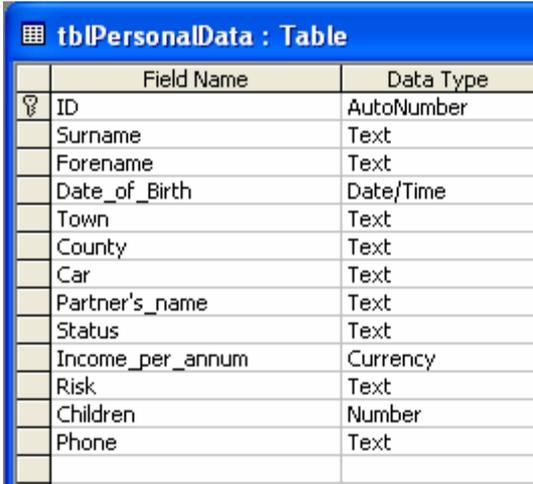# Custom Reports on the Fly from an Access Form Interface

How to give your Access users flexible reports using simple combo box functionality

Reports are a funny thing. You can build a dozen different reports in a small database, and still the user will complain that, "The important fields just aren't shown here." Take the most popular form of report, for instance, what I call a list report. This is the simplest of all reports. Along the top of the page, there are field headings, and below, there are rows of actual data. Each line is a separate record.

Generally speaking, users will use this kind of report as a quick summary. The database that lists customers, for instance, may contain 30 or more fields for each customer, but a simple report showing their names, phone numbers and value of orders will often be enough to let the telemarketing team target a particular call out exercise. This is, of course, until the time that someone, somewhere, wants one or more of those other 30 fields just added to the report.

This is easy, of course, if you're using a full installation of Access and you know how to create your own reports. But for the ordinary data user, who doesn't go (or isn't allowed to go) under the hood, this is unviable. With a little work, you can give users of your Access application – even runtime version users – the ability to add fields to standard reports without having to work in the Report Design window. If they can work a pull-down list, they can make a custom report this way.

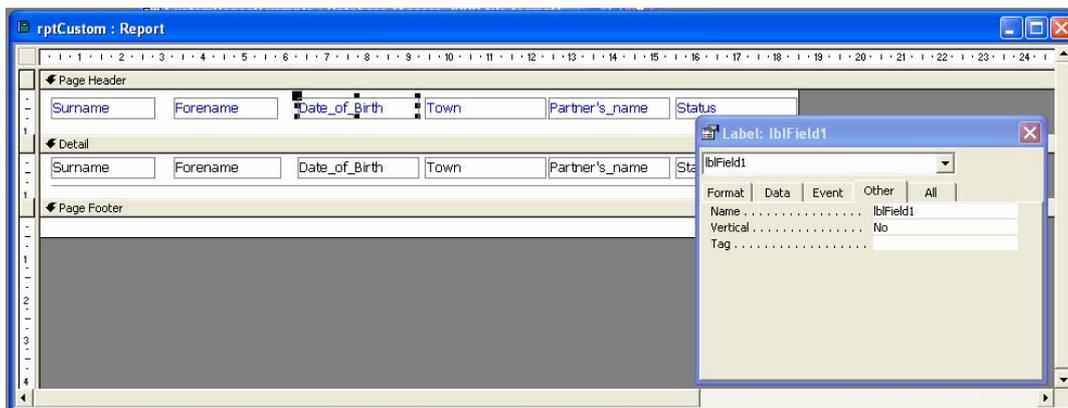Let's start by looking at a pretty simple database that I'll be basing this tutorial on.

There it is. For now, it's just a single table, a good, old-fashioned flat file database. For our custom report, that will be the data source, but it could as easily be a query, bringing together information from a far more complex data structure.

There's a report I've set up, too, which is again pretty conventional:

Design View, though, shows that certain planning has gone into the architecture of the report:
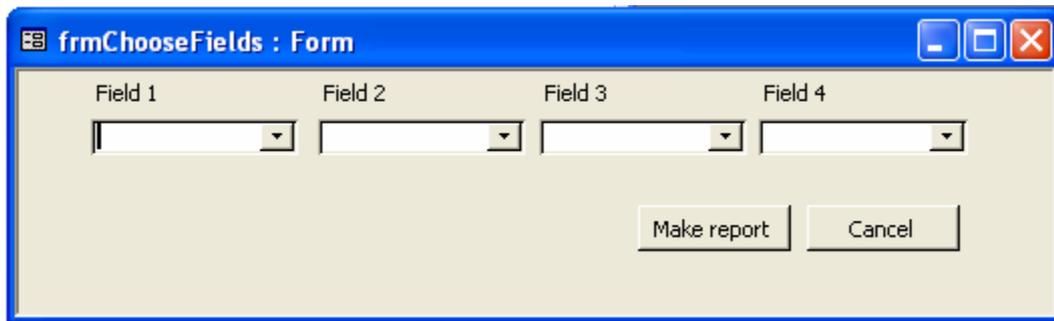

In the Page Header, we start with Surname and Forename, and the text boxes directly below in the Detail section display the values in these fields. These two will be the only fixed items on the report. There are four labels to the right of these in the Page Header, together with corresponding text boxes below in the Detail section.

I have named these objects as below:

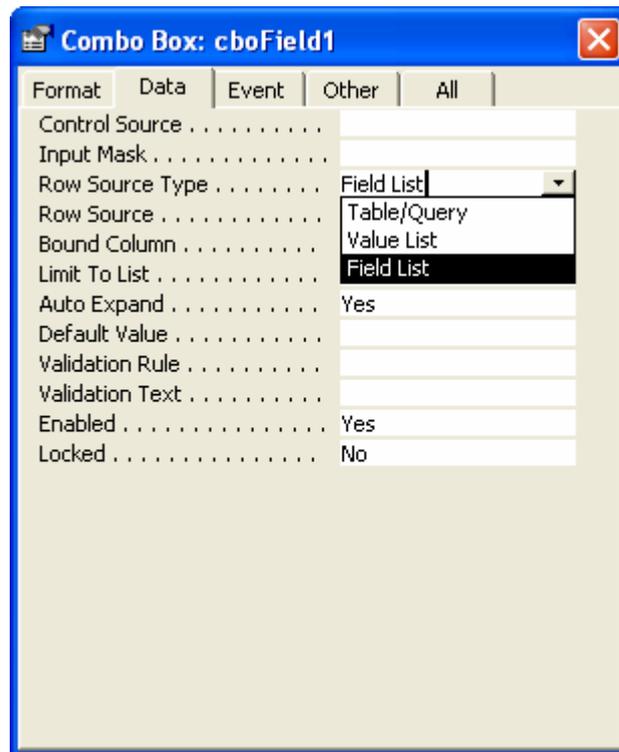| Page Header					
lblSurname	lblForename	lblField1	lblField2	lblField3	lblField4
Detail					
tbSurname	tbForename	tbField1	tbField2	tbField3	tbField4

And this is the key to the entire process. What I will need my code to do is to write new values to the four label objects and the four text box objects to change them into different fields.

But I'm getting slightly ahead of myself, because what I need first is a mechanism for the user to choose what fields he or she wants in the report. I've built a form to do this:
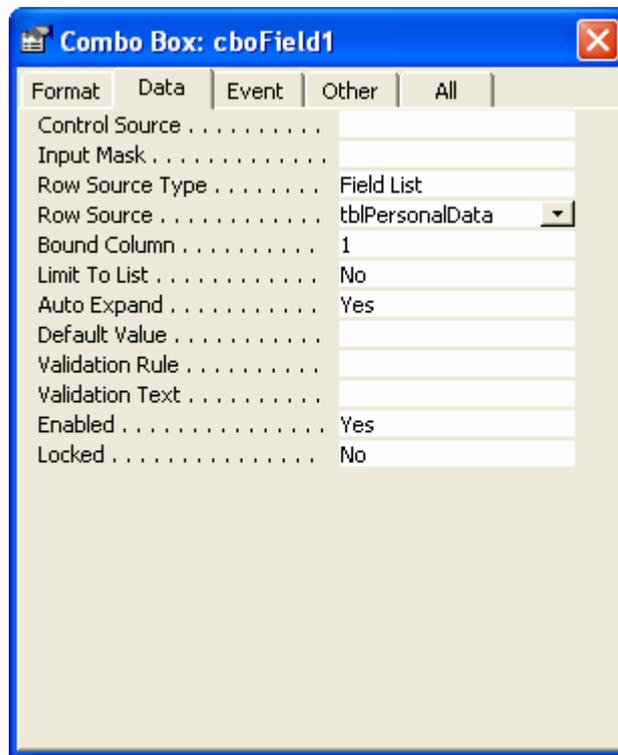

Those four combo boxes area called cboField1, cboField2, cboField3 and cboField4. You can probably see a pattern emerging here. When we get down to the code, by the way, you'll be very grateful I've used prefixes in line with a common naming convention.
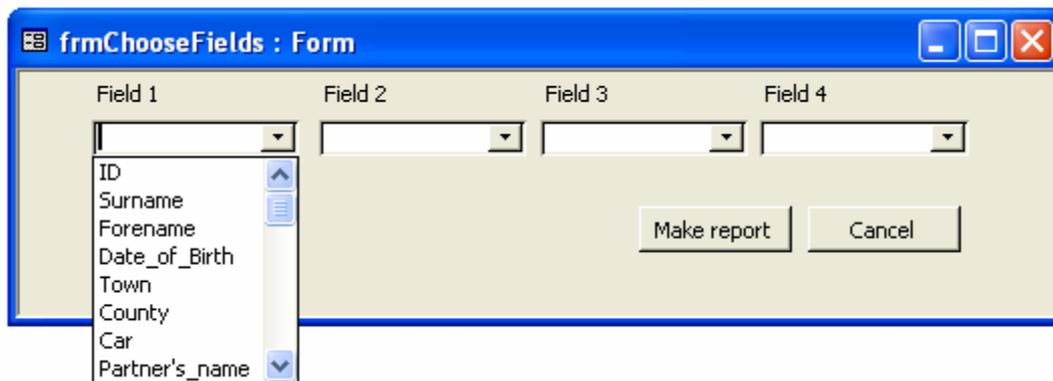
To set the lists for these combo boxes, I used Access's little-known ability to use table or query field names as a row source:

When selecting 'Field List' as the RowSourceType property, I am then offered a list of tables and queries as the RowSource. I chose our main table, of course, which is the record source for the report:

Once this is set, the combo boxes list the field names from the table, thus:

I can't put it off any longer: it's time to do some code. Here's what I want the code to do:

1. Read the value the user selected in cboField1
2. Check whether that value is a 'null'
   a. If it is a 'null' then blank both lblField1 and tbField1 in the report
   b. Otherwise:
      i. write the field name selected as the Caption property of lblField1
      ii. write the field name selected as the ControlSource property of tbField1
3. Repeat for the other three combo boxes

Here's the code, which I put in a new module:

```
Option Compare Database
Option Explicit

Sub MakeReport()
On Error GoTo Err_MakeReport

'Open report in design view to write properties to objects
    DoCmd.OpenReport "rptCustom", acDesign

'Read combo box selections and use subroutine to set report object properties
    SetReportControls Forms!frmChooseFields.cboField1.Value, _
        Reports!rptCustom.lblField1, Reports!rptCustom.tbField1
    SetReportControls Forms!frmChooseFields.cboField2.Value, _
        Reports!rptCustom.lblField2, Reports!rptCustom.tbField2
    SetReportControls Forms!frmChooseFields.cboField3.Value, _
        Reports!rptCustom.lblField3, Reports!rptCustom.tbField3
    SetReportControls Forms!frmChooseFields.cboField4.Value, _
        Reports!rptCustom.lblField4, Reports!rptCustom.tbField4

'Close design view without prompting to save changes
    DoCmd.Close acReport, "rptCustom", acSaveYes

'Open finished report in preview view
    DoCmd.OpenReport "rptCustom", acPreview

Exit_MakeReport:
    Exit Sub

Err_MakeReport:
    MsgBox Err.Description
    Resume Exit_MakeReport

End Sub
```

```
Sub SetReportControls(varFieldName As Variant, conLabel As Control, conTextBox As Control)

'Check if selection is 'null'
    If IsNull(varFieldName) Then 'Blank out the relevant objects
        conLabel.Caption = " "
        conTextBox.ControlSource = ""
    Else 'Write the selected field name to the appropriate objects
        conLabel.Caption = varFieldName
        conTextBox.ControlSource = varFieldName
    End If

End Sub
```

I also need just a little code behind my form to make the buttons work:

```
Option Compare Database
Option Explicit
```

```
Private Sub btnCancel_Click()

    DoCmd.Close

End Sub
```

```
Private Sub btnMakeReport_Click()

    MakeReport

End Sub
```
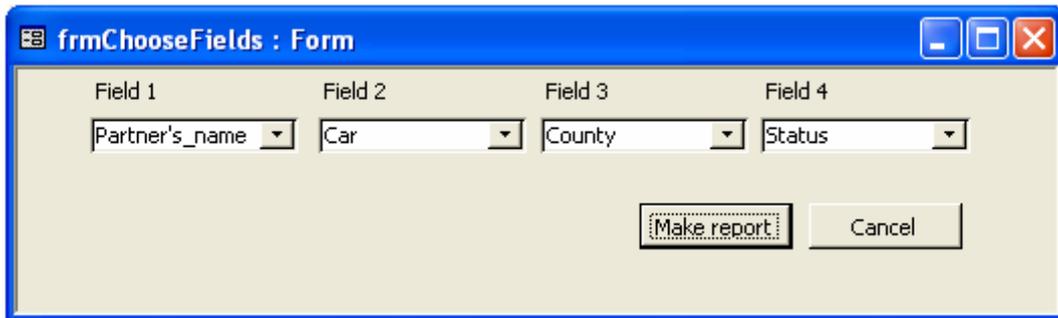
We're now up and running. Just to test it all, here's my selection on the form:

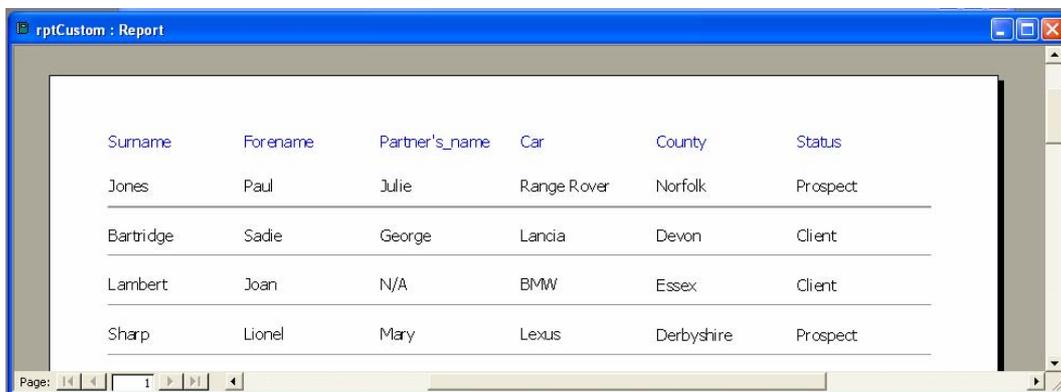And here is the report I get when I hit the button:

So, there we are. Within the context of an Access application, this is a very powerful technique and one which I know from experience can dramatically increase your users' satisfaction with the system.