



Blog

-  [Entries](#)
-  [Summary](#)

Listed by:

- Date
- September 2009
- August 2009
- July 2009
- June 2009
- May 2009
- April 2009
- March 2009
- February 2009
- January 2009
- December 2008
- November 2008
- October 2008
- September 2008
- August 2008
- July 2008
- June 2008
- May 2008
- April 2008

September 23  
JSON 程式設計 – Marshaling/Unmarshaling

上篇文章中說明了如何使用Delphi 2010中通用的JSON相關類別來進行程式設計並且以一個TEmployee類別做為範例來說明如何把TEmployee物件轉換為JSON格式的資料並且藉由DataSnap傳遞到用戶端。

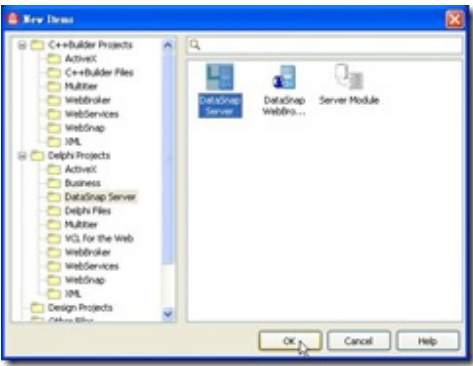
由於類似這種傳遞Value Object的應用在JSON程式設計的世界被非常廣泛的應用，因此DataSnap 2010正式加入了JSON Marshaling/Unmarshaling的功能，允許程式師把物件轉換為符合JSON格式的序列化物件傳遞，例如從伺服端傳遞到用戶端。這樣做的好處是只要支援JSON的技術都可以自由的解析其中的資料並且加入處理，如此一來DataSnap 2010的JSON Marshaling/Unmarshaling技術就可以達成和同質或是異質平台互動的能力，也就是說DataSnap 2010的JSON Marshaling/Unmarshaling技術是已經是一個跨平台的分散式架構了。

使用DataSnap 2010的JSON Marshaling/Unmarshaling非常的簡單，因為對大多數的物件而言，DataSnap 2010能夠自動的把它轉換為TJSONObject型態的物件，然後再加以傳遞，因此開發人員只需要撰寫非常簡易的程式碼就可以輕易的完成。現在讓我們仍然使用上篇文章中的TEmployee類別做為說明，看看使用JSON Marshaling/Unmarshaling功能之後在分散式環境中傳遞物件是如何的容易。

建立DataSnap伺服器

Delphi 2010提供了新的DataSnap精靈幫助開發人員快速建立DataSnap伺服器，如此一來我們就不必每次都不斷重覆的放入TDSServer和TDSTCPServerTransport等元件，可節省我們的開發時間。此外DataSnap 2010也開始支援HTTP/HTTPS通訊協定，DataSnap精靈允許開發人員自動設定更多的元件組態資訊。

啟動Delphi 2010，點選File|New啟動New Items對話盒並且於DataSnap Server分類中選擇DataSnap Server圖像以建立DataSnap伺服器：



接著在New DataSnap Server對話盒中讓我們建立一般的VCL Forms應用程式做為DataSnap伺服器的型態，選擇使用TCP/IP和HTTP/HTTPS通訊協定，並且選擇使用TDSServerModule做為類別做為輸出服務的父代類別，如下所示：

March 2008

February 2008

January 2008

December 2007

November 2007

October 2007

September 2007

August 2007

July 2007

June 2007

May 2007

April 2007

March 2007

February 2007

January 2007

December 2006

November 2006

October 2006

September 2006

August 2006

July 2006

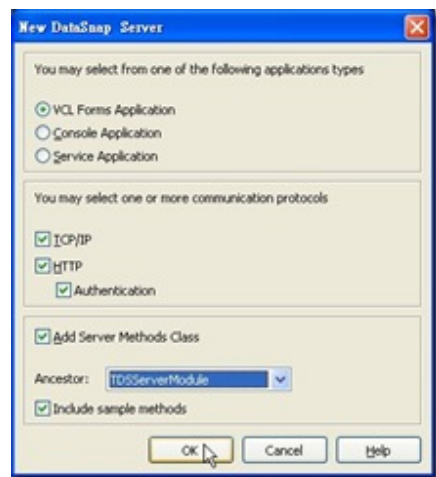
June 2006

May 2006

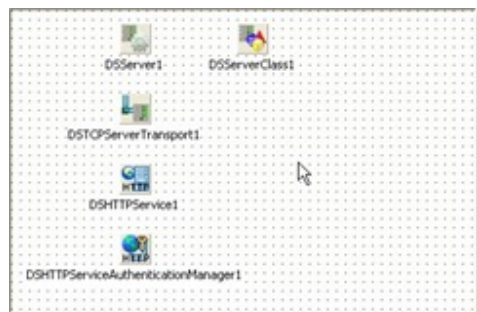
April 2006

March 2006

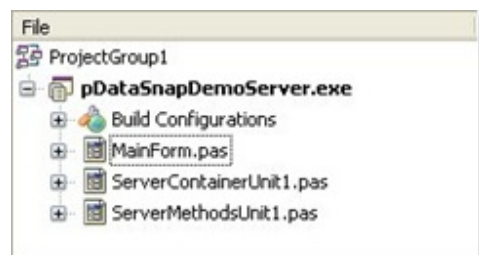
February 2006



點選OK按鈕之後Delphi 2010便會為自動我們建立如下的ServerContainerUnit1程式單元，其中即包含了所有必要的元件。其中的TDSHTTPService和TDSHTTPServiceAuthenticationManager是新的元件，這兩個元件是使用來支援HTTP/HTTPS通訊協定的。



現在讓我們把Unit1 程式單元儲存為名為MainForm的程式單元：



接著點選TDSHTTPService元件，設定它的HttpPort特性值為8080，因為通信埠80已經被Web伺服器使用，因此在這個範例中我使用8080，如果通信埠8080在讀者的機器中已經被使用，那麼您可以使用任何尚未使用的通信埠。請注意TDSHTTPService的RESTContext特性，您可以看到它的特性值是Rest，這代表Delphi 2010的DataSnap伺服器一旦使用TDSHTTPService元件就自動支援RESTful的功能，我們稍後再討論RESTful。

January 2006

December 2005

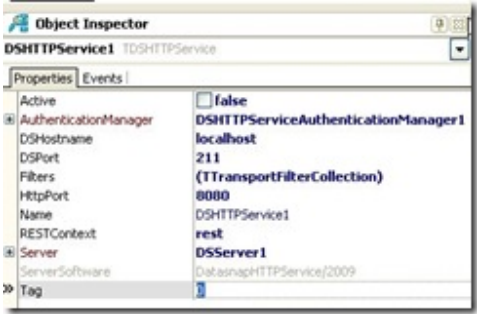
November 2005

October 2005

September 2005

August 2005

July 2005



由於Delphi 2010強化了RTTI提供Reflection的機制，因此DataSnap 2010允許開發人員選擇使用TDSServerModule，TDataModule或是TPersistent類別做為輸出服務的父代類別，因為Delphi 2010都可以藉由RTTI從任何父代類別取得必要的資訊。

現在讓我們在這個DataSnap伺服器專案中加入上篇文章中的TEmployee程式單元，並且輸出GetEmployee方法讓用戶端能夠從伺服器端取得TEmployee物件。請注意GetEmployee回傳TJSONValue型態的數值，由於DataSnap 2010在參數列和回傳值都開始支援TJSONValue，因此這代表只要我們把任何物件轉換為TJSONValue物件，就可以自由傳遞到用戶端，或是從用戶端傳遞回伺服器端，而這個流程就是DataSnap 2010的JSON Marshaling/Unmarshaling技術。

```
TServerMethods1 = class(TDSServerModule)
```

```
private
```

```
{ Private declarations }
```

```
public
```

```
{ Public declarations }
```

```
function EchoString(Value: string): string;
```

```
function GetEmployee : TJSONValue;
```

```
end;
```

要使用JSON Marshaling/Unmarshaling，請加入使用DBXJSONReflect程式單元，如下004行所示。

要以JSON格式傳遞物件非常的簡單，首先我們需要建立我們要傳遞的物件，一個TJSONMarshal物件以及一個TJSONConverter物件。因此下面的011行先建立TEmployee物件，012行建立TJSONMarshal物件，並且在它的建構函式中傳入一個TJSONConverter物件，最後我們只需要呼叫TJSONMarshal物件的Marshal虛擬方法並且傳入要傳遞的物件即可，如014行所示：

```
01 implementation
```

```
002
```

```
003 {$R *.dfm}
```

```
004 uses DBXJSONReflect, uEmployee;

005

006 function TServerMethods1.GetEmployee: TJSONValue;

007 var

008 anEmployee : TEmployee;

009 aMarshaler : TJSONMarshal;

010 begin

011 anEmployee := TEmployee.Create('王大明', 'wtm@somemail.com', '123456');

012 aMarshaler := TJSONMarshal.Create(TJSONConverter.Create);

013 try

014 Result := aMarshaler.Marshal(anEmployee);

015 finally

016 FreeAndNil(anEmployee);

017 FreeAndNil(aMarshaler);

018 end;

019 end;
```

如此一來TJSONMarshal和TJSONConverter類別就可以自動把傳入的物件轉換為JSON的格式並且傳遞出去。

#### 建立DataSnap用戶端

現在建立一個VCL Form應用程式專案，放入TSQLConnection元件，連結到DataSnap伺服器(如果您不知道如何做，請參考筆者的Tiburon遊記系列文章)，建立代表DataSnap伺服器輸出服務的用戶端程式單元uServerProxy，同樣把TEmployee程式單元加入到用戶端專案中，再參考DBXJSONReflect和DBXJSON程式單元，如下001行所示。

在用戶端要反轉JSON物件回正確物件的流程就是Unmarshaling，我們只需要在013行建立TJSONUnMarshal物件，015行呼叫DataSnap伺服器的GetEmployee服務方法，GetEmployee回傳TJSONValue型態的物件，016行先存取回傳的TJSONValue物件的JSON格式文字值，接著018行呼叫TJSONUnMarshal的Unmarshal方法，傳入回傳的TJSONValue物件並且使用**as**運算元把它正確的轉換回TEmployee物件，之後我們就可以自由的存取TEmployee物件的特性值了。

```
001 uses DBXJSONReflect, DBXJSON, uServerProxy, uEmployee;

002

003 {$R *.dfm}

004

005 procedure TForm2.Button1Click(Sender: TObject);

006 var

007   aServer : TServerMethods1Client;

008   unMarshaler : TJJSONUnMarshal;

009   aValue : TJJSONValue;

010   anEmployee : TEmployee;

011 begin

012   aServer := TServerMethods1Client.Create(Self.SQLConnection1.DBXConnection);

013   unMarshaler := TJJSONUnMarshal.Create;

014 try

015   aValue := aServer.GetEmployee;

016   Edit4.Text := aValue.ToString;

017

018   anEmployee := unMarshaler.Unmarshal(aValue) as TEmployee;

019   Edit1.Text := anEmployee.Name;

020   Edit2.Text := anEmployee.EMail;

021   Edit3.Text := anEmployee.Phone;

022

023 finally
```

```
024 FreeAndNil(anEmployee);
```

```
025 FreeAndNil(unMarshaler);
```

```
026 FreeAndNil(aServer);
```

```
027 end;
```

```
028 end;
```

從下圖可以看到TEmployee物件的確可以正確的從DataSnap伺服器傳遞到用戶端，當然用戶端也可以修改數值之後再傳遞回DataSnap伺服器再更新回資料庫。



## 使用DataExplorer存取DataSnap伺服器的服務

由於DataSnap 2010自動支援RESTful架構，因此既然在前面我們選擇讓DataSnap伺服器支援HTTP/HTTPS通訊協定，因此我們現在就可以使用瀏覽器來存取這個DataSnap伺服器的服務，我們只需要使用下面的樣例就可以存取到DataSnap伺服器的服務：

<http://server url/datasnap/rest/服務程式單元/服務方法>

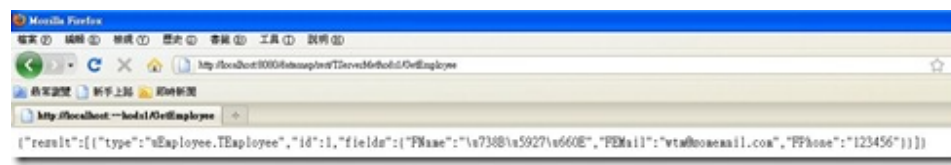
<https://server url/datasnap/rest/服務程式單元/服務方法>

因此要存取前面的DataSnap伺服器，我們只需要使用下面的URL：

<http://localhost:8080/datasnap/rest/TServerMethods1/GetEmployee>

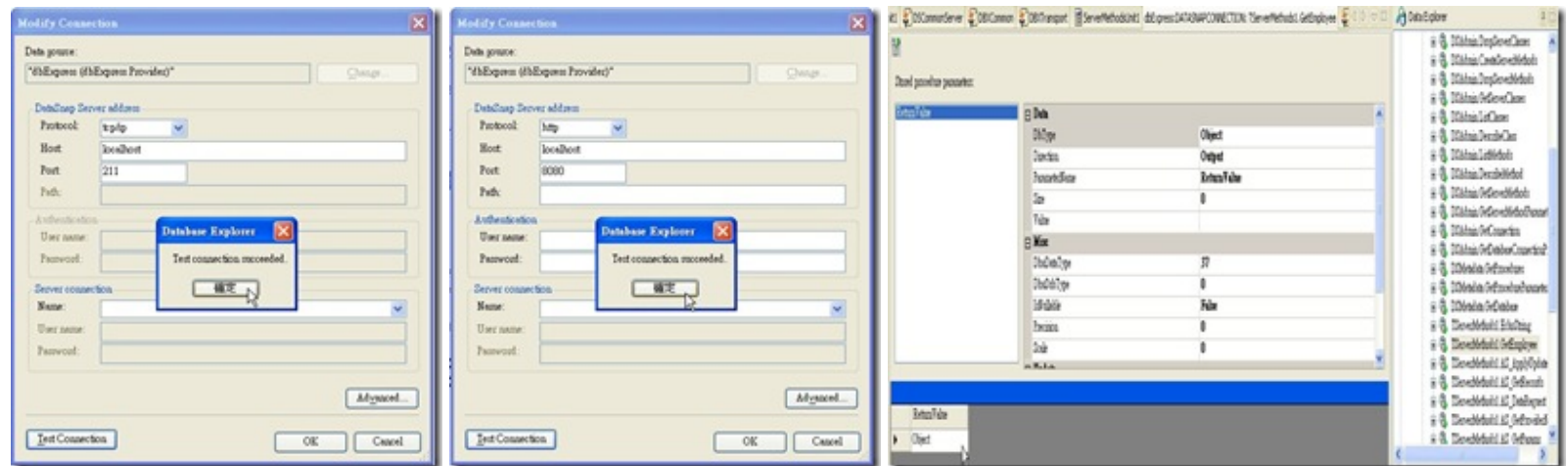
使用8080當然是因為我們前面設定TDSHTTPService元件的HttpPort特性值使用8080。

例如下圖就是筆者使用FireFox存取DataSnap伺服器：



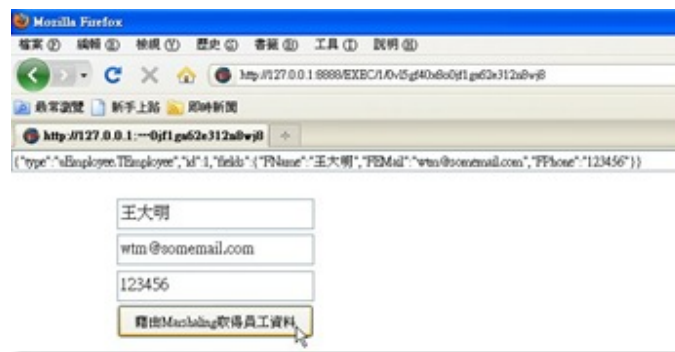
## 使用瀏覽器存取DataSnap伺服器的服務

我們也可以使用Delphi/BCB的DataExplorer來測試和存取DataSnap伺服器，例如下圖的三個圖形分別顯示了筆者使用DataExplorer來測試DataSnap伺服器是否支援TCP/IP，HTTP/HTTPS通訊協定，以及存取GetEmployee服務。請注意第3個圖形，它明確的顯示了DataSnap伺服器回傳的是物件型態(Object)。



## 建立Web用戶端

既然範例DataSnap伺服器支援HTTP/HTTPS通訊協定，因此我們當然可以使用Web應用程式來存取它的服務，例如下圖就是筆者使用VCL For Web建立Web應用程式並且存取DataSnap伺服器的服務：





## 這是如何做到的?

由於篇幅的限制，因此讓我們簡單的說明一下好了。當開發人員使用TJSONMarshal類別Marshaling物件時，TJSONMarshal藉由新的Rtti功能取得任何物件的欄位值，再一一的根據欄位型態呼叫TXXXXConverter類別把欄位值轉換為JSON格式，最後再把所有欄位值組合成TJSONObject的型態傳遞出去。至於Unmarshaling則是進行相反的動作。

那麼DataSnap 2010的Marshaling/Unmarshaling能夠傳遞什麼型態的物件呢? 沒有限制嗎? 這要分2個層面回答。

首先回答第一個問題，Marshaling/Unmarshaling能夠傳遞什麼型態的物件呢?答案是任何從TObject繼承下來的物件都可以，因為Marshaling/Unmarshaling中的實作程式碼已經說明了一切：

```
procedure TTypeMarshaller<TSerial>.MarshalData(Data: TObject);
```

但另外一個關鍵則是物件的欄位型態，前面筆者已經說過，TXXXXConverter類別，因此如果您要傳遞的物件的欄位型態是特別的型態，或是目前DataSnap 2010的TXXXXConverter類別沒有支援的型態，那麼就不能傳遞，當然要克服這很簡單，因為您可以自己寫一個客製化TXXXXConverter衍生類別，並且註冊給DataSnap 2010就可以了，這屬於進階的功能，以後有機會再討論吧。

如何? DataSnap 2010的Marshaling/Unmarshaling功能是不是很強大又很容易使用? 各位讀者可以和上一篇我們自己傳遞TEmployee物件比較一下。善用DataSnap 2010的Marshaling/Unmarshaling功能可以讓我們輕易的開發出土又輕又快的跨平台分散式應用系統，成功的克服了以往COM/DCOM/COM+無法做到的事情，DataSnap 2010提供了目前最佳的分散式解決方案。

我們下次再見，Have Fun!

12:26 PM | [Blog it](#)

### Comments (2)

To add a comment, sign in with your Windows Live ID (if you use Hotmail, Messenger, or Xbox LIVE, you have a Windows Live ID). [Sign in](#)

Don't have a Windows Live ID? [Sign up](#)



**維 李** wrote:  
> 在使用后，不需要释放(FreeAndNil())的吗? TJSONValue 实例的生命周期是Delphi自动管理的吗?

是的, 不需要我們释放, 因為Delphi會自動释放, 答案就在由Delphi 產生的Client Class中, 你從下面產生的程式碼可以看到FInstanceOwner內定為True代表傳回的TJSONObject會由Delphi自動释放:

```
function TServerMethods1Client.GetEmployee: TJSONValue;
begin
  if FGetEmployeeCommand = nil then
  begin
```



```
FGetEmployeeCommand := FDBXConnection.CreateCommand;  
FGetEmployeeCommand.CommandType := TDBXCommandTypes.DSServerMethod;  
FGetEmployeeCommand.Text := 'TServerMethods1.GetEmployee';  
FGetEmployeeCommand.Prepare;  
end;  
FGetEmployeeCommand.ExecuteUpdate;  
Result := TJSONValue(FGetEmployeeCommand.Parameters[0].Value.GetJSONValue(FInstanceOwner));  
end;
```

如果你想自己寫程式碼釋放, 那你要使用下面的程式碼建立TServerMethods1Client

```
aServer := TServerMethods1Client.Create(Self.SQLConnection1.DBXConnection, False);
```

最後再

```
FreeAndNil(aValue);
```

6 hours ago



宇 wrote:

客戶端代碼中

```
aValue := aServer.GetEmployee;
```

得到的 TJSONValue

在使用后, 不需要釋放(FreeAndNil())的嗎?

TJSONValue 實例的生命周期是Delphi自動管理的嗎?

9 hours ago

## Trackbacks

The trackback URL for this entry is:

<http://gordonliwe.spaces.live.com/blog/cns!CCE1F10BD8108687!3826.trak>

Weblogs that reference this entry

- None