# Creazione di una Applicazione Test Server e Client REST

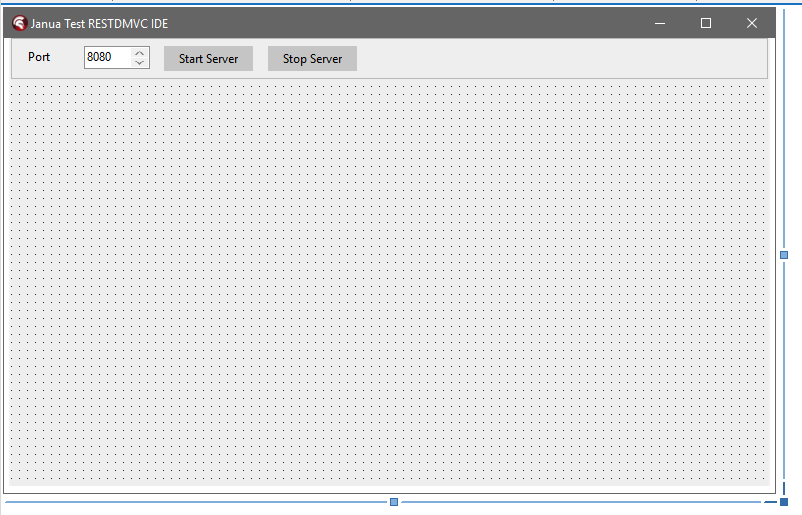
# Container e Server DMVC Interno per i Test

L’idea è di un Form Container (sia VCL che FMX) che possa contenere le varie applicazioni da Testare.

Per il Test di costruzione di questo esempio ho usato

Il form ‘Contenitore’ nasce da una necessità, quella di Mantenere la struttura dell’applicazione da Collaudare e di tutti i suoi Test ma anche poter eseguire ed effettuare il debug dell’application Server REST o di un WebServer all’interno dell’Applicazione (cosa leggermente più complessa).

Quindi occorre Creare una Standard VCL Application come Test Container di tutto il progetto. Che quindi si occupi prima di tutto di avviare il Server REST DMVC Successivamente Testare e verificare che login sia attivo usando un utente di Test.



**type**

TfrmJanuaVCLTestRestDMVCID = **class** (TForm)

pnlMain: TPanel;

**JanuaframeWebServerManager1: TJanuaframeWebServerManager;**

**private**

{ Private declarations }

**public**

{ Public declarations }

**end**;

L’elemento principale è il TJanuaframeWebServerManager che si occupa di fungere da interfaccia tra la Vista (il Form) e il ‘motore’ del DMVC. Si trova come Componente non come ‘frame’ quindi è facile ‘beccarlo’. NOTA l’ho impostato come componente quindi si trova nella Palette.

Per far ‘funzionare’ tutta la baracca Occorrono diverse Uni che vanno riportate in tutti i progetti FMX o VCL in questo caso il nostro esempio è per VCL vedremo di farne uno chiaramente anche per FMX.

Janua.TMS.Error, Janua.Application.Framework, Janua.Uni.Framework, Janua.Vcl.Framework, Janua.ViewModels.Framework, Janua.Orm.Register, Janua.Vcl.MVVM.Framework,

|  |
| --- |
| **uses**  Vcl.Forms,  Janua.TMS.Error,  Janua.Application.Framework,  Janua.Uni.Framework,  Janua.Vcl.Framework,  Janua.ViewModels.Framework,  Janua.Orm.Register,  Janua.Vcl.MVVM.Framework,  ufrmJanuaAllDemosContainer **in** '..\..\..\Samples\Janua\VCL\ufrmJanuaAllDemosContainer.pas' {frmAllDemosContainer};  {$R \*.res}  **var**  errorManager: TJanuaTmsExceptionHandler;    **begin**  TJanuaApplication.AppName := ('desktop.januaproject.it');  Application.Initialize;  errorManager := TJanuaTmsExceptionHandler.Create(Application);  errorManager.Activate;  // \*\*\*\* Here goes the custom configuration for the Application    Application.MainFormOnTaskbar := True;  Application.CreateForm(TfrmAllDemosContainer, frmAllDemosContainer);  Application.Run;  **end**. |

Da notare la configurazione dell’Error Manager ‘custom’ che io uso spesso nelle applicazioni VCL nelle righe, dalla Unit Janua.TMS.Error. Unit che si potrebbe ‘replicare’ per FMX (Tablet/Destk)

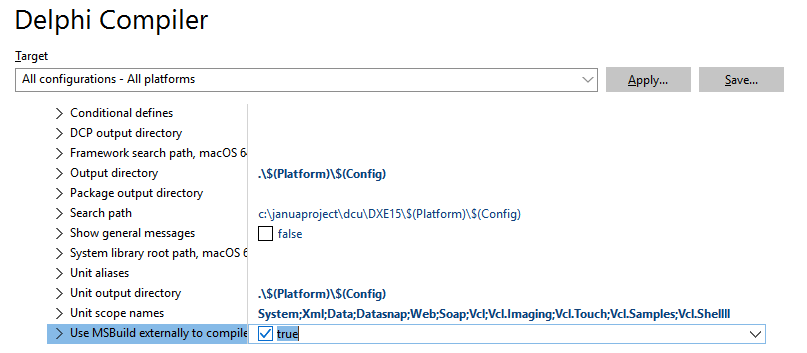
**var**

errorManager: TJanuaTmsExceptionHandler;

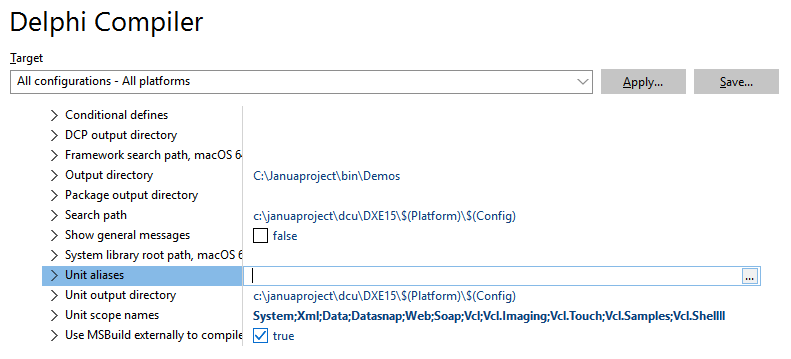
…

errorManager := TJanuaTmsExceptionHandler.Create(Application);

errorManager.Activate;



Usando la Build Configurations occorre cancellare i Default di Unit Output e Output Directory



A questo punto si può almeno testare se compila e si ‘avvia’ (NOTA: spesso non ho configurato gli ambienti a 32 bit perché i miei ambienti target sono da diversi anni solo a 64bit.

Il nostro programma così si avvia (in questo momento è presente giusto l’interfaccia e nessun ‘background’) ho aggiunto un pulsante Login e due controlli UserName e Password. Il primo ‘step’ per ‘avviare’ l’applicazione sarebbe Testare una login con DMVC dopo avere ovviamente fatto partire il server. t

