

### DataSnap 程式設計 3 – 建立具備 CRUD 能力的分散式 JSON 應用程式(1)

在前二篇相關的文章中我們已經介紹了如何使用 Delphi 2009 建立分散式 JSON 伺服器，但是在前 2 篇的文章中建立的範例 JSON 用戶端應用程式只能瀏覽資料，並無法對於資料進行 CRUD 的動作，因此在本篇文章中我們將繼續說明如何在分散式 JSON 應用程式中加入 CRUD 的功能。

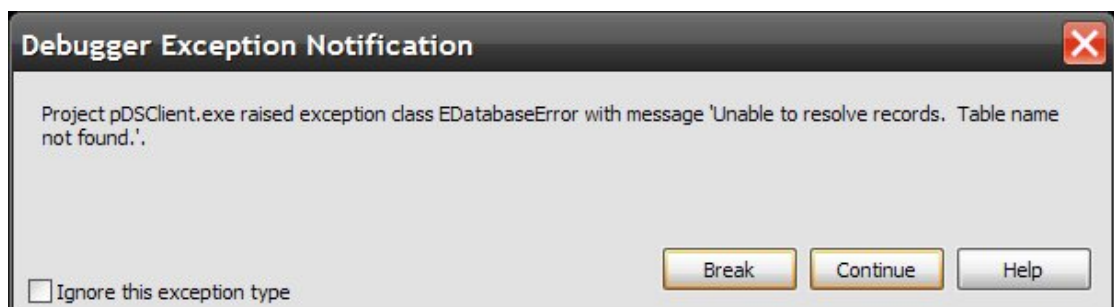
基本上要在分散式 JSON 應用程式中加入 CRUD 的能力可使用兩種方式，第一種是直接使用 Delphi 2009 提供的新元件，第二種則是在 JSON 伺服器中定義中介方法，讓 JSON 用戶端呼叫中介方法來進行 CRUD 的工作。

直接使用 Delphi 2009 的元件在 JSON 分散式應用程式中加入 CRUD 的功能好處是開發快速，接近 Delphi 強調的 RAD 精神，但缺點則是 JSON 用戶端和 JSON 伺服器繫結的太緊密，對於日後的維護和開發比較需要花費較多的成本。而定義中介方法則類似相反，在開始時需要花費較多的時間來定義服務方法和撰寫較多的程式碼，但好處則是 JSON 用戶端和 JSON 伺服器繫結不緊密，對於日後的維護和開發比較輕鬆。

在我們開始說明如何開發 CRUD 的分散式 JSON 應用程式之前，假設我們已經擁有了一個 JSON 伺服器和一個 JSON 用戶端應用程式，這個 JSON 伺服器目前的功能只是把範例資料庫中的『主講人』資料表輸出到用戶端，而 JSON 用戶端目前也只能瀏覽資料，如下所示：



如果現在 JSON 用戶端應用程式試著更新資料回 JSON 伺服器的話，那麼就會產生如下的錯誤：



現在讓我們開始在這個範例應用程式中加入 CRUD 的能力。

## 使用 Delphi 2009 的新元件加入 CRUD 的功能

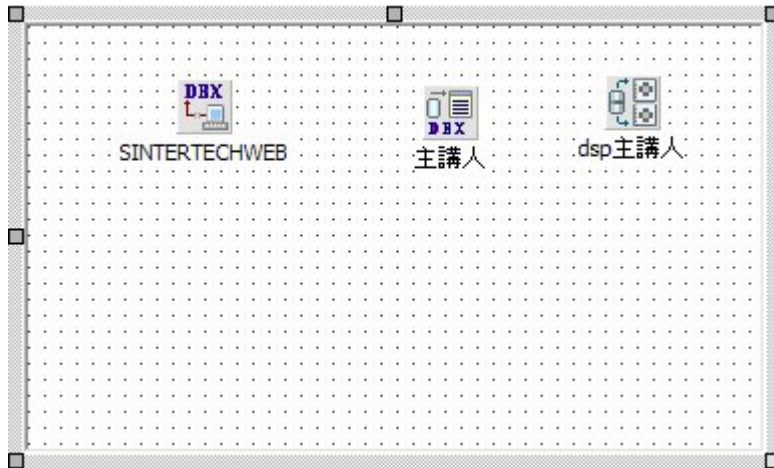
---

要在上述的範例 JSON 應用程式中加入 CRUD 的功能很簡單，只需要使用 Delphi 2009 中提供的新元件：TDSProviderConnection。

TDSProviderConnection 是使用在用戶端，連結到 JSON 伺服器中提供資料的 TDataSetProvider 元件，因此要使用 TDSProviderConnection，首先

我們需要在原先的 JSON 伺服器中加入 TDataSetProvider 元件好讓用戶端的 TDSProviderConnection 連結，使用。

所以，先開啓 JSON 伺服器，打開它的 ServerModule，然後在其中加入一個 TDataSetProvider：『dsp 主講人』，連結它到 ServerModule 中的『主講人』元件：



設定 TDataSetProvider 的特性值如下：

特性	特性值
Name	dsp 主講人
DataSet	主講人

接著編譯和執行 JSON 伺服器，並且準備修改 JSON 用戶端應用程式。

如果您在關閉 JSON 伺服器時，發現它的執行樣例並沒有從 OS 中結束，而無法編譯產生新的 JSON 伺服器時，那是因為您在結束 JSON 伺服器時，並沒有結束 TDSServer 元件的執行，因此您可以在 JSON 伺服器的主菜單的 OnDestroy 事件處理函式中加入如下的程式碼來結束 TDSServer 的執行：

```
procedure TForm2.FormDestroy(Sender: TObject);  
  
begin  
  
    DSServer1.Stop;  
  
end;
```

如此一來您的 JSON 伺服器執行樣例就可以正常結束了。

開啓 JSON 用戶端應用程式，在主表單中加入 TDSProviderConnection 元件和一個 TClientDataSet 元件(我使用一個新的 TClientDataSet 元件來做為展示的說明，讀者也可以使用原先在主表單中的 TClientDataSet 元件)，接著設定如下的特性值：

TDSProviderConnection 元件：

特性	特性值
Name	dspc 主講人
SQLConnection	SQLConnection1
ServerClassName	TDSServerModule1

```

type
10 TDSServerModule1 = class(TDSServerModule)
    SINTERTECHWEB: TSQLConnection;
    主講人: TSQLDataSet;
    dsp主講人: TDataSetProvider;
    private
        { Private declarations }
    public
        { Public declarations }
        function GetSpeaker : TDataSet;
    end;
20
var
    DSServerModule1: TDS
implementation
    {$R *.dfm}
    { TDSServerModule1 }
30 function TDSServerModule1.GetSpeaker: TDataSet;
begin
    Self.SINTERTECHWEB.Connected := True;
    Self.主講人.Active := True;
    Result := Self.主講人;
end;

```

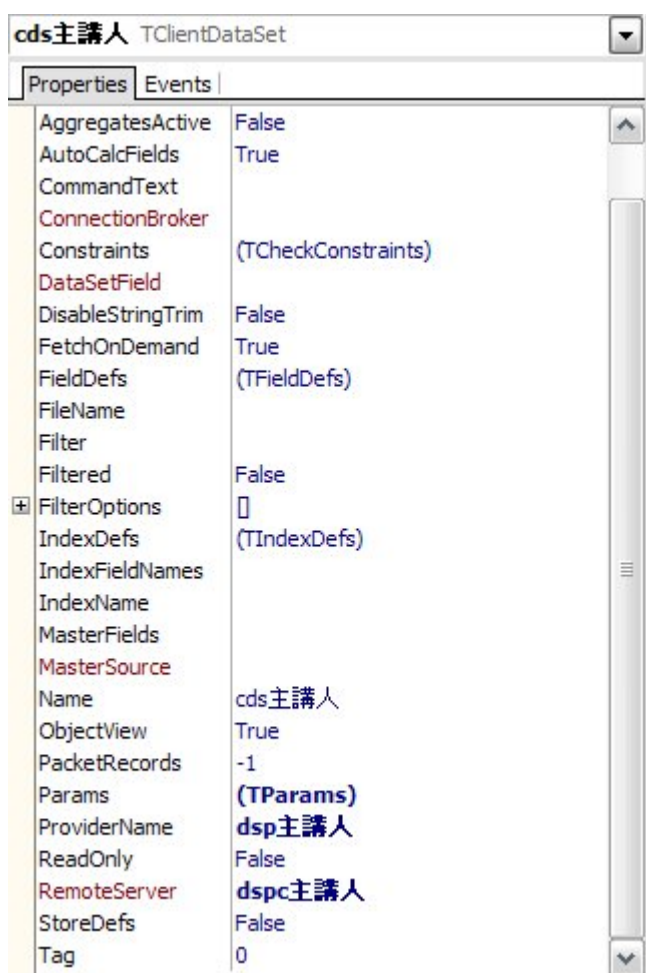


TDSProviderConnection 元件需要討論的特性便是 ServerClassName，ServerClassName 特性需要設定的特性值是 JSON 伺服器中包含 TDataSetProvider 的容器類別名稱，用戶端的 TDSProviderConnection 元件將連結這個遠端 TDataSetProvider 元件並且更新資料回 JSON 伺服器。

由於在前面修改 JSON 伺服器時，TDataSetProvider 是在 TDSServerModule1 類別中連結到『主講人』這個 TSQLDataSet，因此用戶端的 TDSProviderConnection 元件的 ServerClassName 特性就必須設定為遠端的 TDSServerModule1。

設定好了 TDSProviderConnection 元件之後，再讓我們設定新加入的 TClientDataSet 元件如下：

特性	特性值
Name	cds 主講人
RemoteServer	dspc 主講人
ProviderName	dsp 主講人



設定連結 TDSProviderConnection 元件的 TClientDataSet 元件也需要說明一下，這個 TClientDataSet 元件首先必須藉由 RemoteServer 特性連結到用戶端的 TDSProviderConnection 元件，接著在點選 ProviderName 特性之後，ProviderName 特性的特性值編輯器便會顯示 TDSProviderConnection

元件連結的遠端 **TDataSetProvider** 元件的名稱，在這個範例中，遠端 **TDataSetProvider** 元件的名稱自然就是『dsp 主講人』了。

設定好了 **TDSProviderConnection** 和 **TClientDataSet** 元件之後，此時範例用戶端 JSON 應用程式的主表單看起來如下所示：

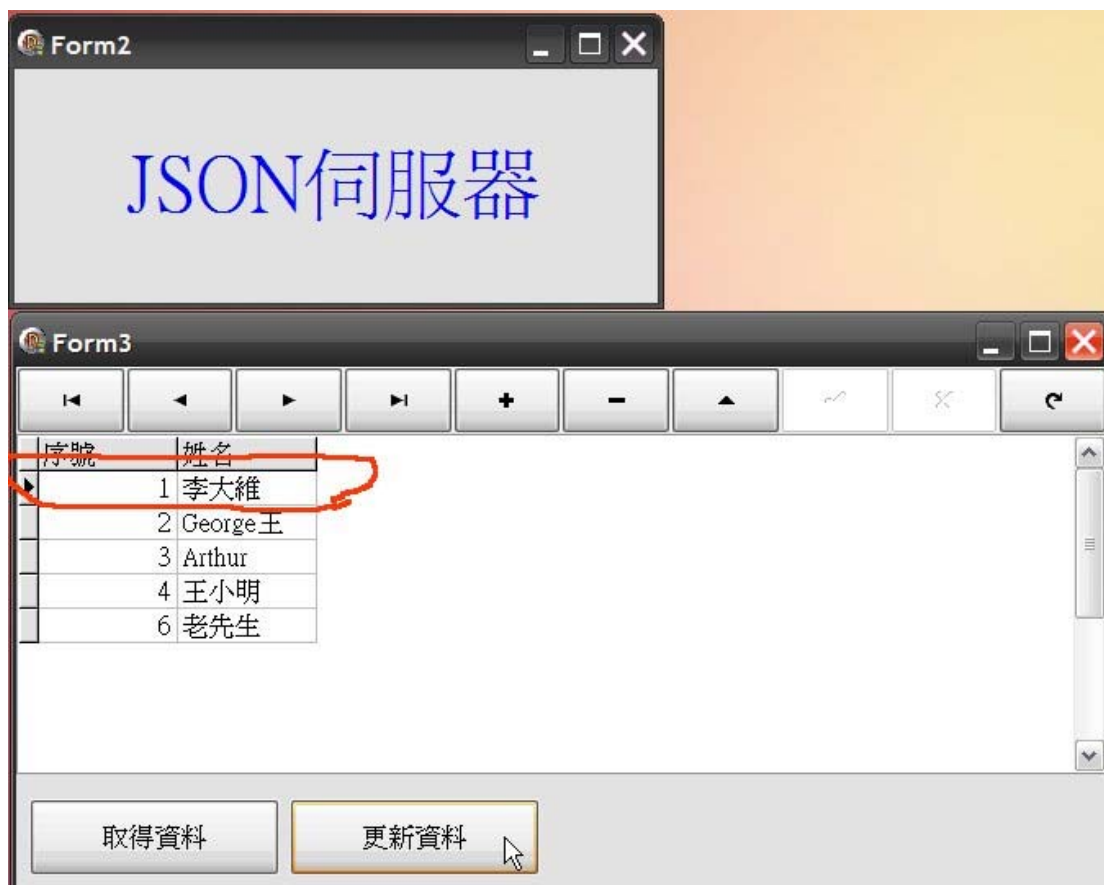


最後的步驟就是修改原先主表單中『更新資料』按鈕的程式碼，現在我們只需要使用新加入的 **TClientDataSet** 元件來更新資料回 JSON 伺服器即可：

```
procedure TForm3.Button2Click(Sender: TObject);
begin
// ClientDataSet1.ApplyUpdates(0);
Self.cds 主講人.ApplyUpdates(0);
end;
```

最後讓我們改變主表單中的 **TDataSource** 的 **DataSet** 特性值為 **cds 主講人**，那麼在 Delphi 整合發展環境的設計時期就可以直接設定『**cds 主講人**』元件的 **Active** 特性值為 **True**，如此一來用戶端的 JSON 應用程式便可以在 IDE 中直接向 JSON 伺服器請求資料，並且資料也可以直接在 IDE 中顯示出來，再也不需要像前面一樣必須在執行時期點選主表單上的『取得資料』按鈕才能夠顯示資料，到這裡為止，範例 JSON 應用系統的行爲幾乎和以前的 C/S 或是使用 COM/DCOM/COM+的應用系統一模一樣了。

現在如果我們執行此範例用戶端 JSON 應用程式，並且試著修改一些資料，例如下圖是把『李維』這個主講人修改為『李大維』，之後再點選主表單中『更新資料』按鈕，那麼現在資料就可以正確的藉由 JSON 伺服器更新回資料庫之中。



現在這個範例 JSON 應用系統已經具備 CRUD 的能力了。

讀者可以下載文討論的範例應用程式壓縮檔：

可異動的 JSON 範例第 1 版.ZIP。