' get password from the user Dim strPwd$
strPwd = InputBox("Please enter your password:") If Len(strPwd) = 0 Then Exit Sub

' build new connection string and assign it to the control strConn = Replace(strConn, "THEPASSWORD", strPwd)
vsr.DataSource.ConnectionString = strConn
```

C#

```
C#
```

```

// build connection string with placeholder for password string strConn = "Provider=Microsoft.Jet.OLEDB.4.0;" + "Data
Source=C:\SecureData\People.mdb;" +
"Password=THEPASSWORD;";
// get password from the user string strPwd = InputBox("Please enter your password:"); if (strPwd.Length == 0) return;
// build new connection string and assign it to the control strConn = Replace(strConn, "THEPASSWORD", strPwd);
c1r.DataSource.ConnectionString = strConn;
```

创建定义应用程序的别名

!MISSING PHRASE 'Show All'!
!MISSING PHRASE 'Hide All'!
另一种可能的情况是，你要允许某些用户看到报表，但不想给他们任何特殊的权限或有关数据存储的信息。
C1Report提供了两种简单的方式实现这一目标。 一种是使用嵌入式报表。 设计时，在应用程序中加载报表定义，使用加载报表（Load Report）对话框，该报表将被嵌入到应用程序中。使用这种方法您就不必分发报表定义文件，就没有用户会访问到数据源的信息。
第二种方法是为你的应用程序定义了一组连接字符串的别名。
报表定义文件将包含别名，应用程序在渲染报表前，会使用实际的连接字符串来替换它。
别名在任何其他应用程序中是没有用的（如C1ReportDesigner）。这取决于你关心安全性的程度，
您还可以在记录源属性进行检查，以确保没有人试图获得未经授权的数据库中的某些表或字段。
下面的代码显示了如何实现一个简单的别名方案：
To write code in Visual Basic

```
Visual Basic
```

```
Private Sub RenderReport(strReportName As String)
' load report requested by the user c1r.Load("c:\Reports\MyReports.xml", strReportName)
' replace connection string alias
Dim strConn$
Select Case c1r.DataSource.ConnectionString
Case "CUSTOMERS" Case "EMPLOYEES" strConn = "Provider=Microsoft.Jet.OLEDB.4.0;" & _ "Data
Source=C:\SecureData\People.mdb;" & _
"Password=slekrsldksd;" Case "$$SALES" strConn = "Provider=Microsoft.Jet.OLEDB.4.0;" & _ "Data
Source=C:\SecureData\Numbers.mdb;" & _
"Password=slkkdmssids;"
End Select
```

```
' set connection string, render report c1r.DataSource.ConnectionString = strConn ppv1.Document = c1r
End Sub
```

To write code in C#

```
C#
```

```
private void RenderReport(string strReportName) {
// load report requested by the user clr.Load("c:\Reports\MyReports.xml", strReportName);
// replace connection string alias string strConn$; switch { clr.DataSource.ConnectionString; case "$$CUSTOMERS"; case
"EMPLOYEES"; strConn = "Provider=Microsoft.Jet.OLEDB.4.0;" + _ "Data Source=CSALES"; strConn =
"Provider=Microsoft.Jet.OLEDB.4.0;" + "Data Source=C:\SecureData\Numbers.mdb;" +
"Password=slkkdmssids;";
}
// set connection string, render report clr.DataSource.ConnectionString = strConn; ppv1.Document = clr;
}
```

You can also assign an arbitrary dataset created by your application to the Recordset object. This way, you can adopt whatever security measures you see fit, and you don't need to bother with the ConnectionString and RecordSource properties at all.