Un corso rapido per lo sviluppo con InstantSolutions Framework di Ethea S.r.l.

Revisione del 09/02/2024 riferita a InstantSolutions 8.6.1

Indice generale

1. Premessa: cos'è InstantSolutions

1.1. Introduzione allo sviluppo con InstantSolutions Framework

InstantSolutions è un modo innovativo di concepire lo sviluppo di applicazioni basato a tutti i livelli sui paradigmi della OOP. Esso combina la potenza dei motori SQL alla flessibilità dello sviluppo ad oggetti in modo semplice e soprattutto trasparente allo sviluppatore. Tutto questo è possibile grazie ad un meccanismo chiamato Object Persistence Framework (OPF), che consente la memorizzazione di oggetti all'interno di database relazionali, come **InstantObjects** e grazie all'utilizzo di un linguaggio di programmazione a oggetti come Delphi.

I dati, sotto forma di classi, sono descritti in un unico Data Dictionary integrato sia nell'ambiente di sviluppo sia nell'applicazione, dal nome **ISWorkbench**. ISWorkbench è il cuore del modello di una applicazione sviluppata con InstantSolutions.

Ma InstantSolutions non è solo un sistema di sviluppo basato su un OPF. Esso mette a disposizione degli sviluppatori 2 Framework Applicativi già pronti per realizzare applicazioni di diverso tipo e con target di utenza differenti.

L'architettura aperta consente l'integrazione di tecnologie di terze parti, mentre la sua struttura stratificata consente di separare nettamente la logica di business dalla parte di presentazione (interfaccia utente).

InstantSolutions 8.5 funziona con Delphi (versioni Delphi 11 e Delphi 12) e supporta tutti i sistemi operativi Microsoft attualmente in manutenzione qualile varie versioni di Windows 10 e Windows 11.

1.2. Lo sviluppo OO e l'obiettivo di InstantSolutions

L'obiettivo del progetto InstantSolutions di Ethea è quello di arrivare ad utilizzare un framework di sviluppo applicativo che aiuti (e, se necessario, costringa) lo sviluppatore ad essere sempre coerente con i principi della OOP, dal livello dell'interfaccia utente fino alla persistenza dei dati in database SQL.

1.3. La memorizzazione di oggetti e il modello relazionale

Nella tecnologia ad oggetti la complessità di un dato è contenuta (incapsulazione) nell'oggetto, i cui dati (attributi) sono accessibili attraverso interfacce semplici e coerenti (metodi). Per contro, il modello relazionale prevede esso stesso una interfaccia coerente (ad esempio il linguaggio SQL) ma, siccome non gestisce in nessun modo la complessità dei dati, è lo sviluppatore a dover costantemente riferirsi a questa complessità astratta. Inoltre, alcuni paradigmi strettamente OO (come l'ereditarietà) sono di difficile implementazione e di complicata manutenzione all'interno di un database relazionale.

Il framework riduce le difficoltà relative all'accesso ai dati, fornendo allo sviluppatore una interfaccia coerente verso i dati, che sono “visti” sempre come classi di oggetti, e gestendo automaticamente la traduzione delle operazioni di accesso ai dati in linguaggio SQL. In situazioni particolari, per raggiungere le prestazioni ottenibili solo da una elaborazione lato server da parte del motore SQL, InstantSolutions fornisce il supporto per la mappatura di viste logiche e stored procedure in classi.

1.4. La stratificazione e la pertinenza contro con lo sviluppo “spaghetti basic”

InstantSolutions ha scelto la logica della stratificazione e pertinenza del codice come principio di base, abituando lo sviluppatore a rimanere nei confini adatti al codice che viene scritto. Eccone lo schema:

Livello	Attività di sviluppo	Manutentore
GUI custom	**Sviluppo GUI custom per funzioni evolute**	**Sviluppatore**
GUI di base	MultiFramework, ConsoleFramework	Ethea
Business logic custom	**Sviluppo logica di business specifica**	**Sviluppatore**
Business logic di base	Logica di business auto-generata da **ISWorkbench**	Ethea
Accesso ai dati	Sviluppo Broker di InstantObjects	InstantObjects
Livello fisico	Sviluppo Database Relazionale SQL	Database Vendor

È possibile notare come lo sviluppatore interviene solo a 2 livelli (evidenziati in grassetto) basati su altrettanti livelli già implementati da Ethea e quindi è portato in modo naturale a non “sconfinare” e a seguire in modo corretto la separazione del codice.

2. Le versioni di Instant Solutions 8.6

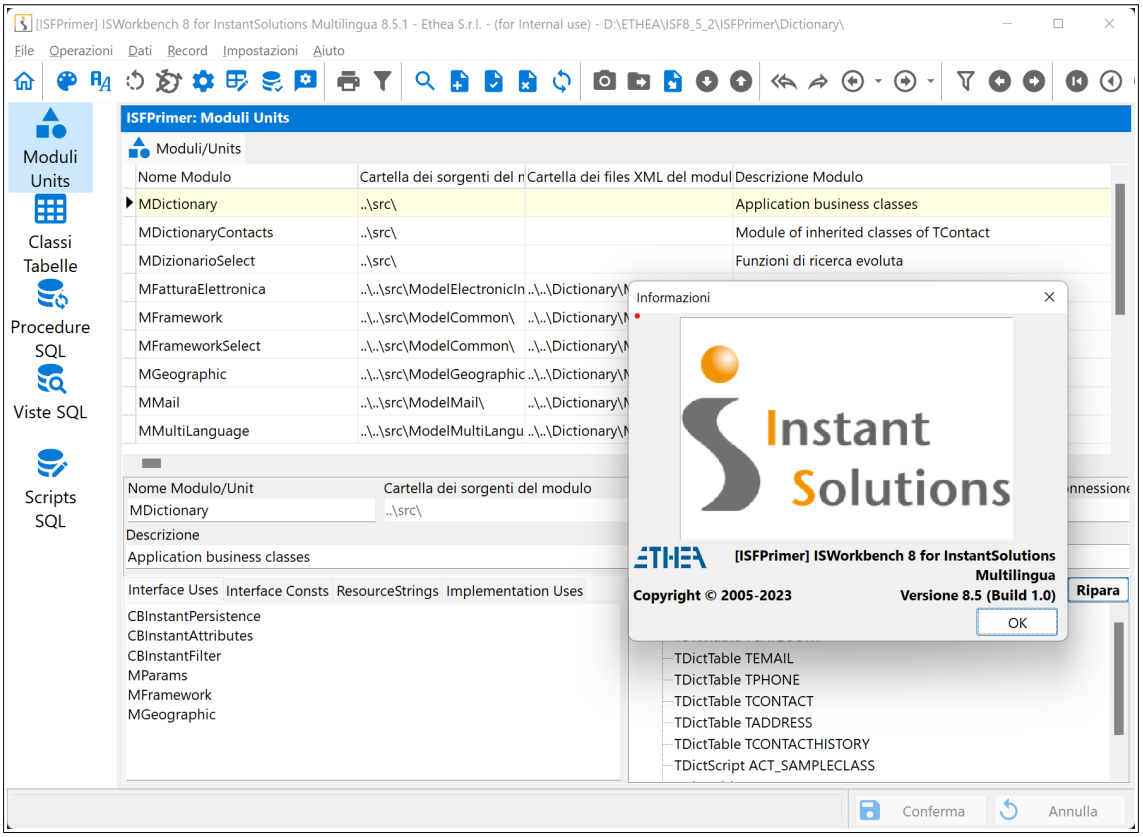
La versione 8.6 esiste in 2 versioni:

1) Versione base (sviluppo in team)

3) Versione multilingua (interfaccia utente in più lingue e gestione dati multilingua).

Per i dettagli sul multilingua leggere la sezione dedicata.

Il cuore dello sviluppo è l'applicazione ISWorkbench, attraverso la quale sono riunite tutte le principali funzioni di organizzazione di un progetto InstantSolutions.



2.1. *Caratteristiche di InstantantSolutions 8.6*

La **versione 8.6** utilizza solo **FireDAC** come Brokers di InstantObjects per l'accesso ai dati sia all'interno di ISWorkBench sia delle applicazioni: in questo modo non è più necessaria la licenza di DBExpress che era necessaria nelle versioni precedenti.

Le versioni di delphi supportate sono la **11 e la 12.** Le versioni precedenti non sono più supportate.

E' stato aggiunto il supporto a **MarkDownHelpViewer** un nuovo sistema di help integrato nelle applicazioni ISF.

E' stato aggiunto il supporto agli **StyledComponents**, un set di componenti visuali/grafici come Pulsanti, Toolbar, DbNavigator e le nuove TaskDialog comprese le animazioni con Skia4Delphi.

Una differenza sostanziale rispetto a ISF7.6 riguarda la la nuova gestione delle icone basate sui componente sviluppato da Ethea e l'utilizzo di **ImageName** anziché **ImageIndex**:

- SVGIconImageCollection (che sfrutta il formato SVG) anche colorate in accoppiata con **VirtualImageList**.

N.B. il supporto a IconFontsImageList è stato rimosso!

Per i dettagli consultare il capitolo: ***Gestione Icone SVG e IconName nei dfm***

La **versione 8** contiene alcune novità già presenti nella 7.6 come la gestione della "HomePage" e il nuovo FlexcelDocProducer, oltre ad un refactoring delle main form che non utilizzano più il sistema di "docking" ma uno "splitview" per ospitare il menu sulla sinistra e il monitor SQL (opzionale) sulla destra.

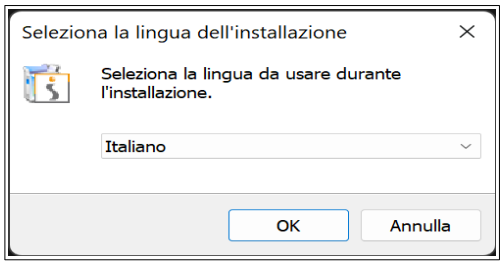
Per maggiori informazioni sul passaggio delle precedenti versioni consultare il documento Migrazione_da_ISF_7_a_ISF_8.pdf.

2.2. *MARS Server framework aggiunto*

A partire dalla versione 7.3.2 è stato introdotto un nuovo framework di sviluppo per applicazioni Server/REST utilizzando le librerie MARS e delphi-neon: vedi capitolo dedicato.

3. Installazione del framework

3.1. *Procedura di installazione del framework e scelta della lingua*

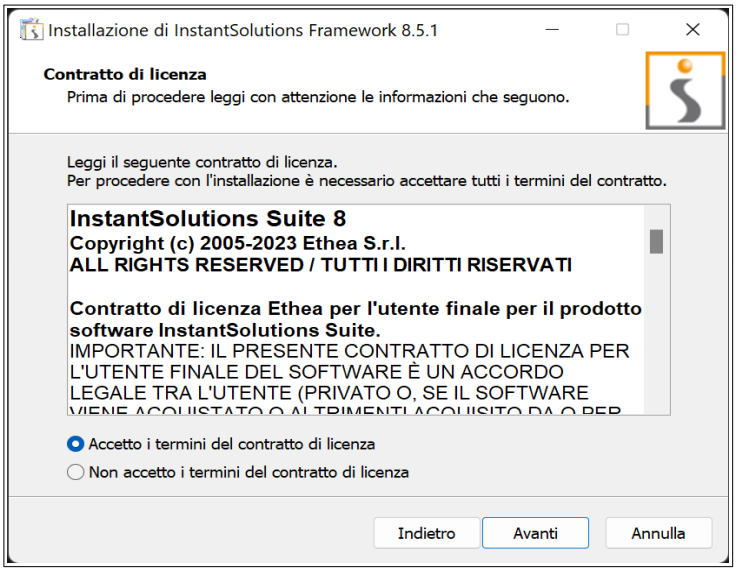


L'installazione del framework è molto semplice. Essa non è invasiva, ma consiste semplicemente nel creare l'ambiente di sviluppo.

N.B. La scelta della lingua, all'inizio dell'installazione ha effetto sul set di files .dfm del framework che vengono installati e utilizzati (in lingua inglese o in lingua italiana). In questo modo nell'IDE di Delphi si potranno vedere le form di base del framework in italiano o in inglese.

Esiste comunque un backup di tali dfm nella cartelle ITA_dfm e ENG_dfm, perciò è sempre possibile utilizzare uno o l'altro set di files, copiandoli nella cartella Src del framework e sovrascrivendo quelli presenti.

L'installazione comprende i seguenti moduli:



- Sorgenti di componenti OpenSource:
  - SynEdit, ChromeTabs, HTMLViewer, OnGuard
- Documentazione
- InstantSolutions Workbench
- Componenti Styled
- Componenti Markdown
- Sorgenti libreria componenti CBLib
- Sorgenti libreria componenti ISFLib
- Sorgenti Framework applicativi
  - MultiFramework Template (form in italiano o in inglese)
  - ConsoleTemplate (applicazioni console)
- Sorgenti Demo dell'applicazione ISFPrimer
- Dizionario classi del framework (XML)
- Template gestione licenze

L'installazione avviene in una cartella a scelta ed è possibile utilizzare più release contemporaneamente sulla stessa macchina in modo semplice. Per poter far questo i proprio progetti devono risiedere all'interno della cartella del framework.

3.2. *Installazione dei componenti nell'IDE di Delphi*

Per installare tutti i componenti nell'IDE di Delphi si può utilizzare il comodo project group fornito (prima si esegue il build dei package “run-time”, poi si installano i package “design-time”.

Nell'IDE ci saranno i nuovi componenti. Alcuni sono i componenti della CBLib, una libreria di base di componenti sviluppati da Ethea (pagine IS.....).

Tra questi componenti spiccano i “doc-producer”, una tecnologia molto evoluta per la generazione di stampe e documenti e il CBXDbMultiEdit, un componente molto utilizzato in ISF per il disegno delle “form” dinamico.

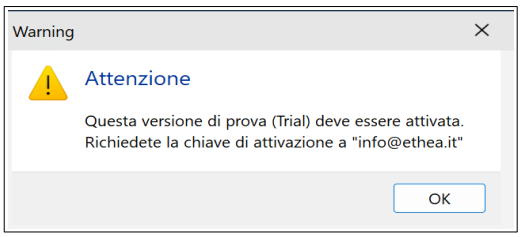
Poi c'è la pagina dei componenti nativi di InstantObjects e la relativa pagina di estensione degli stessi contenuti nella ISFLib.

N.B. Tutti i passaggi relativi all'installazione sono ben spiegati nel documento Installazione_ISF8.rtf fornito.

3.3. *L'uso di ISWorkbench 8 e il relativo repository*

L'installazione fornisce anche nel menu di Windows l'accesso a ISWorkbench per le versioni di Delphi supportate: ISFTemplate. ISWorkbench è solo la GUI che opera su un repository in formato XML presente nella cartella del progetto. Configurando il parametro esso si avvia direttamente aprendo il repository corretto.

Il funzionamento di ISWorkbench può variare a secondo del repository sul quale lo si fa lavorare: all'interno della cartella del repositorory c'è il file ISWorkbench.ini che definisce alcuni importanti parametri per il funzionamento di ISWorkbench. Riferirsi alla guida di ISWorkbench nel capitolo configurazione per ulteriori dettagli.

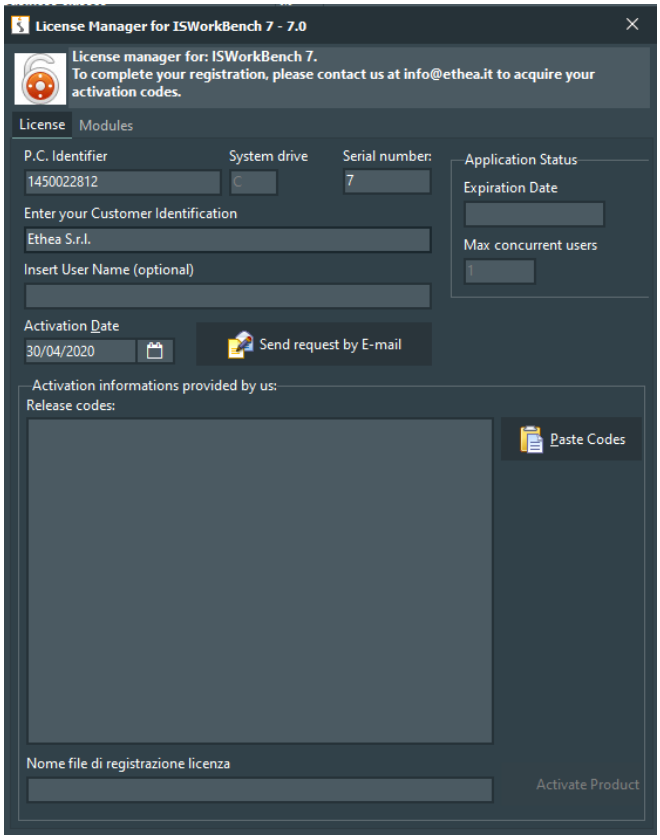


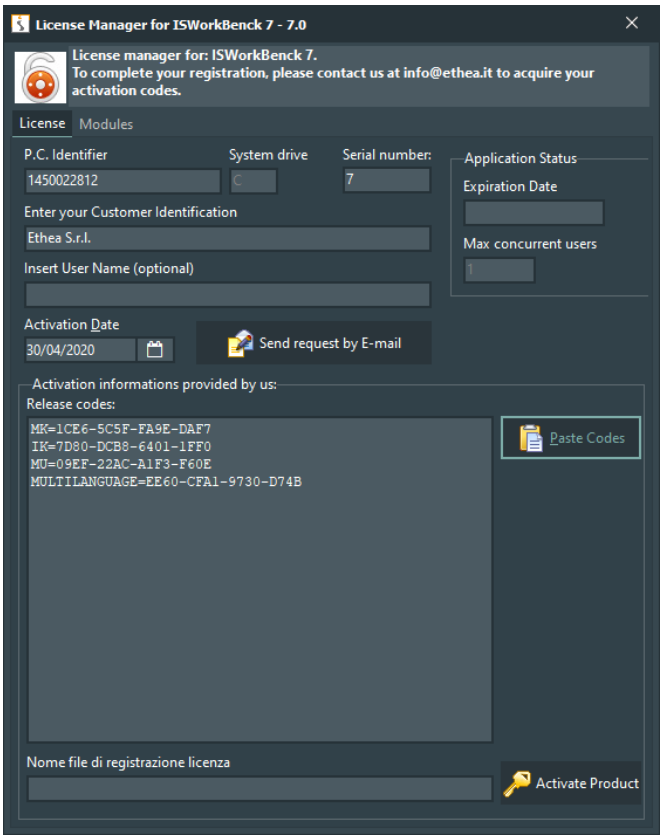
3.3.1. Registrazione ISWorkbench

Per utilizzare ISWorkbench è necessario registrare il prodotto, al primo utilizzo ci sarà questo messaggio, poi verrà proposta la mappa per la registrazione.

Inserire la denominazione della propria società, il nome utente opzionale e inviare la mail con i dati di registrazione ad info@ethea.it.

Nella mail di risposta ci saranno i codici per l'attivazione da copiare nel riquadro in basso, e cliccare su "Activare Product" per l'attivazione.





3.4. Markdown come sistema di Help

Ethea ha abbandonato l'uso di HelpScribble come sistema di generazione di help, sostituendolo con il più moderno sistema di Help basato sul formato MarkDown. Il visualizzatore dell'Help dell'applicazione è disponibile in un progetto Open-Source di Ethea: https://github.com/EtheaDev/MarkdownHelpViewer.

In questo progetto è presente un comodo Setup per installare il visualizzazione di Help in formato Markdown.

ISWorkBench genera la struttura di base dei file di Help di una applicazione ISF.

3.5. Editor per l'Help in formato MarkDown

In un altro progetto OpenSource (https://github.com/EtheaDev/MarkdownShellExtensions) è disponibile un comodo editor che permette facilmente di editare le pagine di Help in formato MarkDown.

3.5.1. MarkDown Help Viewer (nuovo formato)

Ethea ha sviluppato un nuovo meccanismo di visualizzazione dell'Help basato sul files markdown esportati da ISWorlbench.

Per utilizzare il nuovo sistema occorre modificare il file di progetto mettendo al posto della unit ISHTMLHelpviewer la riga:

```
MarkDownHelpViewer in '..\..\..\ext\MarkDownHelpViewer\Source\AppInterface\MarkDownHelpViewer.pas',
```

In questo modo si utilizza la unit della nuova interfaccia e la form di visualizzazione dell'Help basata sul componente MarkDownHelpViewer.

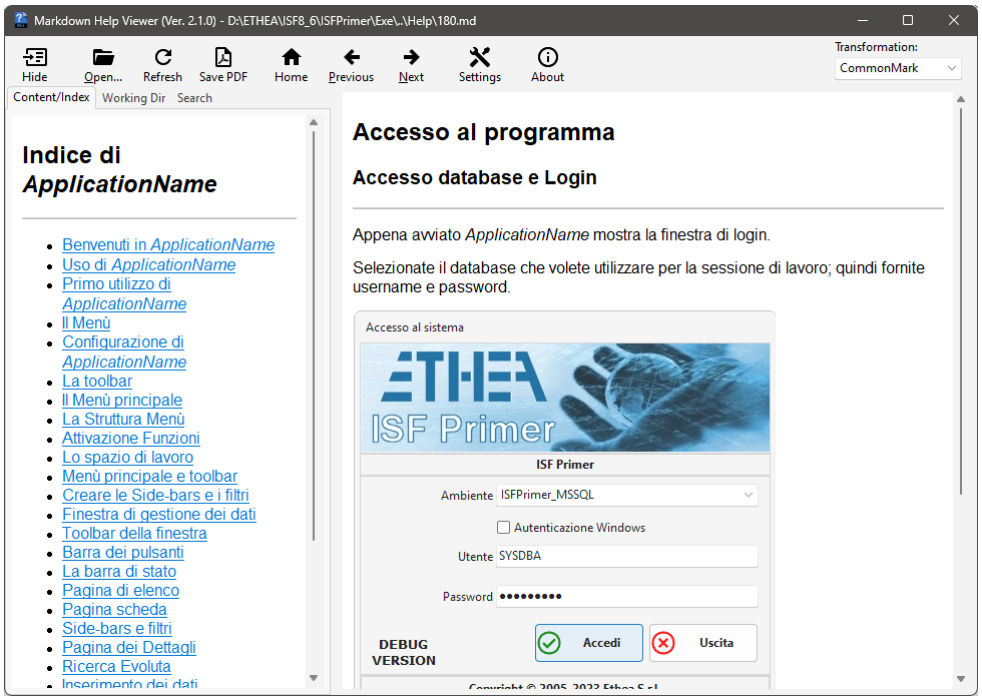
Per fare in modo che l'applicazione carichi il file giusto, occorre impostare il nome del file di help all'interno del file .ini in questo modo:

HelpFile={app}\..\Help\ISFPrimerHelp.md

Per testare il funzionamento dell'help, premendo F1 l'utente viene dirottato sulla pagina specifica della classe/attributo che sta editando o gestendo.

La finestra di help si ricorda dove è stata spostata/posizionata l'ultima volta, anche dopo aver chiuso l'applicazione.

In questa immagine un esempio del markdownhelpviewer invocato dalla finestra di Login:



3.6. Test dell'ambiente di sviluppo

Per verificare se abbiamo installato correttamente l'ambiente proviamo a compilare ed avviare il progetto di esempio ISFPrimer che si trovano nelle cartelle ISFPrimer\Projects\DXXX (XXX è la versione di Delphi).

4. Cenni sulle librerie CBLib (ISRtl, ISVcl) e ISFLib

Uno degli elementi costitutivi di InstantSolutions sono i componenti della libreria CBLib.

ISF è l'insieme di InstantObjects, CBLib e ISFLib

Vediamo alcune caratteristiche della CBLib, che è una libreria di componenti Delphi sviluppata da Ethea, della quale avete tutti i sorgenti.

4.1. I componenti della libreria

La libreria CBLib (costituita da una parte RTL (ISRTL) e una parte VCL (ISVCL), contiene moltissimi componenti, che possono essere utilizzati per le proprie applicazioni InstantSolutions, ma anche per applicazioni diverse.

In base all'impostazione di due variabili globali (CbFlatControls e CBNoBorderControls) è possibile cambiare aspetto a tutti i componenti di una applicazione (per renderla più moderna).

Vediamo una carrellata molto veloce dei componenti.

4.1.1. Componenti ISControls

Fanno parte di questa famiglia tutti i componenti visuali che sono stati customizzati per avere alcune caratteristiche molto comode:

- Tutti i componenti di "editing" hanno una proprietà "EditCaption" che è una label ancorata al componente di editing che facilita la costruzione di "form".
- Le trasparenze sono gestite in modo tale da funzionare correttamente con Windows 10/11 e i temi abilitati.
- Alcuni componenti di editing non presenti in Delphi (editing di numeri da destra, di date/ore con calendario, con un pulsante associato all'edit.
- Un componente "dataaware" che mostra i dati di un dataset in una DbListView (TCBXDbListView)
- Un componente "Data diffing" per la gestione dei confronti e i merge tra dati di due dataset (TCBDataDiffing).

Ecco la lista dei componenti dispobili nella palette "ISControls":

```
TCBChart, TCBXActionList, TCBXBitBtn, TCBXButton, TCBXButtonEdit, TCBXCheckBox, TCBXCheckListBox,
TCBXColorBox, TCBXColorGrid, TCBXComboBox, TCBXControlBar, TCBXCurrencyEdit, TCBXEdit, TCBXFormResize,
TCBXGroupBox, TCBXImage, TCBXImageList, TCBXLabel, TCBXListBox, TCBXListView, TCBXMainMenu, TCBXMaskEdit,
TCBXMemo, TCBXPageControl, TCBXPanel, TCBXPopUpMenu, TCBXRadioButton, TCBXRadioGroup, TCBXRichEdit,
TCBXScrollBox, TCBXSpeedButton, TCBXSpinEdit, TCBXSplitter, TCBXStringGrid, TCBXTabControl,
TCBXThinTrackBar, TCBXTitle, TCBXTreeView, TCBXDBListView, TCBDataDiffing, TCBClock
```

4.1.2. TCBXBitBtn

Per motivi di "rendering" errato delle immagini sui TBitBtn di Delphi, il componente TCBXBitBtn di fatto eredita da TButton, in modo tale da poter visualizzare correttamente le icone IconFonts o SVGIcon disegnate con il canale Alpha (trasparente) tramite GDI+.

4.1.3. Componenti ISDataAccess

Tra i componenti di accesso ai dati spicca il TCBClientDataSet, che ha alcune feature per funzionare direttamente su file, salvando i dati in formato XML "formattato, ordinato e indentato), per facilitarne l'utilizzo con sistemi di versioning. Inoltre ci sono una serie di componenti utili per l'esportazione e l'importazione dei dati in vari formati (Excel, CSV, Text, XML).

Ecco la lista dei componenti dispobili nella palette "ISDataAccess":

```
TCBDatasetToExcel, TCBDatasetToFlexcel, TCBClientDataSet, TCBDataSource, TCBDatasetToCSV, TCBDatasetToText,
TCBDatasetToXML, TCBDatasetFromExcel, TCBDataSetFromFlexCel, TCBDatasetFromCSV, TCBDatasetFromText,
TCBDatasetFromXML, TCBDatasetFromExcel, TCBADOQuery, TCBADOTable, TCBADODataSet
```

4.1.4. Componenti ISDataControls

I componenti "dataaware" della CBLib forniscono le stesse feature dei componenti di editing (in particolare la EditCaption e le modalità di input dei dati specializzati, per importi, numeri, date/ore, ecc...).

In particolare il componente TCBXDbImageLink, consente l'input di una stringa "linkata" ad una immagine.

Tutti questi componenti DataAware sono poi "sfruttati" da uno dei componenti più utilizzati negli applicativi InstantSolutions, vale a dire il **TCBXDbMultiEdit**: questo componente (come vedremo più avanti) consente una gestione molto semplificata ma allo stesso tempo molto potente di una mappa di dataentry, utilizzando il concetto di "Custom Layout".

Ecco la lista dei componenti dispobili nella palette "ISDataControls":

```
TCBDbChart, TCBXDBButtonEdit, TCBXDBCheckBox, TCBXDBComboBox, TCBXDBCurrencyEdit, TCBXDBEdit,
TCBXDBExtendedMemo, TCBXDbGrid, TCBXDBImage, TCBXDbImageLink, TCBXDBLabel, TCBXDBListBox,
TCBXDBLookupComboBox, TCBXDBLookupListBox, TCBXDBMemo, TCBXDBMultiEdit, TCBXDBNavigator, TCBXDBRadioGroup,
TCBXDBText, TCBDbBlob, TCBDBMultiValue, TCBDBReferenceButtons, TCBDBMultiValue, TCBDBNumberBox, TCBNumberBox
```

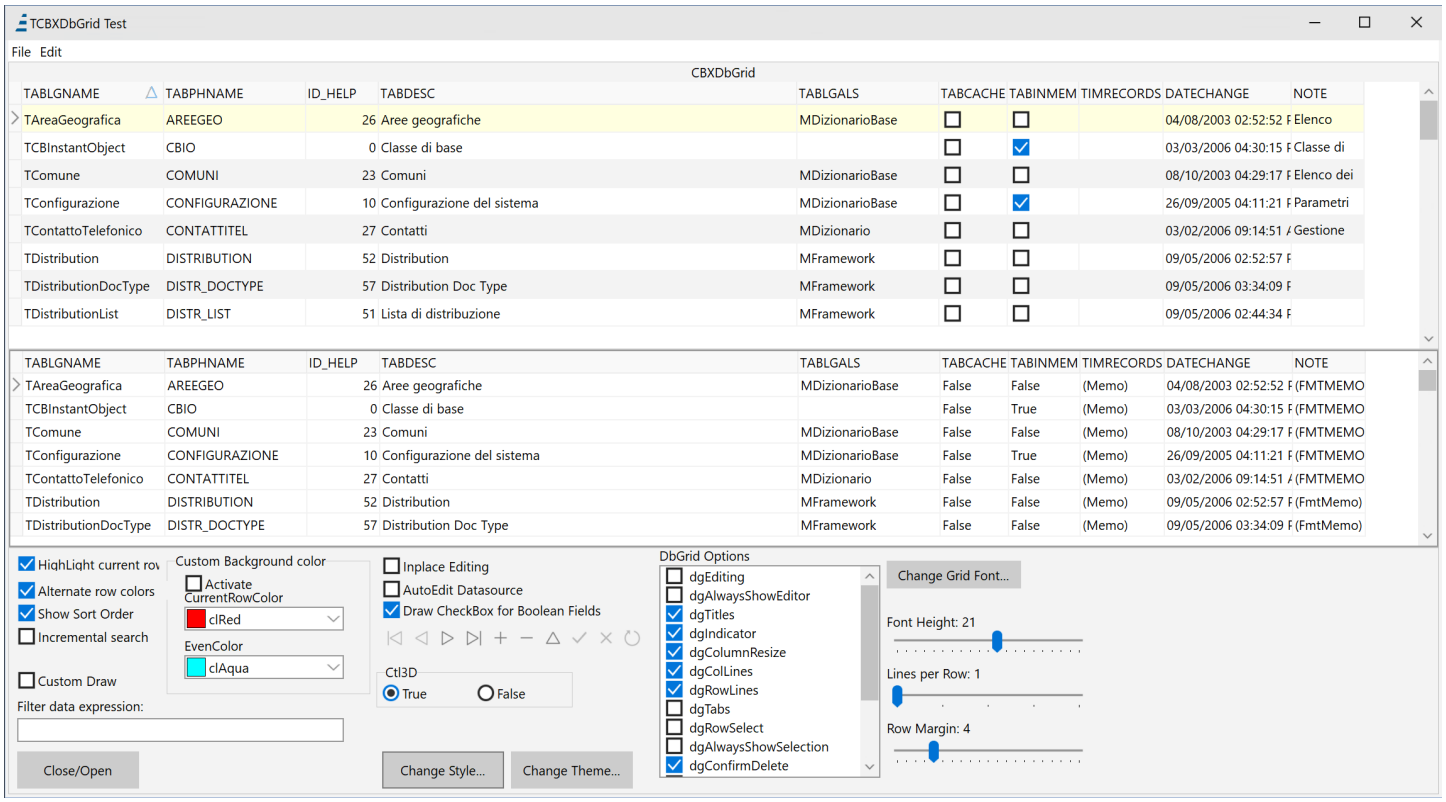
4.1.5. Componenti di editing con BoundCaption

Tutti i componenti di editing della CBLib hanno la caratteristica di avere una **"BoundCaption"** (TCBXBoundLabel) ovvero una label agganciata al componente di editing in una posizione "intorno" al componente, tra quelle disponibili: lpTopLeft (il default), lpTopCenter, lpTopRight, lpBottomLeft, lpBottomCenter, lpBottomRight, lpLeftTop, lpLeftMiddle, lpLeftBottom, lpRightTop, lpRightMiddle, lpRightBottom. Questo permette di mantenere sempre agganciata e allineata la label della "caption" del componente, anche se il componente viene spostato.

4.1.6. La nuova TCBXDbGrid

Oltre a fornire lo stesso concetto di "Custom Layout", la nuova **TCBXDbGrid**, fornisce anche una serie di funzionalità molto avanzate, come la possibilità di ordinare i dati, di fare ricerche incrementali, applicare colori custom, alle righe, ecc…

Da ISF 7.6 è ora possibile anche controllare l'altezza di default delle righe, definendo dei margini, oppure definire che in ogni "riga di record" si devono mostrare "più righe di dati", quando ad esempio si vuole mostrare il contenuto di un campo "Memo": ecco un esempio di TCBXDBGrid con margini e 3 righe per ogni record:



4.1.7. Componenti ISOpenOffice

Sono componenti per l'interfacciamento con OpenOffice. Il più completo è il TOOoWriter, che insieme al TOOoDocProducer consente di utilizzare OpenOffice come editor e generatore di Report.

Ecco la lista dei componenti dispobili nella palette "ISOpenOffice":

TOOoDocument, TOOoWriter, TOOoCalc, TOOoDraw, TOOoImpress.

4.1.8. Componenti ISDocProducer

Sono componenti, tra loro "compatibili", per la gestione, l'editing e la generazione di documenti in vari formati: Openoffice, ReportBuilder, HTML, FO (Formatting Object), XML, ecc…

Dalla versione 7.6 di ISF è stato aggiunto anche un nuovo "DocProducer" per FlexcelReport: fare riferimento al documento **"Utilizzo di Flexcel Report con ISF.pdf"** per maggiori informazioni.

Ecco la lista dei componenti dispobili nella palette "ISDocProducer":

TCBRBuilderDocProducer, TCBHtmlDocProducer, TCBFopDocProducer, TCBOooWriterDocProducer, TCBXMLDocProducer, TCBFlexcelDocProducer.

4.1.9. Componenti ISFireDAC

Nelle precedenti versioni di ISF era stato dato supporto a DBEpress, nella nuova versione sono stati riportati dei componenti con le stesse funzionalità precedenti ma sostituite da FireDAC, vengono forniti alcuni componenti ad-hoc, in particolare TCBFDSimpleDataSet, che consente di fare "caching" lato client di un dataset unidirezionale, e

TCBFDMetadata, che consente di estrarre i metadati dai DB in modo semplice.

Ecco la lista dei componenti dispobili nella palette “ISFireDAC”:

```
TCBFDMetadata, TCBFDConnection, TCBFDDataSet, TCBFDQuery, TCBFDStoredProc, TCBFDTable,  TCBFDSimpleDataSet.
```

4.2. ISFLib

4.2.1. Componenti ISF di InstantObjects

Nella libreria ISF ci sono alcuni componenti utilizzati direttamente dal framework InstantSlutions, la maggior parte dei quali sono estensioni dei componenti InstantObjects nativi per poter accedere ai dati contenuti nel dizionario delle classi gestito con ISWorkbench.

Ecco la lista dei componenti non visuali:

```
TCBInstantExposer, TCBInstantSelector, TCBInstantClientDataSet,
TCBXMLFileAccessor, TCBJSONFileAccessor,
```

Ecco la lista dei componenti visuali, dispobili nella palette “ISF”:

```
TCBInstantExplorer
```

4.2.2. Componente TCBInstantClientDataSet per form native

Per ottimizzare le prestazioni con InstantObjects è stato sviluppato un dataset “speciale” che deriva da TClientDataSet ma che è in grado di rimanere sincronizzato con un oggetto InstantObject, sempre allineato al record corrente del dataset.

Questo componente viene sfruttato all’interno delle “form native”.

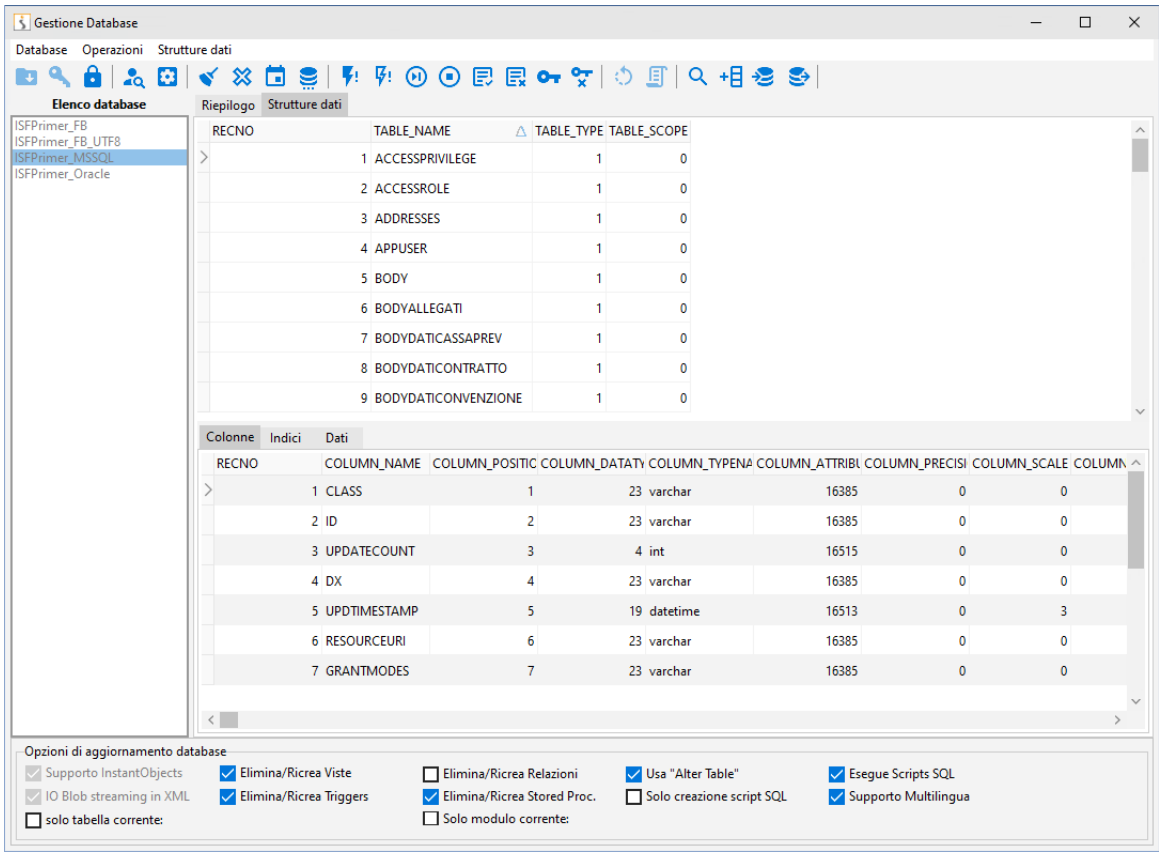
4.2.3. Componente TCBInstantExplorer

Il componente visuale TCBInstantExplorer è un “explorer” molto evoluto rispetto a quello standard di InstantObjects, perché è in grado di ospitare al suo interno sia una TCBXDBGrid (quando deve mostrare una griglia), sia un TCBXDBMultiEdit (quando deve mostrare una maschera di dati).

4.3. Editor evoluti della CBLib

Oltre ai vari componenti esistono anche degli “editor evoluti”, in relazione al funzionamento di alcuni componenti:

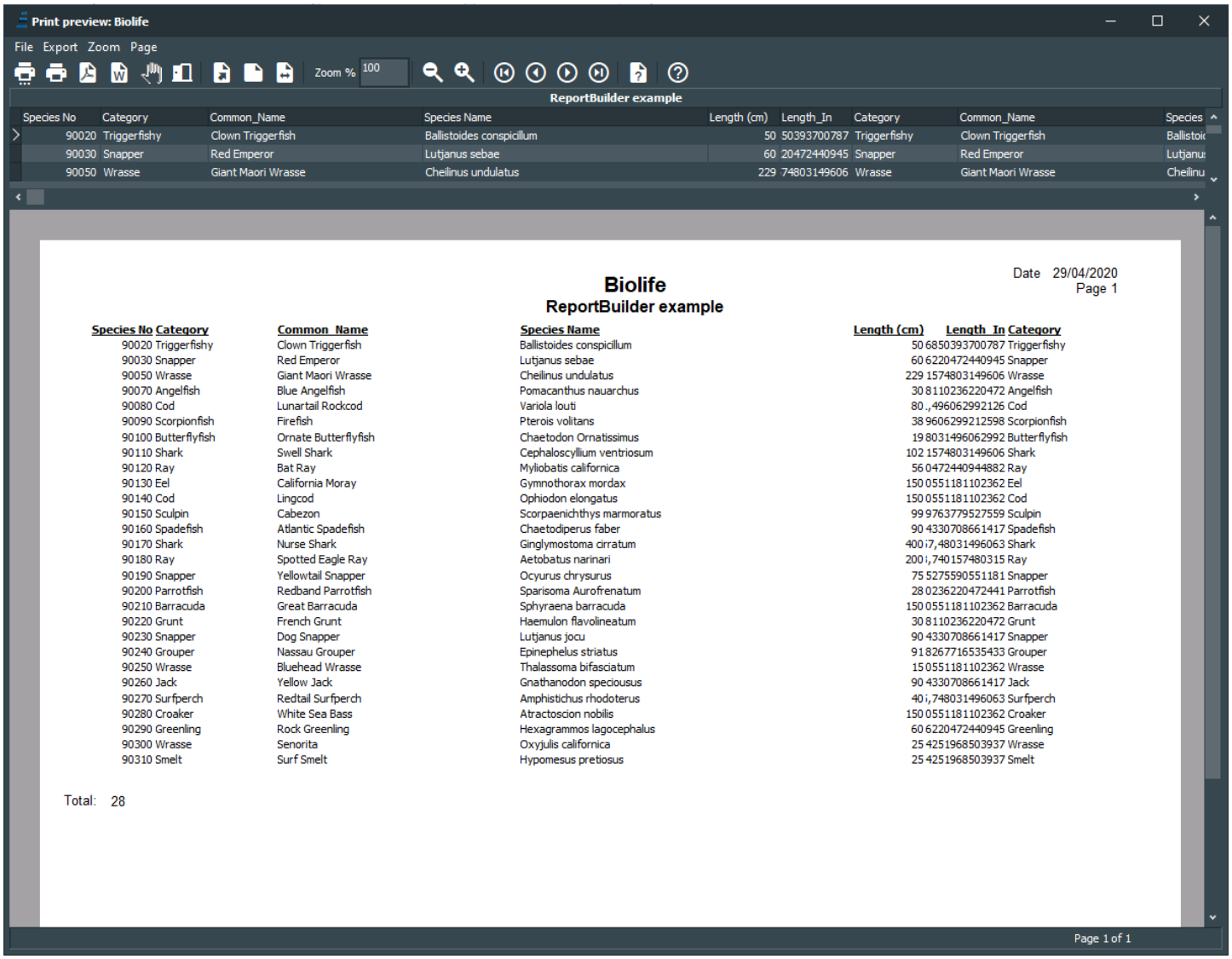
4.3.1. Ethea Database Manager



È una finestra di gestione di fonti di dati “FireDAC”: tramite un file di configurazione è possibile accedere a diversi database, vederne la struttura, ed all'interno di un “editor SQL” è possibile eseguire query di select, insert, update o delete, anche in modalità scripting.

4.3.2. Ethea ReportBuilder Preview (e Table Preview)

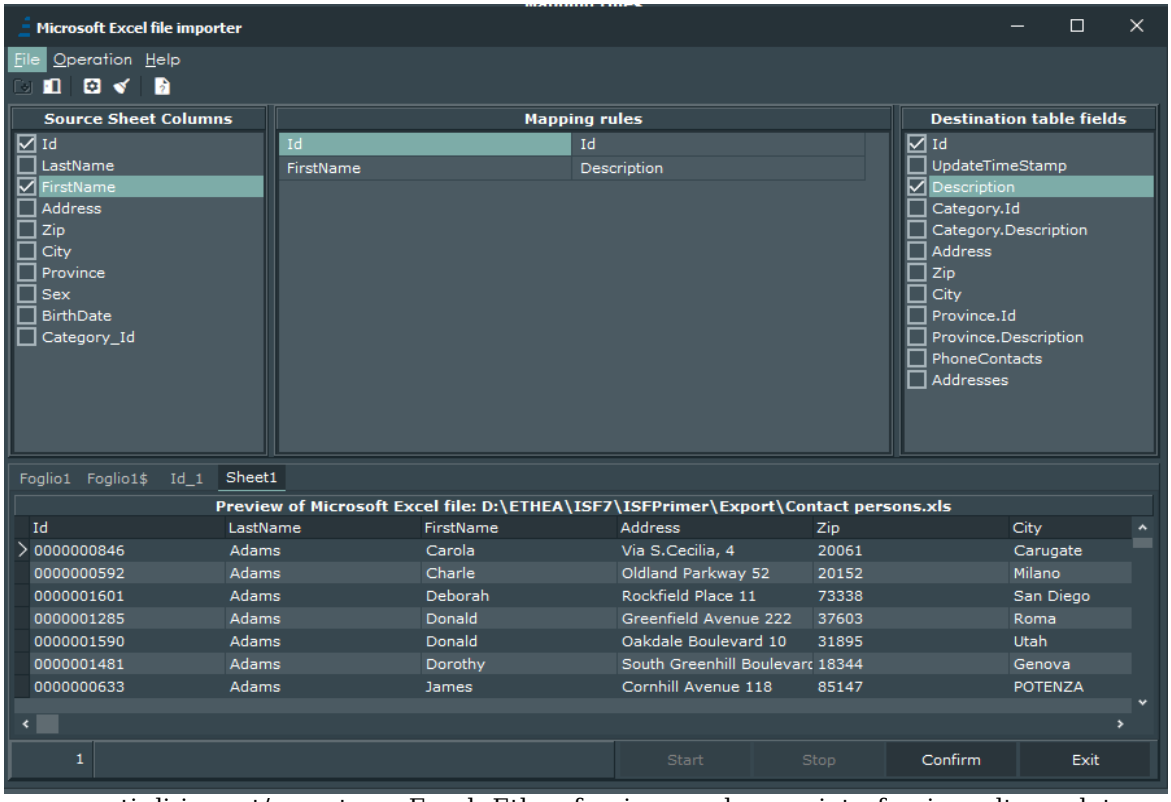
Ethea consiglia ReportBuilder come generatore di report, e fornisce anche un supporto avanzato per questo generatore di report, come il preview customizzato (quello standard di RB è molto limitato), ma soprattutto un preview custom per generare report “tabulari” in automatico, da parte dell'utente: basta fornire il dataset e fa tutto da solo: è possibile spostare colonne, definirne la larghezza, per ottenere il layout di stampa desiderato, tutto “WYSIWYG”. E' possibile anche esportare il report in PDF.



Species No	Category	Common_Name	Species Name	Length (cm)	Length_In	Category
90020	Triggerfishy	Clown Triggerfish	Ballistoides conspicillum	50	6850393700787	Triggerfishy
90030	Snapper	Red Emperor	Lutjanus sebae	60	6220472440945	Snapper
90050	Wrasse	Giant Maori Wrasse	Cheilinus undulatus	229	1574803149606	Wrasse
90070	Angelfish	Blue Angelfish	Pomacanthus nauarchus	30	8110236220472	Angelfish
90080	Cod	Lunartail Rockcod	Variola louti	80	.,496062992126	Cod
90090	Scorpionfish	Firefish	Pterois volitans	38	9606299212598	Scorpionfish
90100	Butterflyfish	Ornate Butterflyfish	Chaetodon Ornatissimus	19	8031496062992	Butterflyfish
90110	Shark	Swell Shark	Cephaloscyllium ventriosum	102	1574803149606	Shark
90120	Ray	Bat Ray	Myliobatis californica	56	0472440944882	Ray
90130	Eel	California Moray	Gymnothorax mordax	150	0551181102362	Eel
90140	Cod	Lingcod	Ophiodon elongatus	150	0551181102362	Cod
90150	Sculpin	Cabezon	Scorpaenichthys marmoratus	99	9763779527559	Sculpin
90160	Spadefish	Atlantic Spadefish	Chaetodiperus faber	90	4330708661417	Spadefish
90170	Shark	Nurse Shark	Ginglymostoma cirratum	400	i7,48031496063	Shark
90180	Ray	Spotted Eagle Ray	Aetobatus narinari	200	i,740157480315	Ray
90190	Snapper	Yellowtail Snapper	Ocyurus chrysurus	75	5275590551181	Snapper
90200	Parrotfish	Redband Parrotfish	Sparisoma Aurofrenatum	28	0236220472441	Parrotfish
90210	Barracuda	Great Barracuda	Sphyraena barracuda	150	0551181102362	Barracuda
90220	Grunt	French Grunt	Haemulon flavolineatum	30	8110236220472	Grunt
90230	Snapper	Dog Snapper	Lutjanus jocu	90	4330708661417	Snapper
90240	Grouper	Nassau Grouper	Epinephelus striatus	91	8267716535433	Grouper
90250	Wrasse	Bluehead Wrasse	Thalassoma bifasciatum	15	0551181102362	Wrasse
90260	Jack	Yellow Jack	Gnathanodon speciousus	90	4330708661417	Jack
90270	Surfperch	Redtail Surfperch	Amphistichus rhodoterus	40	i,748031496063	Surfperch
90280	Croaker	White Sea Bass	Atractoscion nobilis	150	0551181102362	Croaker
90290	Greenling	Rock Greenling	Hexagrammos lagocephalus	60	6220472440945	Greenling
90300	Wrasse	Senorita	Oxyjulis californica	25	4251968503937	Wrasse
90310	Smelt	Surf Smelt	Hypomesus pretiosus	25	4251968503937	Smelt

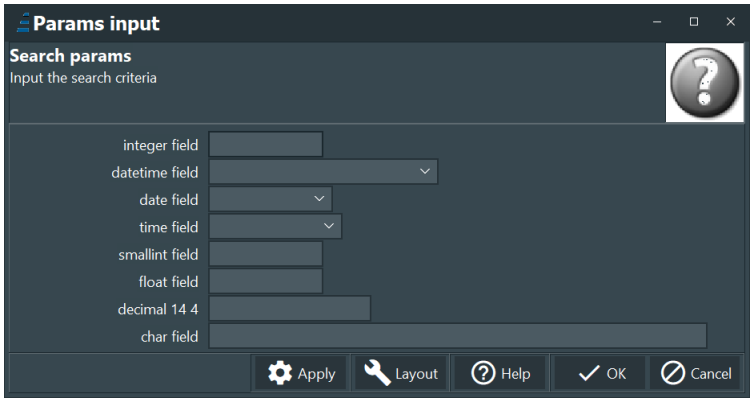
Total: 28

4.3.3. Ethea Excel Import Mapping Tool



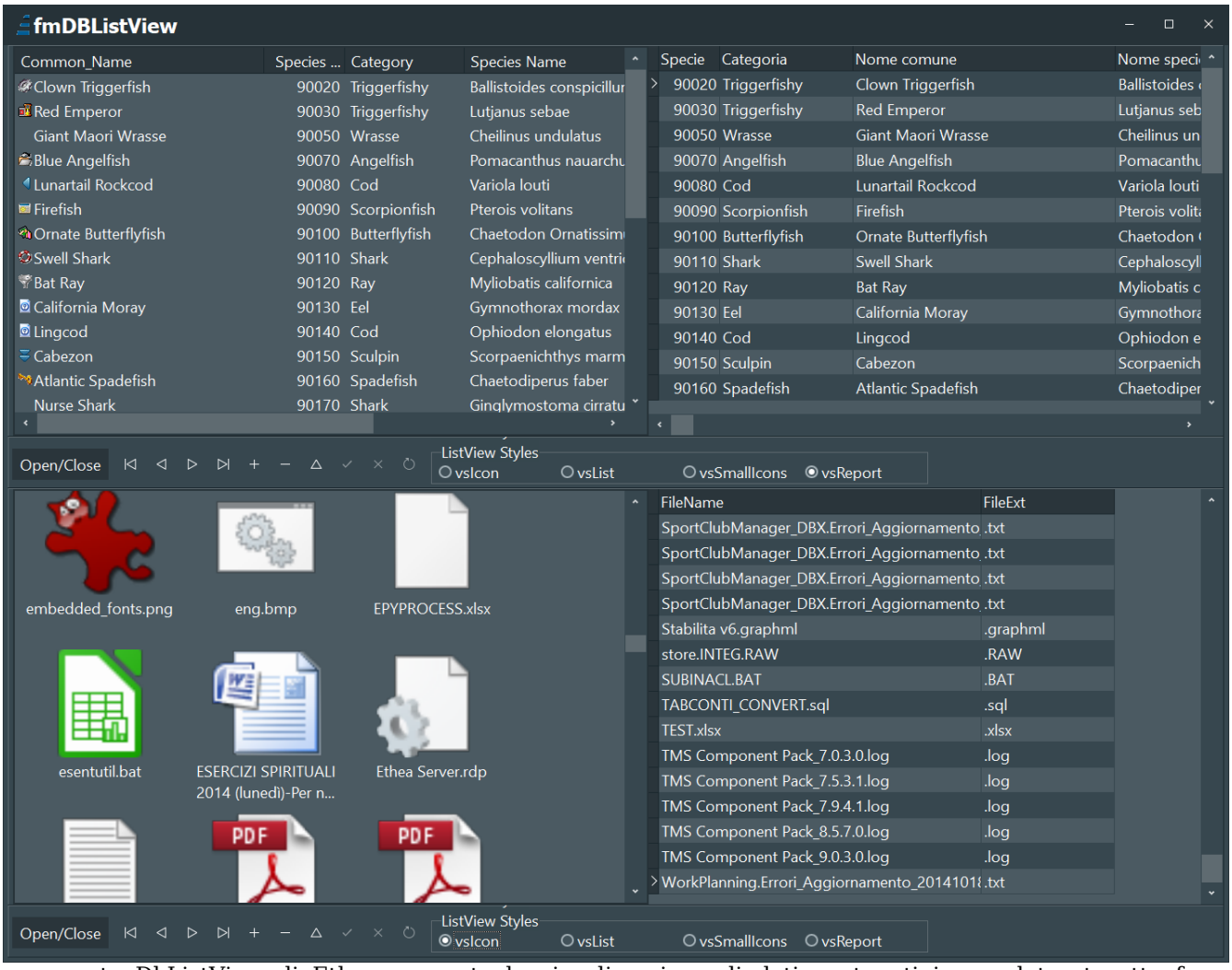
Oltre ai componenti di import/export per Excel, Ethea fornisce anche una interfaccia molto evoluta per importare i dati contenuti in un foglio Excel: selezionando il “range” e definendo la “mappatura” dei campi è molto semplice importare i dati all'interno di un Dataset.

4.3.4. Ethea Param input form



Esiste anche la possibilità di fornire facilmente all'utente una finestra per l'input dei parametri di una query SQL, tramite questo editor: l'input è definito dal tipo di parametri, quindi sfrutta i componenti CBLib, oltre alla possibilità di definire un layout “custom” per ogni serie di parametri.

4.3.5. DbListView: un modo “alternativo” alla classica DbGrid.



Il componente DbListView di Ethea consente la visualizzazione di dati contenuti in un dataset sotto forma di ListView, con le varie modalità: icone, piccole e grandi, list o report. In questa applicazione di esempio vediamo come funziona. E' sufficiente fornire il dataset alla proprietà DataSource (come per un normale componente dataaware), una imagelist di icone e un evento per associare il valore contenuto in un record all'icona e il gioco è fatto.

4.3.6. Ethea CBDataDiffing editor

Il componete TCBDataDiffing, consente il confronto tra dati di dataset diversi, con la possibilità di fornire una interfaccia utente molto intuitiva per decidere quali sono i dati corretti (così come avviene per un sistema di diffing di files di testo).

L'utilizzo del componente è molto semplice, basta impostare:

DataSetA e **DataSetB**: sono i due dataset da confrontare

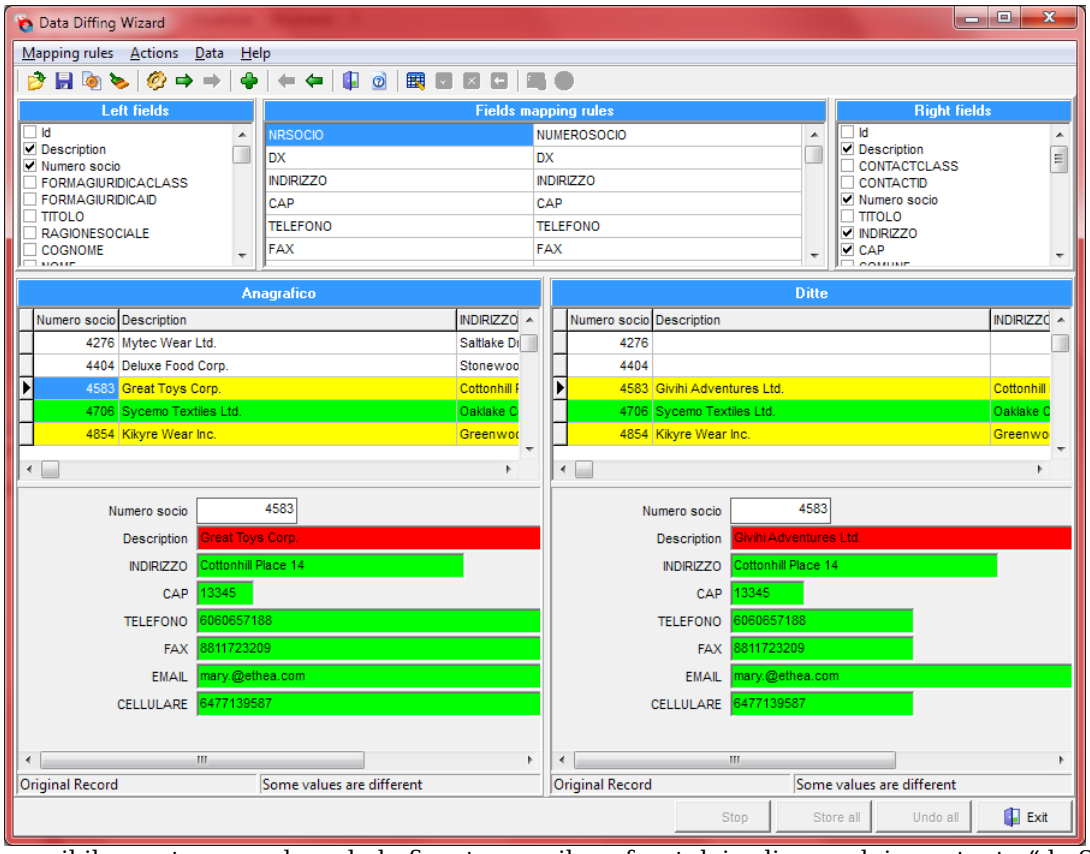
DataSetAKeyFields e **DataSetBKeyFields**: sono i campi “chiave” che hanno corrispondenza tra i 2 dataset

DiffingMode: “dmAllRecords” oppure “dmSingleRecord”, per confronto di tutto il DatasetA o solo di un record.

Poi occorre fornire un metodo di “ricerca” del record nel DataSetB nell'evento: **RecordFinder**.

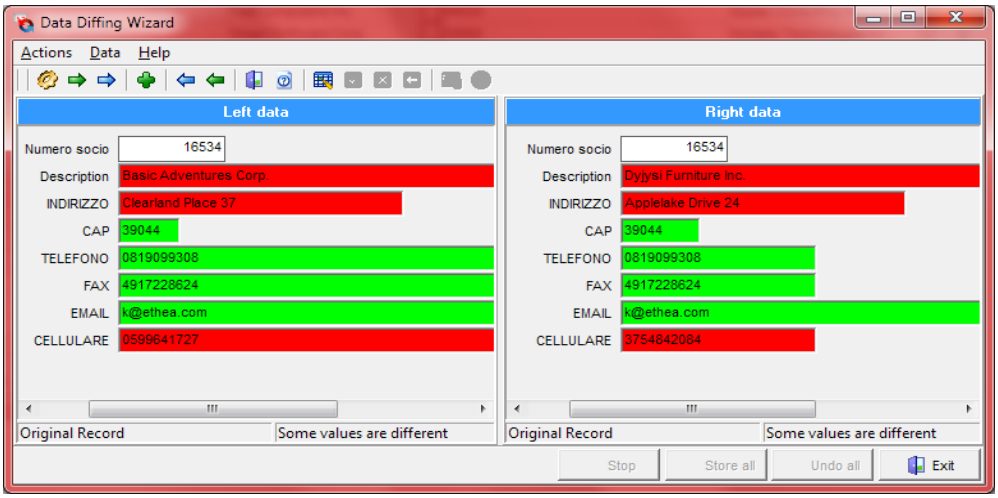
Infine occorre definire la “mappatura” dei campi da confrontare nella proprietà **FieldMapping**, ma è anche possibile farlo “run-time” se si imposta **MappingRules** a “mrShowEditable” (di default la mappatura non è visibile).

A questo punto basta invocare il metodo **ShowDiffViewer** per mostrare l'editor di diffing: in figura si può vedere l'editor in modalità “dmAllRecords” con le MappingRules visibili e modificabili: è possibile caricarle o salvarle in un file di testo .map utilizzando il viewer, ma il componente fornisce anche i metodi LoadMappingFromFile e SaveMappingToFile per gestire la mappatura a run-time, per non mostrare all'utente finale le regole di mappatura.



E' ovviamente possibile mostrare anche solo la finestra per il confront dei soli record, impostanto “dmSingleRecord”, ed è possibile anche nascondere le “regole di mappatura”.

L'utente effettua le operazioni con il menu contestuale, e memorizza i dati premendo “store all”.



5. InstantObjects: il cuore di InstantSolutions

5.1. Conoscere InstantObjects per usare al meglio InstantSolutions

InstantSolutions è un framework avanzato di sviluppo di applicazioni basato sulla libreria di componenti InstantObjects. Questa libreria open-source (https://github.com/EtheaDev/InstantObjects) permette alle applicazioni Delphi di programmare attraverso classi invece che attraverso DataSet. Occorre conoscere almeno le basi del funzionamento di InstantObjects per poter utilizzare al meglio InstantSolutions, anche se InstantSolutions "nasconde" diversi aspetti della programmazione tipica con InstantObjects.

Innanzitutto ISWorkbench racchiude tutte le funzionalità (e molte altre) per la definizione delle classi, evitando di utilizzare l'editor di InstantObjects integrato in Delphi (che è molto più povero e limitato di ISWorkbench).

E' importante almeno conoscere gli aspetti essenziali di InstantObjects quali:

- Il **Connector**: un componente in gradi accedere ad una "sorgente" di oggetti InstantObjects (tipicamente tali oggetti stanno su un Database, ma nel caso di ISF stanno anche su file XML)
- Il **Broker**: è una parte importante di InstantObjects perché si occupa di "adattare" l'accesso tramite FireDAC: con ISF8 gli unici 2 broker utilizzati sono quello per FireDAC e quello XML per i file di configurazione.
- Il **Selector** e l'**Exposer**: sono i "dataset" di InstantObjects che servono esclusivamente per agganciare i valori contenuti in un oggetto all'interfaccia utente realizzata con i componenti DataAware.
- La classe **TInstantQuery**, per fare interrogazioni tramite il linguaggio "IQL" (Instant Query Language) e recuperare una lista di oggetti da una sorgente (Connector). Es. Il selector contiene al suo interno una TInstantQuery.
- Il metodo **TInstantObjectClass.Retrieve**: è un "costruttore" dell'oggetto. Serve per "recuperare" un oggetto di una certa classe a partire dal suo Id (occorre fare attenzione al meccanismo di reference-counting e ricordarsi di fare la .free dell'oggetto quando si è terminato di utilizzarlo).
- Il metodo **TInstantObjectClass.Store**: serve per "memorizzare" un oggetto dentro il suo "Storage" (Database o file)
- Il metodo **TInstantObjectClass.Dispose**: serve per "cancellare" un oggetto dal suo "Storage" (Database o file)

Rimandiamo a tutta la documentazione di InstantObjects per conoscere nel dettaglio questi aspetti di InstantObjects. Seguendo questo Tutorial alcuni di questi aspetti vengono dati per scontati: recuperare la documentazione necessaria se non si comprendono alcuni passaggi.

5.1.1. Il linguaggio IQL e i filtri su valori di attributi di "dettaglio"

ISF utilizza il linguaggio di query di InstantObjects denominato IQL (Instant Query Language).

La sintassi del linguaggio è abbastanza semplice, ad esempio:

```
SELECT * FROM ANY TContact
WHERE Address LIKE :AddressParam
ORDER BY Description
```

Con una query di questo tipo (lanciata attraverso un oggetto TInstantQuery) è possibile recuperare tutti gli oggetti di tipo TContact (anche quelli derivati grazie alla clausola ANY) che abbiano un certo indirizzo (con passaggio di parametri) ordinati per Descrizione.

Tuttavia tale linguaggio presenta delle limitazioni, specialmente quando occorre cercare degli oggetti in base al valore contenuto in qualcuno dei suoi dettegli.

Per risolvere questo problema con ISF, prima della versione 4.9, era possibile farlo effettuando interrogazioni con la **clausola IN**, rinunciando però ad alcune feature di IQL, come la traduzione automatica dei nomi degli attributi di una classe nel relativo storagename.

Se ad esempio si volevano cercare tutti i contatti della rubrica che abbiano un indirizzo a "Los...", si poteva utilizzare questa sintassi:

Query 1

```
SELECT * FROM TISContactPerson
WHERE ID IN
[(SELECT CONTACTID FROM ADDRESSES WHERE CITY LIKE 'Los%')]
```

Da notare che la parte contenuta dentro le parentesi quadre viene ignorata da IQL e viene passata direttamente al server SQL, pertanto è necessario utilizzare i nomi "storage" e non i nomi logici della classe o dell'attributo.

Questa interrogazione viene così tradotta:

```
SELECT t1.Class AS Class, t1.Id AS Id FROM PERSONS t1
WHERE (t1.Class = 'TISContactPerson') AND (t1.Id IN
( SELECT CONTACTID FROM ADDRESSES WHERE CITY LIKE 'Los%' ) )
```

Per definire una ricerca determinata dagli oggetti di un'altra classe che referenzia la nostra classe di ricerca è possibile utilizzare una subquery dove al posto dell'asterisco va indicato il campo reference.

Es. per estrarre tutte le regioni referenziate tramite la proprietà Regione della classe TISProvincia la cui descrizione

inizia per B:

Query 2:

```
SELECT * FROM TISRegion
WHERE
Id IN [(SELECT REGIONID FROM PROVINCE WHERE DX LIKE 'B%')]
```

Questa query viene tradotta in questo modo:

```
SELECT t1.Class AS Class, t1.Id AS Id FROM REGION t1
WHERE (t1.Class = 'TISRegion') AND
(t1.Id IN  ( SELECT REGIONID FROM PROVINCE WHERE DX LIKE 'B%' ) )
```

5.1.2. L'estensione del linguaggio IQL: le clausole “EXISTS” e “USING”

Dalla versione 4.9 di ISF (corrispondente alla versione 2.2 di InstantObjects) è possibile fare lo stesso tipo di interrograzione utilizzando le nuove clausole Exists e Using, con il vantaggio che non bisogna più utilizzare i nomi “storage” le parentesi quadre, ed in generale è un modo più pulito e standard di fare la stessa cosa.

In pratica è possibile applicare la clausola EXISTS ad una sub-query ma occorre indicare anche qual'è il campo “reference” che nella classe di dettaglio ha il riferimento verso la classe principale.

Es. per effettuare la stessa query di prima basta scrivere:

Query 1:

```
SELECT * FROM TISContactPerson
WHERE EXISTS(SELECT * FROM TISAddress WHERE City LIKE 'Los%' USING Contact)
```

Questa query viene tradotta in questo modo:

```
SELECT t1.Class AS Class, t1.Id AS Id FROM PERSONS t1
WHERE (t1.Class = 'TISContactPerson') AND
(EXISTS(SELECT l1s1t1.Class AS Class, l1s1t1.Id AS Id FROM ADDRESSES l1s1t1 WHERE ((l1s1t1.Class = 'TISAddress') AND (l1s1t1.CITY LIKE 'Los%')) AND ((l1s1t1.CONTATTOClass = 'TISContactPerson') AND (l1s1t1.CONTATTOId = t1.Id))))
```

Allo stesso modo, la seconda query va scritta così:

Query 2:

```
SELECT * FROM TISRegion
WHERE EXISTS (SELECT * FROM TISProvince WHERE Description LIKE 'B%' USING Region)
```

Questa query viene tradotta in questo modo:

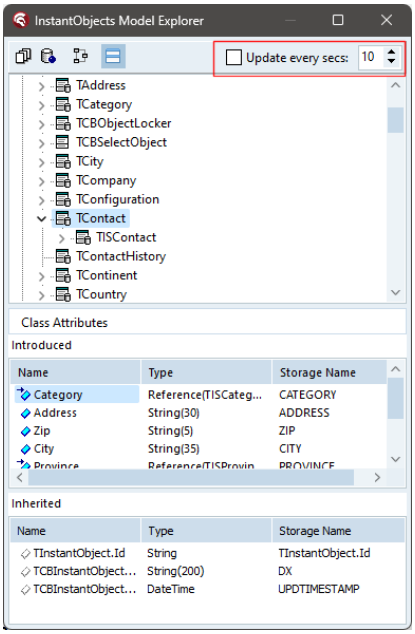
```
SELECT t1.Class AS Class, t1.Id AS Id FROM REGION t1
WHERE (t1.Class = 'TISRegion') AND (EXISTS(SELECT l1s1t1.Class AS Class, l1s1t1.Id AS Id FROM PROVINCE l1s1t1 WHERE ((l1s1t1.Class = 'TISProvince') AND (l1s1t1.DX LIKE 'B%')) AND ((l1s1t1.REGIONClass = 'TISRegion') AND (l1s1t1.REGIONId = t1.Id))))
```

Da notare che la traduzione della query ora è molto più performante perché la JOIN viene fatta con Class e Id, che è la Primary Key della tabella, mentre prima la JOIN avveniva solo per Id.

Potete applicare questi concetti nei rispettivi capitoli che riguardano la gestione dei “filtri” di ricerca e del tuning dell'applicazione.

5.1.3. L’editor delle classi integrato nell’IDE di Delphi

Quando si installano i package di InstantObjects, oltre ai componenti vengono installati una serie di editor per le classi, disponibile nella voce di menu: View/InstantObjects Model Exploter (vedi figura).



Dalla versione 8.5.3 è stato introdotto un meccanismo per migliorare le prestazioni dell’IDE quando si utilizza InstantObjects con modelli di dati molto grandi. In precedenza, il model “in memoria” veniva aggiornato ogni mezzo secondo e siccome era una operazione onerosa per certi progetti, questo rallentava lo sviluppo dell’applicazione.

Dato che questo meccanismo “automatico” era pensato per lo sviluppo delle classi direttamente con questo strumento oppure modificando direttamente il codice nelle unit di modello, era necessario tenere sempre allineato il modello in memoria e i sorgenti.

Siccome con ISF la definizione delle classi avviene all’interno di ISWorkBench (come vedremo più avanti) questa “feature” non è più necessaria, perciò di default l’aggiornamento non è più attivo (vedi il checkbox non abilitato nel riquadro rosso).

Nel momento in cui è necessario riattivarlo è possibile farlo abilitando il checkbox e definendo il ritardo (esperro in secondi) nella casella a fianco: non ha senso richiedere un aggiornamento del model in memoria per un intervallo inferiore al secondo: di default è stato impostato con 10 secondi.

N.B. anche se il model non viene aggiornato in tempo reale, nel momento in cui si procede alla compilazione del progetto viene calcolato in quel momento, per poter generare il file .mdr necessario alla compilazione.

6. InstaltSolutions on the road con il MultiFramework

6.1. Premessa: il progetto Primer

Per valutare al meglio InstantSolutions è il momento della prova su strada, partendo da una semplice (ma non banale) applicazione, che nel suo piccolo è in grado di mostrare tutte le caratteristiche della programmazione con un OPF.

Già nel framework InstantObjects esiste una famosa applicazione demo dal nome “Primer” che mostra l'uso di InstantObjects. (la potete trovare sotto {ISF_Install_Dir}\InstantObjects\Demos\PrimerCross, che non è da confondere con ISFPrimer.

Vogliamo mostrare come una applicazione simile si può sviluppare in modo ancora più facile ed evoluto con InstantSolutions.

Si tratta di una applicazione per la gestione di una rubrica di contatti. Ogni contatto può essere una persona o un società, quindi dovrà esistere una classe di contatti (TContact) dalla quale derivare la classe delle persone (TContactPerson) e quella delle società (TContactCompany).

6.2. Utilizzo del MultiFramework

N.B. Dentro ISF c'è già una cartella ISFPrimer che contiene il risultato finito di questo corso, e può essere utile per confrontare il lavoro che si sta facendo con quello corretto.

Iniziamo l'applicazione partendo dal “MultiFramework Template” (che è quello più evoluto che usa un sistema a più finestre con sessioni aperte) per la nostra applicazione di prova, copiando l'intera cartella \ISFTemplate\ del template nella cartella del nostro progetto che chiameremo: MyISFPrimer. Da notare che un progetto ISF deve “stare” in un certo specifico punto rispetto al framework stesso, per via dei riferimenti alle cartelle “relativi”.

6.3. Preparazione ISWorkbench e repository XML

Configuriamo ISWorkbench per farlo puntare al nuovo progetto ISFPrimer.

- Modifichiamo il file ISWorkbench.ini presente nella cartella del repository XML (MyISFPrimer\Dictionary) cambiando ISFTemplate in MyISFPrimer. Di conseguenza rinominiamo il file del database di esempio allo stesso modo (il database si trova nella cartella MyISFPrimer\Database).
- Sempre all'interno si ISWorkbench.ini modifichiamo HELPFILE valorizzandolo MyISFPrimerHelp.hsc e HELPDOC valorizzandolo MyISFPrimerHelp.hsc
- Modifichiamo anche il file FDConnections.ini (sempre nella cartella MyISFPrimer\Dictionary), cambiando i nomi delle connessioni, ad esempio [ISFTemplate_Interbase] in [MyISFPrimer_Interbase] e così via, quindi la relativa path del database (basta cambiare il nome del db) ..una modifica analoga verrà poi fatta al file MyISFPrimer.xml, più avanti.
- Modifichiamo i 2 files FunzioniVuoto.md e HelpVuoto.md (sono file di sorgente in formato Markdown) con il MarkDown Editor, sostituendo la stringa [ApplicationName] con MyISFPrimer.
- Creiamo un collegamento a ISWorkbench.exe (come quelli di esempio installati nel menu) aggiungendo come primo parametro la path del repository XML del nostro progetto (MyISFPrimer\Dictionary) in modo tale che lanciandolo sia già posizionato sulla cartella corretta. Verifichiamo la connessione al DB, provando a far girare la “verifica del database”.
- Proviamo a generazione l'help dell'applicazione: verrà aggiornato il file MyISFPrimerHelp.md partendo dal file FunzioniVuoto.hsc che abbiamo personalizzato. Riapriamo il file MyISFPrimerHelp.md e compiliamolo in modo da generare il file MyISFPrimer.md che è il file di help vero e proprio utilizzato dall'applicazione.

6.4. Preparazione dell'applicazione ISFPrimer

Premessa: si è preferito lasciare “manuali” queste modifiche, a carico del programmatore, così è in grado di cogliere come è configurata una applicazione ISF. Ci sono pochi files che contengono le informazioni necessarie, che vanno configurati opportunamente.

Interveniamo a livello di progetto Delphi per creare il nostro progetto MyISFPrimer.

- Apriamo il progetto ISMultiTemplate.dpr della cartella MyISFPrimer\Projects\DXXXX (in base alla propria versione di Delphi) e salviamolo come MyISFPrimer nella stessa cartella.
- Cambiamo versione, Copyright del progetto e l'icona dell'applicazione nelle project options.

  (N.B. Se si riduce il numero della versione occorre cambiarlo a mano nel DB per evitare che il controllo blocchi l'avvio).
- Cambiamo Application Title ed altri riferimenti a “Template” contenuti nel dpr.
- Compiliamo l'applicazione, che verrà generata nella cartella \MyISFPrimer\Exe.
- Prima di avviarla modifichiamo alcuni files nella cartella \MyISFPrimer\Exe e ne correggiamo il contenuto:
  - ISMultiTemplate.exe.manifest va rinominato come MyISFPrimer.exe.manifest.
  - ISMultiTemplate.ini va rinominato come MyISFPrimer.ini

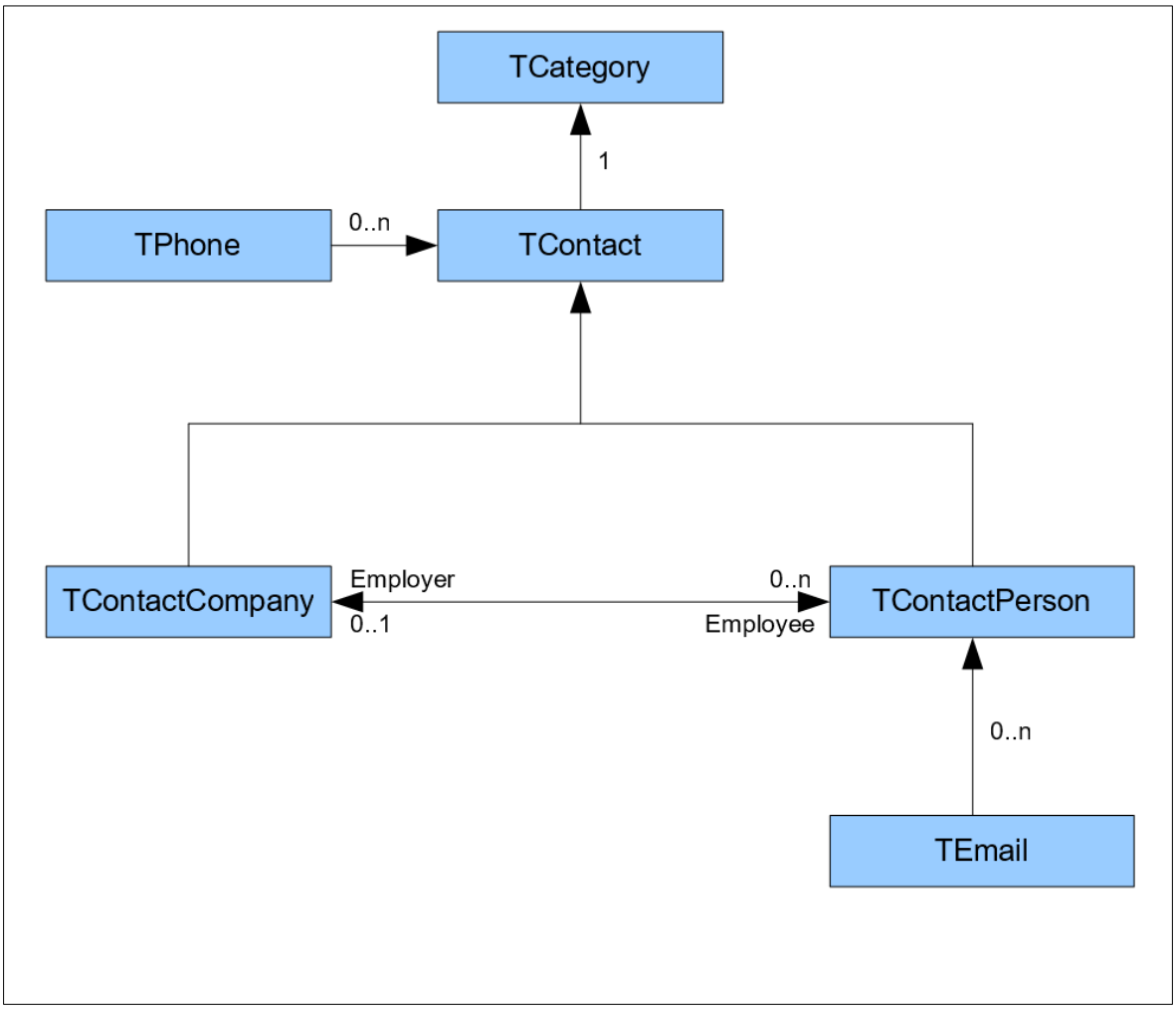
- ISMultiTemplate.xml va rinominato come MyISFPrimer.xml (idem per il file _DEBUG).
- Inoltre bisogna modificarne le sezioni <Name></Name> e <ConnectionString></ConnectionString> adattandole al nostro progetto MyISFPrimer, analogamente a quanto fatto prima per il file FDConnections.ini di ISWorkbench.
- Personalizziamo le immagini di splash e sfondo a piacimento.

6.5. Creazione delle classi dell'applicazione

6.5.1. Diagramma delle classi

Intendiamo creare una applicazione per la gestione di una rubrica. Il diagramma a cui ci ispiriamo è lo stesso del progetto Primer di InstantObjects, con lo sola eccezione della classe TAddress, che nel progetto originario era una classe di tipo TPart (in pratica un oggetto contenuto in un altro oggetto), mentre nel nostro caso l'indirizzo principale sarà contenuto nella classe di base dei contatti per semplificare. Anche la classe TCountry non verrà creata.

Successivamente creeremo anche la classe TAddress, ma come “dettaglio” della classe TContact, per gestire indirizzi “alternativi”.



6.5.2. Creazione delle classi di base

- Creiamo le classi per le “tabelle di lookup” (TCategory) e le classi secondarie da utilizzarsi nelle relazioni master-details utilizzando ISWorkbench. Queste classi le creiamo nel modulo MDictionary.

CATEGORIES (Tcategory)
Categoria nominativo della rubrica

Field name	Attribute Name	Type	Ref.	Size	Vis.	Dec.	Req.	Extnd description
ID	Id	Character	No	32	6		Yes	Category id
DX	Description	Character	No	200	30		Yes	Category
UPDTIMESTAMP	UpdateTimeStamp	Date and Time	No		0		Yes	Update TimeStamp
SCORE	Score	Small integer	No	3	3		No	Score

EMAILS (Temail) Embedded

E-mails del nominativo della rubrica

Field name	Attribute Name	Type	Ref.	Size	Vis.	Dec.	Req.	Extnd description
ID	Id	Character	No	32	32		Yes	Id
DX	Description	Character	No	200	100		Yes	E-mail
UPDTIMESTAMP	UpdateTimeStamp	Date and Time	No		0		Yes	Update TimeStamp
EMAILTYPE	EmailType	Character	No	40	40		No	E-mail type

PHONES (Tphone) Embedded
Contatti telefonici del nominativo della rubrica

Field name	Attribute Name	Type	Ref.	Size	Vis.	Dec.	Req.	Extnd description
ID	Id	Character	No	32	32		Yes	Number
DX	Description	Character	No	200	100		Yes	Descrizione
UPDTIMESTAMP	UpdateTimeStamp	Date and Time	No		0		Yes	Update TimeStamp

- Generiamo i sorgenti per le nuove classi (Modules/Units nella colonna di sinistra, poi l'icona con gli “ingranaggi” in alto). Oltre a creare le classi di base nel modulo (MDictionary) vengono creati anche singole units, una per ciascuna classe, che vengono salvate nella cartella src.
- Aggiorniamo la struttura del database, con la sequenza: 1) elimina relazioni, 2) aggiorna, 3) crea relazioni.
- Aggiungiamo a mano nel dpr tali units, sia come files.pas che nella lista delle classi gestite da InstantObjects.
- Ricompiliamo e avviamo l'applicazione: si presenta la finestra di login:

- accedendo come SYSDBA (masterkey), per aggiungere le funzioni utilizzando il menu Aggiorna Funzioni: verrà creata la funzione con id “GSTD_nomeclasse”
- Avviamo la funzione “sistema/gestione funzioni” e sistemiamo a mano le funzioni create, impostando le icone (email: 118, telefoni: 108, Categorie: 68).
- Aggiungiamola al menu dei parametri di sistema:
  - clicchiamo su “Edit menu structure” sulla toolbar della finestra del menu
  - Selezioniamo a sinistra la cartella dei parametri di sistema e facciamo doppio-click a destra sulla funzione della classe categorie che verrà aggiunta alla cartella parametri.
- Connettiamoci come utente normale (utente: user; password: password) e avviamo la funzione che si trova nel menu “Parametri di sistema”
- proviamo ad inserire alcuni oggetti della classe TCategory.
- Osserviamo che cosa è successo sul database, per capire come ISF e InstantObjects memorizza i dati, attraverso ISWorkbench (gestione database/strutture dati/dati).

6.5.3. Creazione della classe di base per i contatti della rubrica

- Creiamo la classe di base dei contatti: TContact, aggiorniamo i sorgenti e il database, aggiungiamola nel dpr come abbiamo fatto per le altre classi in precedenza.

CONTACTS (TContact)
Contatti della rubrica

Field name	Attribute Name	Type	Ref.	Size	Vis.	Dec.	Req.	Extnd description
ID	Id	Character	No	32	10		Yes	Contact Id
DX	Description	Character	No	200	50		Yes	Contact

UPDTIMESTAMP	UpdateTimeStamp	Date and Time	No		0	Yes	Update TimeStamp
CATEGORY	Category	Reference	Sì		0	Yes	Category
ADDRESS	Address	Character	No	30	30	No	Address
ZIP	Zip	Character	No	5	5	No	Zip code
CITY	City	Character	No	35	35	No	City
PROVINCE	Province	Reference	Sì		0	No	Province
PHONECONTACTS	PhoneContacts	Container	Sì		0	No	Phone contacts

- Dal programma, entriamo come SYSDBA
- Creiamo la funzione e configuriamola (icona 138) come abbiamo fatto prima.
- Aggiungiamola al menu Anagrafica come abbiamo fatto prima.
- Lanciamo la funzione stavolta dal menu “ad albero”
- Proviamo ad editare un contatto generico indicando (per ora a mano) come Id '0000000001', inserendo i suoi dati e il dettagli del telefono.

6.5.4. Creazione delle classi che ereditano dalla classe di base in un nuovo modulo

- Prima di creare le classi che erediteranno da TContact creiamo il modulo **MDictionaryContacts**:
  - Nome Unit: MDictionaryContacts
  - Descrizione: Modulo delle classi derivate da TContact
  - Uses: MDictionary, MContact e MGeographic.
- Creiamo le 2 classi TContactPerson e TContactCompany, come classi che **ereditano da TContact** indicando come **modulo di appartenenza MDictionaryContacts**, e aggiungiamo alle persone il cognome e il nome, oltre alle e-mail e alle aziende l'url del sito web.

  Notare che gli attributo “ereditati” dalla classe TContact vengono mostrati come “inherited” e non sono modificabili se non in alcuni campi (es. la descrizione). Questo perché sono attributi derivati dalla classe TContact.

Archivio PERSONS (TContactPerson)
Persone della rubrica

Field name	Attribute Name	CharCase	Type	Ref.	Size	Vis.	Dec.	Req.	Extnd description
*ID	Id	Normal	Character	No	32	10		Yes	Contact Id
*DX	Description	UpperCase	Character	No	200	50		Yes	Contact
*UPDTIMESTAMP	UpdateTimeStamp		Date and Time	No		0		Yes	Update TimeStamp
*CATEGORY	Category		Reference	Sì		0		Yes	Category
*ZIP	Zip	Normal	Character	No	5	5		No	Zip code
*CITY	City	Normal	Character	No	35	35		No	City
*PROVINCE	Province		Reference	Sì		0		No	Province
*PHONECONTACTS	PhoneContacts		Container	Sì		0		No	Phone contacts
LASTNAME	LastName	Normal	Character	No	30	30		Yes	Last Name
NAME	FirstName	Normal	Character	No	30	30		No	First Name
EMAILS	Emails	Normal	Container	Sì		0		No	E-mails
SEX	Sex	Normal	Character	No	1	1		Yes	Sex
BIRTHDATE	BirthDate		Date	No		0		No	Birthdate
PHOTO	Photo		Image (link)	No	100	200		No	Photo

*Inherited Fields

Archivio COMPANIES (TContactCompany)
Aziende della rubrica

Field name	Attribute Name	Type	Ref.	Size	Vis.	Dec.	Req.	Extnd description
*ID	Id	Character	No	32	10		Yes	Contact Id
*DX	Description	Character	No	200	50		Yes	Contact
*UPDTIMESTAMP	UpdateTimeStamp	Date and Time	No		0		Yes	Update TimeStamp
*CATEGORY	Category	Reference	Sì		0		Yes	Category
*ZIP	Zip	Character	No	5	5		No	Zip code
*CITY	City	Character	No	35	35		No	City

*PROVINCE	Province	Reference	Sì		0	No	Province
*PHONECONTACTS	PhoneContacts	Container	Sì		0	No	Phone contacts
WEBSITE	WebSite	Character	No	100	100	No	Url Sito Web

*Inherited Fields

- Aggiorniamo il database
- Creiamo/Aggiorniamo le classi
- Aggiungiamo a mano nel dpr del nostro progetto Delphi le units del modulo MDictionaryContacts e MDictionaryContactsConsts e le units delle classi MPerson e MCompanies (ricordarsi di aggiungere anche le stesse unit nella sezione {$R *.mdr}
- Come sempre, creiamo le funzioni (icona Company:91, icona Person: 60) e le relative voci di menu.
- Proviamo a fare il data-entry di queste 2 nuove classi

E' possibile specificare se una stringa è normale, UpperCase o LowerCase.

6.5.5. Creazione di un attributo Container (Master-Detail) e della classe relativa

Per InstantObject il concetto di "Master-Detail" è completamente ribaltato, cioè il riferimento (foreign-key) non sta nella tabella di "dettaglio" per puntare al proprio "master", bensì è la classe/tabella master che contiene l'elenco dei riferimenti ai record di dettaglio, cosi come avviene normalmente in una generica classe Delphi, come ad esempio la classe TStringList che contiene al suo interno l'elenco dei riferimenti agli oggetti (Objects[x]) a cui ogni elemento della stringlist fa riferimento.

Qesto concetto però non consente di effettuare delle ricerche sfruttando il linguaggio SQL, partendo dalle classi di dettaglio di una classe "master", pertanto in ISF si è cercato di evolvere il concetto di Container semplice di References, al concetto di Container di References verso una classe di dettaglio che a sua volta ha un Reference verso il proprio oggetto Master.

Questo tipo di doppio riferimento circolare, utilizzando InstantObject nativo può creare qualche problema di "riferimenti circolari" che non consentirebbe agli oggetti referenziati con un meccanismo di reference-counting di liberarsi dalla memoria, creando una situazione di stallo. Con ISF invece questo problema è stato risolto, ma non solo: quando si crea un oggetti di dettaglio, il reference al proprio oggetto master viene creato e gestito automaticamente.

Per fare un esempio creiamo la classe degli indirizzi alternativi di un oggetto TContact, chiamandola TAddress.

Archivio ADDRESSES (TAddress)
Indirizzi dei contatti

Field name	Attribute Name	Type	Ref.	Size	Vis.	Dec.	Req.	Extended description
ID	Id	Character	No	32	32		Yes	Id
DX	Description	Character	No	200	100		Yes	Address
UPDTIMESTAMP	UpdateTimeStamp	Date and Time	No		0		Yes	Update TimeStamp
CONTACT	Contact	Reference	Sì		0		Yes	Owner Contact of the address
ADDRESSTYPE	AddressType	Character (list)	No	20	20		Yes	Address type
ADDRESS	Address	Character	No	30	30		Yes	Address
ZIP	Zip	Character	No	5	5		No	Zip
CITY	City	Character	No	35	35		No	City
PROVINCE	Province	Reference	Sì		0		No	Province

N.B. L'attributo Contact è definito come Reference ma soprattutto è stato anche definito il suo Container nella classe "master", che ancora non esiste e dovremo crearlo. Nella classe master TContact, dobbiamo aggiungere quindi l'attributo Addresses che abbia come storage name quello definito nell'attributo Contact della classe TAddress:

Field name	Attribute Name	Type	Ref.	Size	Vis.	Dec.	Req.	Extended description
ADDRESSES	Addresses	Container	Sì				No	Contact addresses

N.B. Avendo creato un nuovo atttributo nella classe "di base" dobbiamo avere l'accortezza di aggiornare anche le classi che derivano da TContact, cioè TContactCompany e TContactPerson: è sufficiente "fingere" di modificare un campo della classe derivata (es. la descrizione) per ottenere il refresh dei suoi attributi e vedere che è comparso anche Address tra gli attributi "inherited".

Rigeneriamo il database e i sorgenti. Ricompiliamo e avviamo l'applicazione per vedere che ora, nella finestra dei Contatti (e anche delle Persons e delle Company) esisite una pagina in più che mostra i "dettagli" di un indirizzo.

6.6. La gestione delle immagini in un progetto ISF

In un applicativo ISF le immagini sono risorse condivise "al di fuori" del database dei dati, che contiene solo i nomi dei files, oppure "al di dentro" attraverso l'uso di campi Blob.

6.6.1. Gestione delle immagini “esterne”

Per usufruire di questo meccanismo occorre definire la cartella delle immagini. In ambito multiutente sarà una cartella condivisa in rete.

Entriamo nell'applicazione come *SYSDBA* con password *masterkey* in modo da abilitare anche i menu Amministratore e il menu Sistema. Apriamo il menu Amministratore/Configurazione ed andiamo nella pagina delle “Cartelle” per cambiare l'impostazione “Cartella condivisa immagini” con **{app}\..\images**.

Provare a modificare l'immagine associata ai dati di una persona (il campo Photo). Dal menu scegliere “Persons”, si apre la finestra delle Persone. Scegliere un record, andare in “form” e fare doppio click su photo e selezionare un'altra immagine. Attenzione: essa verrà prima copiata nella cartella delle immagini condivise e poi “agganciata” nella mappa.

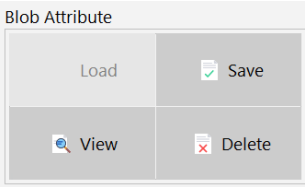
6.6.2. Uso del componente TCBDBBlob: Gestione delle immagini “interne”

Per poter gestire un'immagine “interna” al database, è necessario creare un campo “Blob” (Binary Large Object). Proviamo con la classe TContactCompany a creare il campo “CompanyLogo”:

Field name	Attribute Name	Type	Ref.	Size	Vis.	Dec.	Req.	Extended description
LOGO	CompanyLogo	Blob	No		200*		No	Company Logo

*200 è la “With in form”

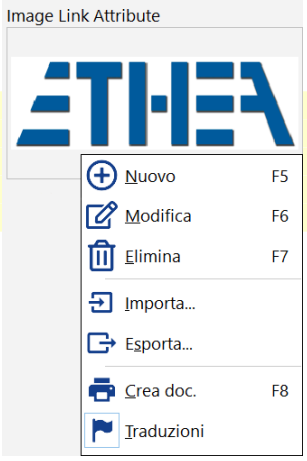
- Rigeneriamo/aggiorniamo come sempre classi e database
- Ricompiliamo e riavviamo l'applicazione: nella Form delle company ora appare anche il campo “Company Logo” in questo modo, con il “Viewer” di default di un campo Blob:



I 4 pulsanti consentono di:

- Caricare un file nel campo Blob
- Salvare il contenuto del Blob in un file
- Visualizzare un file
- Cancellare il contenuto del Blob
- N.B. Il viewer di default non è in grado di mostrare in automatico il contenuto del blob, occorre customizzare la form e fornire alcune indicazioni, ma questo “complessità” esula da questo corso rapido.

Quello che possiamo fare per mostrare le immagini è registrare un “viewer” in grado di farlo: fortunatamente un viewer per le immagini già pronto è disponibile, ed è attivato da queste due righe della unit UmultiAppSpecific.pas:




```
initialization
  //Assegno il viewer delle immagini
  TCBDbBlob.RegisterViewerClass(TCBDbBlobImageViewer);
finalization
  TCBDbBlob.UnregisterViewerClass(TCBDbBlobImageViewer);
```

Il risultato è quello mostrato nell'immagine: quando nel campo Blob viene caricata un'immagine essa viene subito mostrata e le operazioni sono “disponibili” sotto forma di menu contestuale.

6.6.3. Uso di TCBDbBlobHTMLViewer: memo con contenuto HTML

Dalla versione 6.1 è disponibile questo nuovo componente in grado di gestire un campo memo/testuale con contenuto in formato HTML, con la possibilità di cliccare sugli hyperlink in esso mostrati. In ISWorkbench occorre definire un campo Memo HTML:

Field name	Attribute Name	Type	Ref.	Size	Vis.	Dec.	Req.	Extended description
MEMOHTMLATTR	MemoHTMLAttribute	Memo HTML Attribute	No		0		No	Memo HTML Attribute

Rigeneriamo database e classi e osserviamo che sul DB il campo così creato è un campo memo.

Come per i campi Blob, quello che possiamo fare per mostrare il contenuto HTML del memo formattato correttamente è registrare un “viewer” in grado di farlo: è disponibile un viewer per i dati in formato HTML, ed è attivabile da queste righe della unit UmultiAppSpecific.pas:

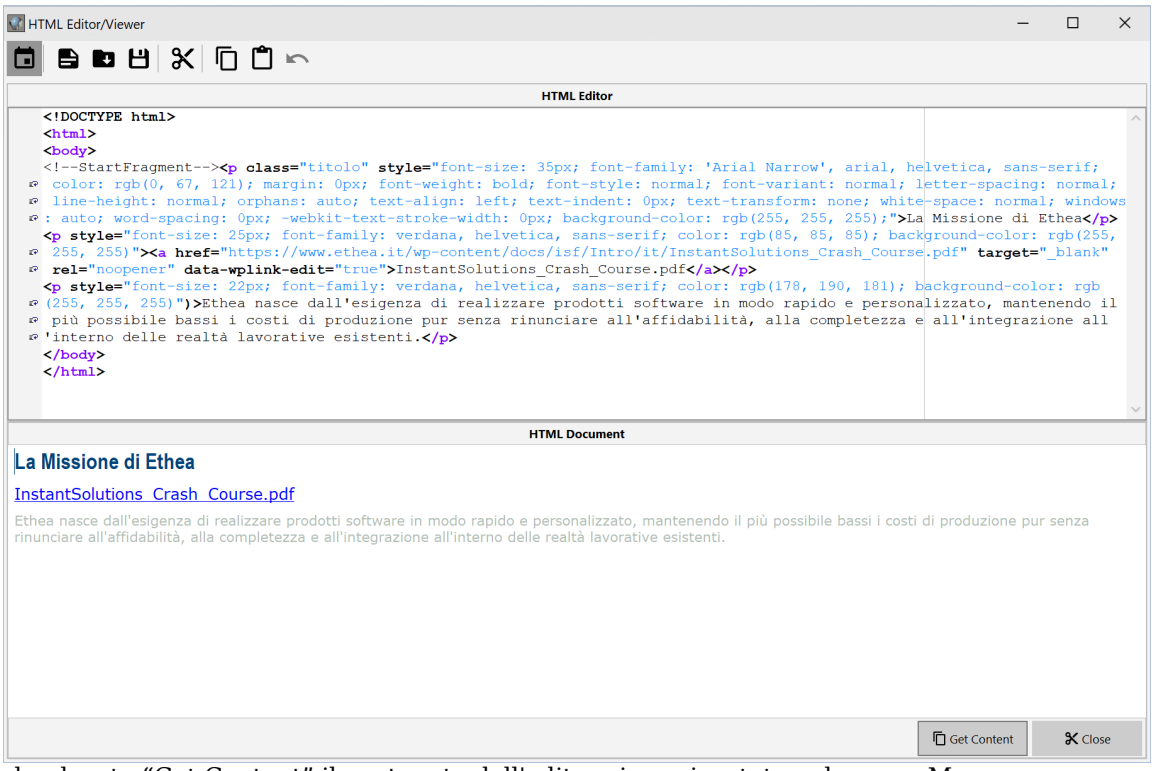
```
uses
  ....
  CBDBBlob, CBDBHtmlViewer, FCBHtmlFileViewer;
```

Il risultato è quello mostrato nell'immagine: quando nel campo Memo viene caricato un file HTML esso viene subito mostrato e le operazioni sono “disponibili” sotto forma di menu contestuale.

Da notare che gli hyperlink presenti nel testo HTML sono cliccabili: viene aperto il browser e visualizzata la pagina corrispondente.

E' anche possibile editare il contenuto del campo HTML selezionando dal menu contestuale: “Show Content”: si aprirà l'editor HTML suddiviso in 2 parti: la parte superiore è il testo HTML modificabile, la parte inferiore è il

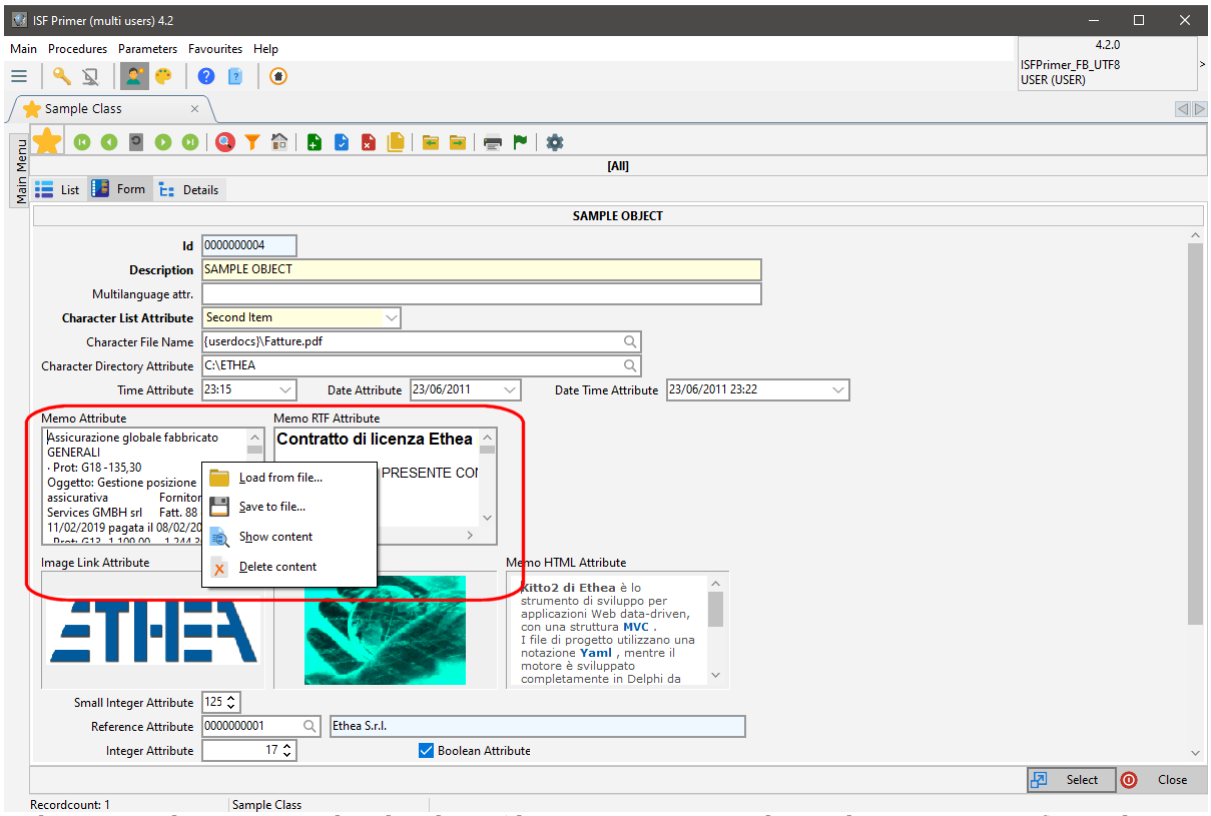
preview del contenuto:



Cliccando sul pulsante “Get Content” il contenuto dell'editor viene riportato nel campo Memo.

6.7. Gestione di campi “Memo”, “RTF” e HTML con editor integrato

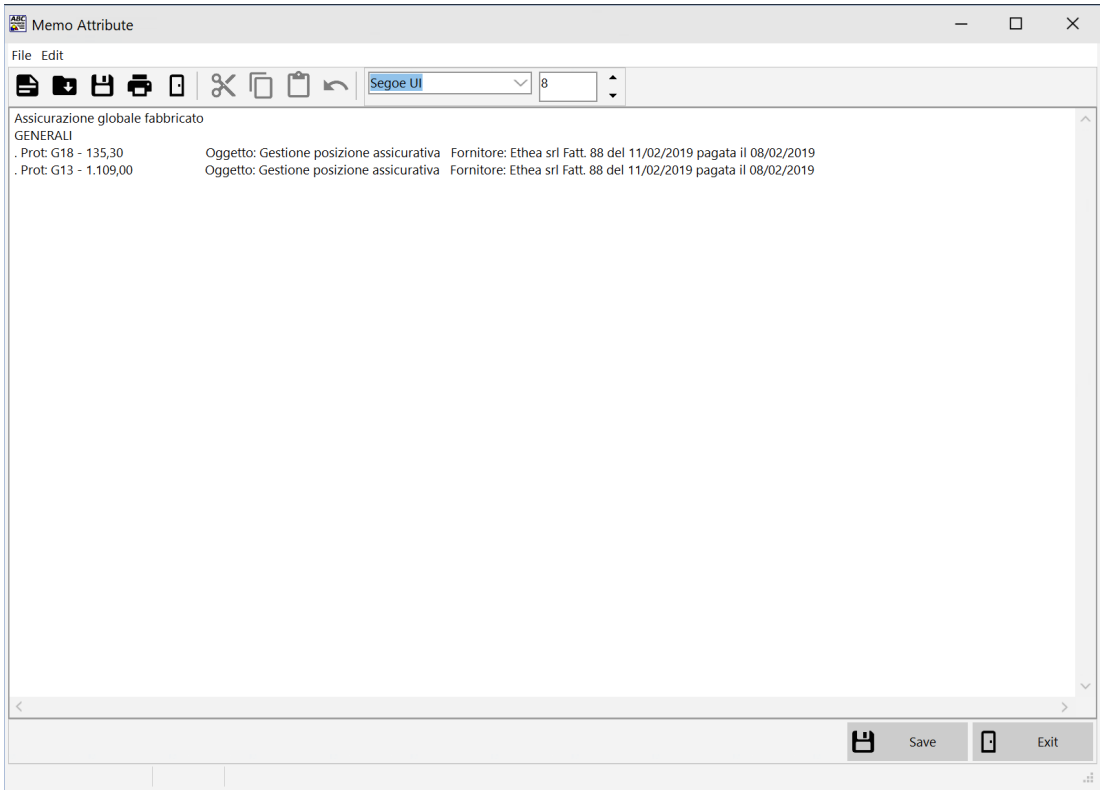
Dalla versione 7.5 di ISF è possibile sfruttare la possibilità di definire un nuovo tipo di campo “Memo (RTF)” all’interno di ISWorkBench. A livello database il campo è gestito come un semplice Memo di testo, in quanto il contenuto RTF è un testo contenente i comandi RichText di formattazione. Sempre dalla stessa versione, a livello di maschera, questi 2 tipi di campi vengono gestiti con 2 nuovi componenti evoluti per permettere la gestione più semplificata di questi campi: TCBXDBMemo e TCBXDBRichEdit.



Come si vede in maschera, entrambi gli editor (di un memo “normale” e di un memo “rtf) ora hanno un menu contestuale che permette di caricare e salvare il contenuto in un file, nonché di visualizzarlo, selezionando “Show content” oppure facendo doppio-click.

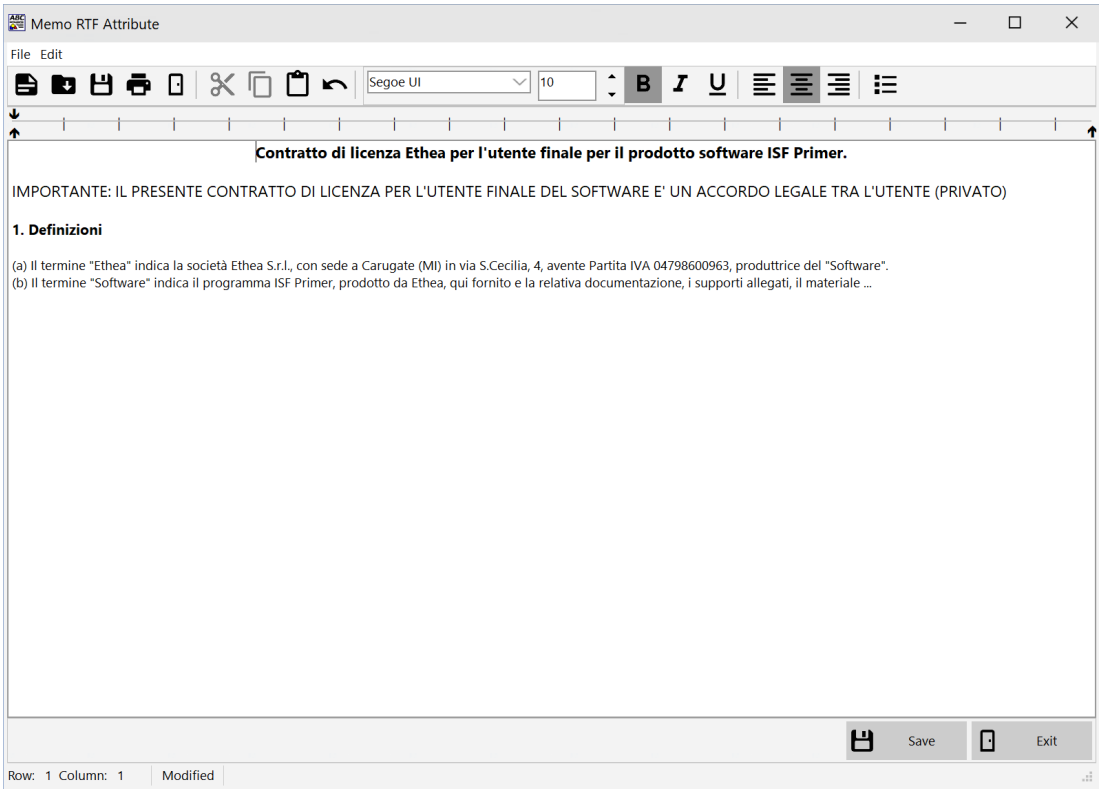
6.7.1. Visualizzatore/Editor campo memo

Nel caso di un memo “normale” compare un editor di testo per facilitare l’input del campo.



6.7.2. Visualizzatore/Editor contenuto RTF

Nel caso di un memo “rtf” compare un editor Rtf per facilitare l’input del campo e consentire la formattazione.



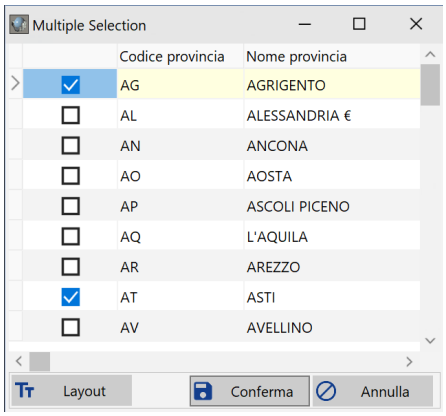
6.8. Gestione di un campo “MultiValue”

In ISF è possibile definire un campo stringa che contiene una serie di valori, di vario tipo (Date, numeri, importi, ecc...) oppure una sequenza di “reference” a un'altra classe.
Proviamo a creare un nuovo attributo di tipo “MultiValue” nella classe SampleClass:

Field name	Attribute Name	Type	Ref.	Size	Vis.	Dec.	Req.	Extended description
MULTIREF	MultiReference	Carattere (MultiValue)	Si		0		No	Multi Ref. Attribute

Lasciamo il default per il “Tipo di valore” cioè “References” e definiamo la classe Refrenziata come TProvince.
Rigeneriamo database e classi e osserviamo che sul DB il campo così creato è un campo stringa.
A video, l'input si presenza così:

Premendo sul pulsantino con i tre puntini si apre la form di selezione multipla del reference impostato (le province).
Nel campo stringa il valore viene memorizzato come una serie di codici:
AL;AN;AP;BA



6.9. Utilizzo di un campo “MultiValue” in una classe di ricerca

Un campo multivalue può essere utilizzato anche in una classe di ricerca, per cercare tanti valori omogenei.
Per gestire il passaggio dei valori del campo multivalue occorre utilizzare questa sintassi nel metodo:

```
function TISXXX.GetWhereCondition(TargetClass: TClass): string;
begin
  WhereCond := '';
  if MultiProvince <> '' then
    AddAndWhereCond(CalcMultiORCondition('Province.Id',MultiProvince));
end;
```

La chiamata a CalcMultiORCondition genera tanti parametri quanti sono i multivalori contenuti nel campo MultiProvince; ad esempio se ho tre codici provincia verrà generata la WhereCondition così:

```
(Province.Id = :Province_Id_1 OR Province.Id = :Province_Id_2 OR Province.Id = :Province_Id_3);
```

Quindi i parametri vanno valorizzati con i vari codici provincia contenuti nel campo MultiProvince: per fornire questi valori occorre ereditare un altro metodo:

```
procedure TISXXX.SetParamValues(TargetClass: Tclass; ParamValues: TParams);
begin
  inherited;
  SetMultiORValues('Province.Id',MultiProvince, mcReference, ParamValues);
end;
```

Attenzione al terzo parametro di tipo TmultiValueContentType: questo tipo è definito in CBClasses e serve per indicare il tipo di valore contenuto nel campo MultiValue.
Se ad esempio ipotizziamo di avere un campo di ricerca contenente una serie di “Date” avremo:

```
  SetMultiORValues('BirthDate',MultiDates, mcDate, ParamValues);
```

Implementando semplicemente questi 2 metodi nella propria classe di ricerca è possibile fare ricerche multivalore.

6.10. Aggiornamento dell'help dell'applicazione

- Proviamo ora ad aggiornare l'help dell'applicazione.
- Da ISFWorkbench aggiorniamo il file dell'help (creando quindi il file MyISFPrimerHelp.md) che finirà nella cartella Help del nostro progetto.
- Dall'applicativo premiamo F1: l'help è contestuale al punto in cui ci troviamo!

6.11. Scrittura del codice Delphi per le regole di business

Iniziamo a scrivere del codice Delphi nelle classi per implementare alcune regole di business.

6.11.1. Creare un codice progressivo per una classe

- Utilizziamo un meccanismo per generare un codice progressivo per il codice del contatto (e rendiamo read-only il campo): questo meccanismo verrà ereditato anche dalle altre 2 classi!

```
procedure TISContact.AfterCreate;
begin
  inherited;
  Id := TISIdGenerator.CalcNewProgress('TISContact', GetIDSize);
end;
```

6.11.2. Creare regole di validazione

- Creiamo una regola di validazione di un campo: il valore di e-mail deve necessariamente contenere @.

```
procedure TISEmail.SetDescription(const Value: string);
begin
  //Vaildazione e-mail
  if (Value <> '') and (Pos('@',Value)=0) then
    raise Exception.Create('e-mail deve contenere @');
  inherited;
end;
```

6.11.3. Ricercare oggetti con l'uso di InstantQuery

- Nella classe TISCity è presente un meccanismo di recupero di altre informazioni su comune e provincia a partire dal CAP (Zip code). **Questo esempio è importante per capire l'utilizzo dell'oggetto TInstantQuery:**

```
class function TISCity.RetrieveCityFromZip(const Zip: string): TISCity;
var
  InstantQuery : TInstantQuery;
begin
  InstantQuery := InstantDefaultConnector.CreateQuery;
  Try
    //cerco direttamente il comune con il CAP passatomi
    InstantQuery.Command := 'SELECT * FROM TISCity WHERE Zip = :ZIP';
    //Se nella TinstantQuery ho dei parametri devo fare questo, altrimenti non "vede" i parametri:
    InstantQuery.FetchParams(InstantQuery.Command, InstantQuery.Params);
    InstantQuery.Params.ParamByName('ZIP').AsString := Zip;
    InstantQuery.Open;
    if InstantQuery.ObjectCount = 0 then
    begin
      //Se non trovo il CAP direttamente provo ad "azzerare" le ultime due cifre
      InstantQuery.Close;
      InstantQuery.Params.ParamByName('ZIP').AsString := Copy(Zip,1,3)+'00';
      InstantQuery.Open;
    end;
    if InstantQuery.ObjectCount > 0 then
    begin
      Result := InstantQuery.Objects[0] as TISCity;
      Result.AddRef; //è un funzione di retrieve
    end
    else
      Result := nil;
  Finally
    InstantQuery.Free;
  End;
end;
```

Lo utilizziamo nella classe del contatto e ne beneficeranno le classi TISContactPerson e TISContactCompany:

```
procedure TISContact.SetZip(const Value: string);
var
  ISCity : TISCity;
begin
  //valido il Value
  if (Value <> '') and (Length(Value)<>5) then
    raise Exception.Create(Format('Il valore di %s deve essere lungo %d caratteri',
      [CONTACT_ZIP_TITLE,5]));
  inherited;
  //Recupero la città e la provincia a partire dal CAP
  if (Self.City = '') and (Self.Province = nil) and (Value <> '') then
  begin
    ISCity := TISCity.RetrieveCityFromZip(Value);
    try
      if Assigned(ISCity) then
      begin
        Self.Province := ISCity.Province;
        Self.City := ISCity.Description;
      end;
    finally
      ISCity.Free;
    end;
  end;
  UpdateAttributes;
end;
```

6.11.4. Calcolare automaticamente la descrizione in una classe

Come abbiamo visto tutte le classi che ereditano da TCBInstantObject avranno sempre un attributo “descrizione”. Questo attributo è molto utile per descrivere un oggetto (es. quando lo si mostra in una form con codice+descrizione). Spesso però il l'attributo Description non deve essere imputato dall'utente ma deve essere calcolato automaticamente sulla base del contenuto di altri attributi.

Prendiamo la classe TISContactPerson e definiamo l'attributo Descrizione come “sola lettura” dentro ISWorkbench e rigeneriamo le classi.

Scriviamo un metodo privato CalcDescription nella classe TISContactPerson.

```
procedure TISContactPerson.CalcDescription;
begin
  Description := Trim(LastName+' '+FirstName);
end;
```

Questo metodo deve essere richiamato ogni volta che cambiano gli elementi che lo determinano in questo caso quando si settano Cognome e Nome. Scriviamo nella sezione protected i “setter” di questi attributi (in override) e implementiamoli così:

```
procedure TISContactPerson.SetLastName(const Value: string);
begin
  inherited;
  CalcDescription;
end;

procedure TISContactPerson.SetFirstName(const Value: string);
begin
  inherited;
  CalcDescription;
end;
```

Applichiamo questo principio anche alla classe degli indirizzi: la descrizione di un indirizzo è la composizione dell'indirizzo stesso.

Aggiungiamo la regola di calcolo dell'indirizzo scrivendo un metodo privato CalcDx nella classe TISAddress.

```
procedure TISAddress.CalcDx;
var
  ProvId : string;
begin
  if Assigned(Province) then
    ProvId := '('+Province.Id+')'
  else
    ProvId := '';
  Description := Address+' '+Zip+' '+City+' '+ProvId;
end;
```

Scriviamo nella sezione protected (in override) i “setter” di tutti gli attributi che concorrono all'indirizzo e

implementiamoli così:

```
procedure TISAddress.SetZip(const Value: string);
var
  ObjCity : TISCity;
begin
  inherited;
  //ricerco il City tramite il Zip
  if (Value <> '') and (City = '') then
  begin
    ObjCity := TISCity.RetrieveCityFromZip(Value);
    Try
      if ObjCity <> nil then
      begin
        City := ObjCity.Description;
        Province := ObjCity.Province;
      end;
    Finally
      ObjCity.Free;
    End;
  end;
  CalcDx;
end;

procedure TISAddress.SetCity(const Value: string);
begin
  inherited;
  CalcDx;
end;

procedure TISAddress.SetAddress(const Value: string);
begin
  inherited;
  CalcDx;
end;

procedure TISAddress.SetProvince(Value: TProvince);
begin
  inherited;
  CalcDx;
end;
```

Il metodo di SetZip contiene il codice per recuperare il comune dal cap, come già visto per la classe TContact.

6.11.5. Sincronizzare dati tra l'oggetto master e i suoi dettagli

Il concetto che abbiamo introdotto nella struttura delle classi come indirizzi “alternativi” pone il problema di avere tra gli indirizzi alternativi anche quello contenuto sull'oggetto master, cioè l'indirizzo principale e di mantenerne sincronizzati i valori.

Trovate l'implementazione di questi metodi nell'esempio fornito con ISF, nei metodi:

```
TISContact.RetrieveDefaultAddress
TISContact.UpdateContactFromDefaultAddress
TISContact.AddOrUpdateDefaultAddress
```

servono per sincronizzare i dati fra loro, e vengono invocati nei metodi AfterStore sia di TContact che di TAddress.

6.11.6. Master-Detail: con l'approccio classico

ISF rimane sempre e comunque aperto ad un approccio classico di relazione Master-Detail, specie in situazione dove entrambe le classi hanno una valenza tale da non poter sempre considerare una classe come dettaglio di un'altra, e viceversa. Per questo motivo in ISF la classe TCBInstantSelector consente anche la relazione Master-Detail con un altro Selector o Exposer. Basta indicare in DataSource il datasource del dataset master, e indicare nella proprietà WhereCondition la condizione di where con i parametri necessari alla relazione master-detail (il loro nome deve corrispondere al nome di campi chiave presenti nel dataset master). Il TCBInstantSelector master si occupa automaticamente di aggiornare il proprio dettaglio (chiudendolo, assegnando i nuovi parametri e riaprendolo) ogni volta che cambia di record.

6.11.7. UpdateAttributes e RefreshLayoutAttributes

Uno dei problemi che spesso annoiano lo sviluppo di applicazioni e di interfacce utente è la necessità di riflettere a livello di layout di una mappa le condizioni del record (oggetto in questo caso) corrente. Tipicamente si visualizzano o si rendono obbligatori o meno alcuni campi (attributi, in questo caso) in base al contenuto di altri.

Con ISF occorre intervenire a 2 livelli:

1) Implementare a livello di classe il metodo UpdateAttributes (override) e scrivere la logica di variazione dei valori degli attributi in base al valore di un altro attributi.
2) Fornire al TCBInstantSelector o TCBInstantExposer (collegato ad un pannello di editing TCBXDBMultiEdit), la proprietà "RefreshLayoutFields" indicando la lista dei campi che con il loro valore condizionano il valore di altri attributi. In questo modo, quando il selector/exposer applica la modifica del field all'oggetto sottostante sa che deve "refreshare" gli attributi di tutti i suoi campi (es.visible, readonly o required). Di consequenza il pannello di editing collegato si riadatterà automaticamente a queste nuove impostazioni.

Proviamo ad implementare questa cosa aggiungendo un metodo UpdateAttributes nella classe TISContact:

```
procedure TISContact.UpdateAttributes;
begin
  if Zip <> '' then
    _City.Required := True
  else
    _City.Required := False;
end;
```

Provando ora a caricare una persona o una company non possiamo lasciare vuoto il campo del comune se abbiamo specificato il Cap (attributo Zip). Rimane il problema che a livello di interfaccia utente questo "cambiamento" non è visibile. Occorre impostare la proprietà RefreshLayoutFields del Selector/Exposer come già detto.

Se si usa il Framework GUI Multi (vedi capitolo specifico) basta indicare questo valore nella mappa della funzione in "campi autorefresh".

N.B. Dalla versione 5.3 è anche possibile definire tale informazione a livello di classe, nella funzione RefreshLayoutAttributes. Questo farà in modo che qualsiasi Selector/Exposer collegato sia in grado di recuperare tale informazione automaticamente, quindi non è più necessario indicarla nella funzione associata.

```
class function TISContact.RefreshLayoutAttributes: string;
begin
  Result := 'Zip';
end;
```

In questo modo, in tutte le mappe che mostrano una classe derivata da TISContact (Companies e Persons) non appena si modifica il campo Zip (sbiancandolo o inserendo un valore) cambierà il colore del campo Comune (che riflette il fatto che è diventato obbligatorio o meno). E' anche possibile far sparire o comparire un campo in editing (impostando Attributo.Visible a True o False).

6.11.8. Estendere una classe del framework

A volte c'è la necessità di estendere una classe del framework, aggiungendo per esempio degli attributi. Non essendo possibile intervenire direttamente sulla classe del framework stessa, pena la retrocompatibilità futura, è possibile sfruttare un meccanismo di "notifica" che InstantObjects mette a disposizione. In pratica si registra una classe che riceve le notifiche da un'altra classe (quella del framework) ogni volta che un oggetto di tale classe viene manipolato.

Un esempio di come operare è presente nella unit UObjectNotifier.pas presente nella cartella src di ISFPrimer.

6.12. Utilizzo dei Messaggi all'interno di InstantSolutions

Per mostrare i messaggi agli utenti, InstantSolutions mette a disposizione alcune funzioni semplificate:

ISInform o **ISInformFmt** per un messaggio di informazioni

ISWarning o **ISWarningFmt** per un messaggio di warning

ISError o **ISErrorFmt** per un messaggio di errore bloccante

la versione **Fmt** consente il passaggio di parametri al messaggio formattato come avviene normalmente per Format.

Per i messaggi di richiesta di conferma è possibile utilizzare: ISConfirm o ISConfirmFmt

Se si vuole maggior controllo sui pulsanti o il tipo di dialog si può chiamare:

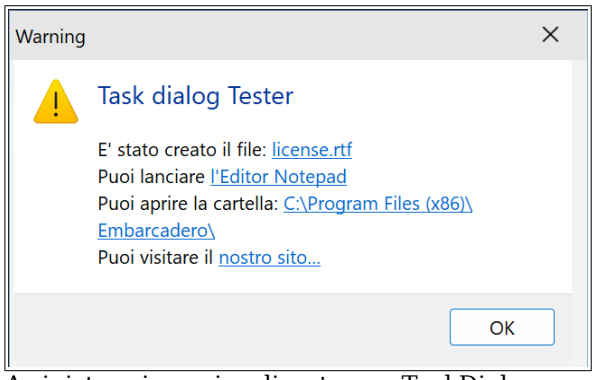
ISMessageDlg o **ISMessageDlgFmt**.

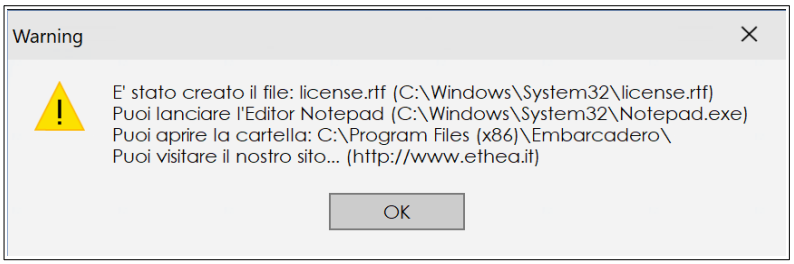
E' importante utilizzare queste funzioni in quanto chiamano le funzioni di libreria in grado di gestire autonomamente se utilizzare il task dialog (disponibile da Windows Vista in poi) o il message dialog.

Inoltre, è anche possibile formattare il messaggio attraverso hyperlink: con l'uso dei taskdialog gli hyperlink sono attivi e l'utente ci può cliccare sopra.

Ad esempio, con questo messaggio:

```
  Msg :='E'' stato creato il file: '+StringToHRef('C:\Windows\System32\license.rtf','license.rtf')+sLineBreak+
        'Puoi lanciare '+StringToHRef('C:\Windows\System32\Notepad.exe','l ''Editor Notepad')+sLineBreak+
        'Puoi aprire la cartella: '+StringToHRef('C:\Program Files (x86)\Embarcadero\')+sLineBreak+
        'Puoi visitare il '+StringToHRef('http://www.ethea.it','nostro sito...');
```





A sinistra viene visualizzato con TaskDialog, grazie agli hyperlink, a destra con il vecchio message dialog.

Con il taskdialog l'utente può cliccare sopra gli hyperlink, aprire il file, accedere ad una cartella o ad un sito, ecc...

Da notare l'utilizzo della funzione StringToHRef che in modo semplice formatta una due stringhe facendole diventare un Hyperlink come ad esempio:

```
StringToHRef('c:\windows\system32\Notepad.exe','Editor')
```

produce una stringa:

```
<A HREF="c:\windows\system32\Notepad.exe”>Editor</A>
```

Allo stesso modo è possibile estrarre gli elementi di una stringa Href (ExtractHrefValues) o ripulirla (HRefToString).

6.13. La gestione della “funzione” (TISFunzione) nel MultiFramework

Di default le funzioni create hanno una serie di parametri preimpostati che possono essere facilmenti modificati ed adattati. La cosa importante è capire che il MultiFramework avvia le finestra attraverso le regole definite nelle funzioni e che queste regole sono in prima battuta personalizzabili a livello di funzione e comunque modificabili a livello applicativo. Per esempio possiamo cambiare l'ordinamento delle persone con “Nome”.

Vediamo quali solo i parametri che è possibile indicare nell’oggetto “Funzione”:

- **Id:** Codice della funzione, di default **GSTD_TIS*CLASSNAME***
- **Description:** Descrizione della funzione
- **HelpContext:** è l’indice della pagina per l’help
- **FormClass:** è la classe della Form del Framework che viene utilizzata
- **ModalForm:** Finestra “Modale”
- **EditFormClass:** se specificata, è la classe della Form del Framework che viene utilizzata in caso di inserimento/modifica: a volte si usa la TformEdit, che è pensata per l’editing di un oggetto singolo, senza dettagli.
- **ModalEdit**: Finestra di Edit “Modale”
- **InsertFormClass:** se specificata, è alternativa alla EditFormClass in caso di inserimento: spesso si indica un Wizard che guida l’utente nell’iserimento dei dati.
- **ModalInsert**: Finestra di Insert “Modale”
- **IOClassName:** è la classe InstantObject gestita nella funzione
- **RequestedLoadMode:** Tipo di caricamento dei dati (scelta con tendina): il default è definito nel file .ini
- **FirstAction:** è la prima azione che la form esegue (si sceglie da una tendina: se è vuota equivale a “Browse”)
- **CustomParameters:** sono parametri opzionali: è possibile scrivere qualsiasi “attributi=valore” purché l’attributo sia una proprietà “Published” della form.
- **ExtendedDescription:** è una descrizione estesa (200 caratteri)
- **Flag_Menu:** Visibile nel menu ad albero
- **ImageIndex:** Indice dell’immagine associata alla funzione (si sceglie da una tendina)
- **ImageName:** Nome dell’immagine associata alla funzione (si sceglie da una tendina)
- **WhereCond:** Condizione di filtro “globale”: se la form è nativa va scritto in SQL altrimenti in IQL
- **WhereCondDesc:** Descrizione della condizione di filtro “fissa”
- **OrderBy:** Condizione di ordinamento (SQL o IQL)
- **RefreshFields:** Elenco dei campi che scatenano l’aggiornamento del layout della mappa
- **ConfigMode:**
- **ReferenceStopLevel:** Livello di dettaglio del reference: di default è 1, solo i reference di primo livello
- **AllDetailFields:** Mostra tutti i campi degli oggetti referenziati (default no)
- **LargeTable:** Flag per indicare che la tabella è grossa e che non deve essere aperta subito, ma solo dopo

una ricerca

- **FilterPages:** definizione dei filtri che verranno mostrati come pagine dell'eleneco (è un contenitori di oggetti TISFuncFilter)
- **Roles:** definizione dei ruoli che verranno mostrati con la "SideBar" (è un contenitore di oggetti TISFuncRole
- **DetailPages:** definizione delle pagine degli oggetti "detail" che si vogliono mostrare (è un contenitore di TISFuncDetails)
- **ActiveFilterPage:** Filtro di pagina attivo di default
- **ActiveRole:** Ruolo attivo di default
- **ActiveDetailPage:**
- **Context:** Identificativo del contesto dei dati

6.14. La gestione del "Contesto" nelle funzioni

Quando a livello applicativo si devono gestire molti dati, organizzati in contesti diversi, può essere utile suddividere le funzioni in modo da "filtrare" automaticamente i dati in base al contesto.

Ad esempio in un gestionale di gestione delle utenze (ACQUA, LUCE, GAS), dove l'informazione "TipoFornitura" è trasversale a quasi tutti gli archivi, è utile poter definire a livello di ISFunction un "Context" della funzione, ad esempio "ACQUA" e nel filtro globale REF_TIPO_FORNITURA_REF.ID='ACQUA': in questo modo la finestra mostrerà solo i dati riferiti ai consumi dell'ACQUA, ma sopratutto, in fase di "lookup" sui reference cercherà l'esistenza di una funzione "speciale" dal nome **ACQUA_TISCLASS*NAME***: se la trova invoca questa funzione anziché la standard GSTD_TIS*CLASSNAME*. Così facendo l'utente naviga sempre nei dati filtrati nel contesto in cui si trova.

A livello di oggetti, esiste una variabile di classe "FCurrentContext" che viene tenuta sincronizzata dal framework, ed è quindi possibile sapere in quale contesto un oggetto sta operando in quel momento.

6.15. Come si creano e si memorizzano gli oggetti: DemoData.pas

Prendendo spunto dalle analoghe unit della demo Primer (DemoData.pas e RandomData.pas) di instantObjects costruiamo una unit per creare dei dati di prova dell'applicazione. E' utile vedere il codice scritto: notare che si lavora a classi e oggetti, nessun concetto di "DataSet" occorre per scrivere i processi elaborativi.

Nella unit Dfunzioni (che serve anche per customizzare le icone dell'applicazione) proviamo a creare N contatti dopo esserci collegati al database:

```
uses DemoData;

procedure TdtmdFunzioni.AfterConnect(ConnectionDef : TInstantConnectionDef);
begin
  ............
  CreateRandomContacts(10);
end;
```

6.16. Come si esportano i dati

6.16.1. Esportazione via codice

La libreria CBLib fornisce alcune classi già pronte per l'esportazione di qualsiasi "DataSet" in vari formati, quali CSV, TXT, XML, Excel. Per esportare i dati da codice basta chiamare una semplice funzione, passando una serie di parametri specifici e una serie di event-handler per la scelta dei record o dei campi da esportare: es.

```
uses CBExport;

...
  ExportDataSetToCSVFile(nil,DataSet,FileName,
    IncludeFieldNamesChecked,Delimiter,QuotedChecked,
    AcceptRecord,AcceptField,GetFieldValue);
```

6.16.2. Esportazione in Excel tramite template (xlt o xltx)

In caso di esportazione in Excel è possibile specificare anche un foglio Excel di "modello" (xlt o xltx) da utilizzare per l'esportazione, in modo tale che il risultato sia formattato secondo le specifiche del modello, es:

```
uses CBExport;

...
  ExportDataSetToExcelFile(DefaultProgressHandler,ExportExposer,FileName,
    EXCEL_SHEET_NAME,True,AcceptRecord,AcceptField,GetFieldValue,
    True, TemplateFileName);
```

Il template deve avere particolari caratteristiche:

1) deve avere un "Range" definito con il nome di "Sheet1" o "Foglio1" (la costante EXCEL_SHEET_NAME)

2) tale range deve essere di 2 righe: la prima deve contenere il nome del campo, la seconda un valore di esempio formattato nel modo corretto (stringa o numero o data, ecc...).

Il modo più semplice per costruire un template è quello di esportare un file excel con il "Wizard", selezionando i campi che interessano. Aprire quindi il file generato; cancellare tutti i record, lasciandone uno solo, e "azzerare" i valori del primo record mettendo dei valori vuoti; salvare il file come xlt (nella cartella dei modelli del progetto).

In questo modo il Wizard lo proporrà come file di modello di default per l'esportazione (vedere come esempio il modello "Persons_to_export.xlt" nella cartella DocTemplates di ISFPrimer).

6.16.3. Esportazione attraverso la GUI standard FExportWizard

In ISF (MultiTemplate) esiste già una form Wizard di esportazione dei dati, resa già disponibile per esportare qualsiasi DataSet da qualsiasi finestra.

In questa form è anche possibile definire delle regole di esportazione e salvarle per poterle ricaricare in un secondo momento.

6.16.4. Importazione attraverso la GUI standard FImportWizard

In ISF (MultiTemplate) esiste già una form Wizard di importazione dei dati, resa già disponibile per importare qualsiasi DataSet da qualsiasi finestra.

In questa form è anche possibile definire la mappatura dei campi di importazione e le regole di importazione sono salvabili per poterle ricaricare in un secondo momento.

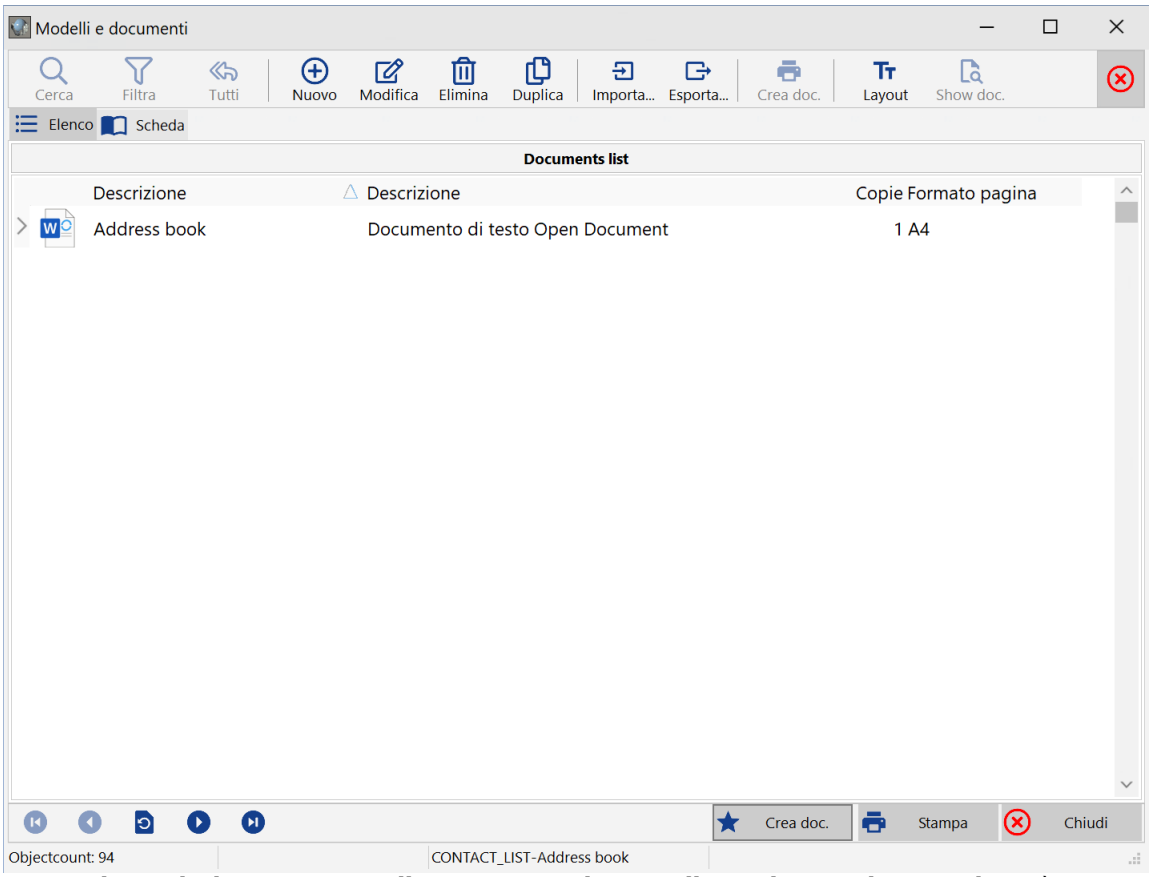
6.17. Come si creano le stampe con ISF

In ISF è molto facile creare delle stampe utilizzando diverse tecnologie implementate nella classe "CBDocProducer" della CBLib. La cosa interessante è che il lavoro viene svolto principalmente a run-time con strumenti che possono essere affidati anche all'utente finale o ad una persona che non abbia competenze di sviluppo Delphi.

Proviamo a creare lo stesso tipo di stampa con tecnologie diverse.

6.17.1. La mappa di "selezione stampa"

La mappa che consente all'utente di selezionare il documento da "generare" o da "stampare" mostra l'elenco delle stampe compatibili con la "classe" chiamante: per "generare" un documento (il file relativo) senza stamparlo premere "Crea doc." mentre per stamparlo premere "Stampa".



N.B. Ci sono 2 tipologie di documenti, quelli "monorecord" e quelli "multirecord": quando si è posizionati su una stampa "mono record" si attivano anche i pulsanti "Generate" e "Print All" che consentono di generare/stampare molti documenti, uno per ogni elemento della lista dalla quale la scelta della stampa è stata attivata.

6.17.2. Generazione di una lista (Report)

Elenco con OpenOffice:

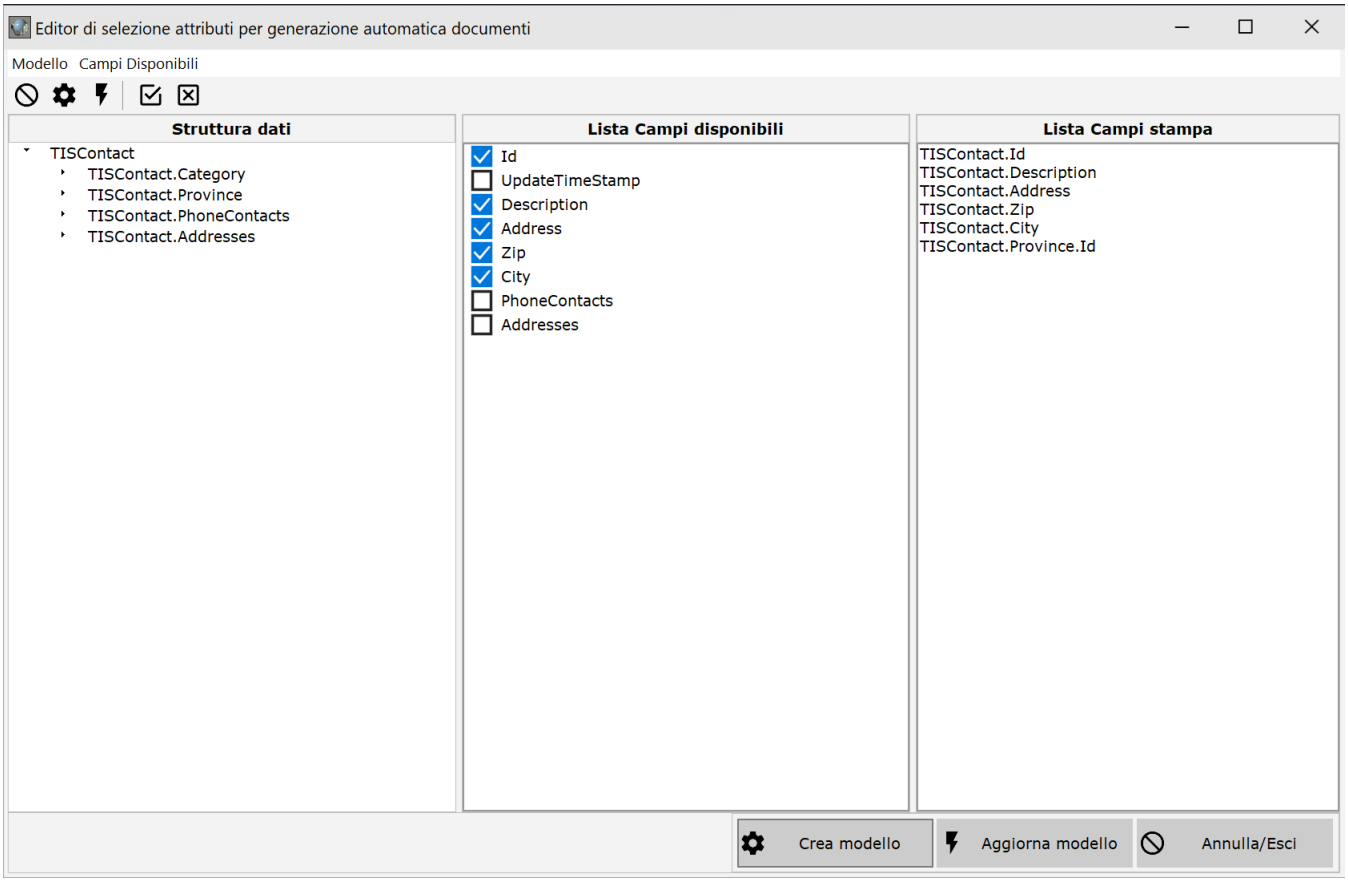
Proviamo a creare delle stampe di elenco di contatti (Login come SYSDBA).

Entriamo nella funzione “Address Book” clicchiamo sulla stampante per accedere alla finiestra mostrata in figura che mostra la lista delle stampe disponibili per questa classe. Inizialmente è vuota.

Creiamo un nuovo record così:

```
Model: CONTACT_LIST
Description: Address book
File Name: Contact_List
Document type: D
Copies: 1
Editable: No
Page format: A4
Landscape: Si
Owner Class: TISContact
Use Source Data: No
Records number: -1
```

Confermiamo i dati ed andiamo col cursore nel campo “Field for the document”: si attiva il pulsante di layout. Attiviamo il layout editor e selezioniamo i campi per la stampa:




```
TISContact.Id
TISContact.Description
TISContact.Address
TISContact.Zip
TISContact.City
TISContact.Province.Id
```

Sempre da dentro il “Layout editor” generare il template, ed uscire.

Provate a lanciare la stampa per vedere il risultato.

Elenco con FOP:

Cloniamo il template appena creato per provare con FOP, cambiando:

```
Modello: CONTACT_LIST_FOP
Tipo Documento: F
```

Generiamo il template e proviamo a generare la stampa.

6.17.3. Generazione di una “scheda” con dettagli

Lavoriamo stavolta sulle “persone”, e dalla relativa mappa, clicchiamo sulla stampante per accedere alla lista delle stampe disponibili per questa classe.

Creiamo una scheda con OpenOffice:

```
Model: PERSON_FORM
Description: Person Form
File Name: Person_Form
Document type: D
Copies: 1
Editable: No
Page format: A4
Landscape: No
Owner Class: TISContactPerson
Use Source Data: No
Records number: 1
```

Confermiamo i dati ed andiamo col cursore nel campo “Field for the document”: si attiva il pulsante di layout. Attiviamo il layout editor e selezioniamo i campi per la stampa:

```
TISContact.Id
TISContact.Description
TISContact.Address
TISContact.Zip
TISContact.City
TISContact.Province.Id
```

Sempre da dentro il “Layout editor” generare il template, ed uscire.

Provate a lanciare la stampa per vedere il risultato. Con OpenOffice è possibile gestire anche i dettagli, ma solo di primo livello.

Collegare una foto in OpenOffice:

Per aggiungere la foto occorre creare un'immagine nel template e collegarla ad una immagine esterna.

Facendo proprietà sull'immagine, andiamo nella pagina “Collegamento” e indichiamo come Nome dell'immagine il nome del campo, in questo caso: TISContactPerson.Photo

In fase di creazione del documento, l'immagine esterna verrà collegata automaticamente.

Creiamo una scheda con FOP:

Cloniamo il template appena creato per provare con FOP, cambiando:

```
Modello: PERSON_FORM_FOP
Tipo Documento: F
```

Generiamo il template e proviamo a generare la stampa.

Con un editor XML modifichiamo il template (il file xsl) per ottenere il layout desiderato.

La gestione delle immagini con ReportBuilder

Per le immagini in ReportBuilder occorre includere la uses raIDE (abilita il RAP) nella unit UmultiAppSpecific per poter a run-time aggiungere un evento OnGetPicture su una DbImage così:

```
procedure DBImage1OnGetPicture(aPicture: TPicture);
begin
  aPicture.LoadFromFile('..\sito_Template\immagini\'+TISContactPerson_pipe['Foto']);
end;
```

N.B. Con ReportBuilder è fondamentale avere acquistato la versione enterprise che include il RAP che consente di fare tutto a run-time attraverso un linguaggio di scripting.

6.17.4. Mostrare la finestra di selezione stampante

E' possibile decidere se mostrare sempre la finestra di impostazione/scelta della stampante, quando si sta generando un report che ha come destinazione una stampante.

Per farlo occorre modificare la funzione GSTD_TISDOCREPORT (accedendo al programma con i privilegi di sistema) e impostare nella proprietà “parametri opzionali”:

```
ShowPrinterSettings=True
```

6.17.5. DocProducerEvent: per controllare eventi particolari legati ai reports

Ereditando la form di base è possibile intervenire su alcuni aspetti dei reports, grazie all'evento DocProducerEvent. Per esempio è possibile cambiare la stampante di default per alcuni reports specifici (perché per esempio devono essere dirottati su una stampante/fax virtuale), oppure fare una operazione non appena un report è stato generato (come aggiornare un flag di “stampato” sull'oggetto), ecc...

L'evento DocProducesEvent viene invocato molte volte, durante il processo di generazione di un report, con questa sequenza:

1) etSetMasterDataSet

2) etBeforeCreateTemplate

3) etAfterCreateTemplate

4) etOnPrinterSettings (solo se la destinazione del report è una stampante).

5) etBeforeCreateDocument

6) etAfterCreateDocument

In base a ciò che si intende fare occorre sempre testare in che “fase” di generazione del report ci si trova (come si vede nell'esempio sotto).

Inoltre una classe di tipo DocProducer può lavorare utilizzando un dataset e i suoi nested-dataset fields (come lo è normalmente un oggetto TInstantSelector o TInstantExposer). A volte è necessario “passare” al “producer” più di un dataset: è possibile farlo attraverso l'evento “DocProducerEvent”, come mostrato in questo esempio.

```
procedure TfmMiaForm.DocProducerEvent(DocReport: TISDocReport; EventType: TISDocEventType);
begin
  inherited;
  if (EventType = etSetMasterDataSet) and (DocReport.Id = 'XXX') then
  begin
    if DocReport.DocProducer.DetailDataSets.Count = 0 then
    begin
      //Aggiungo al DocProducer il dataset di dettaglio
      With DocReport.DocProducer.DetailDataSets.Add do
      begin
        DetailDataSet := MioSelector;
        DetailAliasName := GetDataSetAliasName(DetailDataSet);
      end;
    end;
  end;
end;
```

6.17.6. Customizzare i dati da stampare

Il meccanismo “automatico” di ISF prevede che le stampe che si vanno a generare siano stampe che sono collegate al “DataSet” attivo sulla finestra che ha richiamato l'operazione di stampa.

Spesso però c'è la necessità di lanciare una stampa i cui dati stanno “altrove” e non sono direttamente legati al DataSet attivo sulla finestra chiamante: in questi casi è possibile intervenire per dire al framework che il dataset da utilizzare per la stampa è un altro. Questa possibilità, per ora, è disponibile solo per il Framework “MultiWindows” e vi rimandiamo all'esempio più avanti al capitolo: **“personalizzare una stampa ad-hoc”**.

6.18. Abilitare filtri e ricerche con il MultiFramework

6.18.1. Definizione di filtri di pagina, ruoli e classi di ricerca

Nel MultiFramework esiste un concetto di ricerca/filtro sugli oggetti che incrocia 3 elementi: filtri di pagina, ruoli, filtri di classi di ricerca.

6.18.2. Abilitazione di filtri e ruoli

Possiamo provare a utilizzare una form del framework più evoluta, TformDataSideBar e creare tre “ruoli” per i maschi e le femmine o tutti nella finestra delle persone.

- Role Description: Male – Filter Expression: Sex='M' – Image Index: 104
- Role Description: Female – Filter Expression: Sex='F' – Image Index: 84

Possiamo definire dei ruoli nella form dei contatti utilizzando la form TformDataSideBar:

- Role Description: All – Filter Expression: 1=1 – Image Index: 138
- Role Description: Companies – Filter Expression: Class = 'TISContactCompany' – Image Index: 91
- Role Description: Companies – Filter Expression: Class = 'TISContactCompany' – Image Index: 91
- Role Description: Persons – Filter Expression: Class = 'TISContactPerson' – Image Index: 68

6.18.3. Definizione di una classe di ricerca predefinita: WhereCondition e OrderByCondition

- Role Description: All – Filter Expression: 1=1 – Image Index: 60

Una classe di ricerca in InstantSolution è una classe speciale che fornisce una condizion di Where e di Orderby.

Per sviluppare una classe di ricerca occorre definire la classe in ISWorkbench, ereditando da TCBSelectObject. Nel codice Delphi occorre implementare il metodo **GetWhereCondition** della classe di ricerca. La classe di ricerca è una classe che può essere riutilizzata anche su più classi “target” (che è un parametro della GetWhereCondition.

In pratica GetWhereCondition deve restituire una query IQL o SQL (a seconda la si utilizzi in form standard o native) agli oggetti della classe “target” perciò la classe di ricerca può essere la stessa ma in base alla classe target può avere una sintassi diversa, con nomi di campi diversi.

Una classe di ricerca “predefinita” non prevede che ci siano operatori logici da imputare (and, or, ecc...) ma la logica è interna al metodo GetWhereCondition. Es. se devo fare una ricerca tra un range di 2 date definisco nella classe di ricerca un campo “dal” e un campo “al” e nella GetWhereCondition creerò la condizione >= valore di Dal e <= valore di Al, testando se i campi Dal e Al sono stati imputati. Per i valori come le date, dato che ogni motore database formatta il valore in modo diverso, è necessario utilizzare dei paramtri:

```
'DataStipula >= :DallaDataStipula
```

e fornire il valore del campo attraverso il metodo SetParamValues.

Per definire l'ordinamento del risultato della ricerca occorre scrivere il metodo **GetOrderByCondition**. Normalmente l'ordine viene stabilito dalla “Funzione” (di default è la descrizione) e l'utente può cambiare l'ordine cliccando sui titoli della griglia di elenco. Quando si forza la orderby in una classe di ricerca si cambia l'ordine prestabilito, altrimenti rimane quello preesistente.

Ad esempio la classe di ricerca TISSelLocalita permette di cercare oggetti di una classe che abbia al suo interno i campi: Cap, Comune e Provincia.Id, esattamente come la nostra classe TContact.

Per rendere disponibile una classe di ricerca predefinita occorre “registrare” alla nostra classe la classe di ricerca, nell'initialization:

```
  //registrazione classi di ricerca
  CBRegisterSelectClasses(TISContact,[TISSelLocalita]);
```

A questo punto nella form dei contatti è disponibile il pulsante di “filtro” (imbuto) per abilitare la ricerca.

Una cosa molto comoda della ricerca è la possibilità di memorizzarne i criteri e richiamarli successivamente.

Il filtro di ricerca viene “combinato” con il concetto di “ruolo” e di “filtro pagina” già visti.

6.18.4. Esempio: creare una classe di ricerca per gli indirizzi

Per **definire la clausola di Where** in base all'input dell'utente, è sufficiente ereditare da TCBSelectObject ed implementare nella unit della classe il metodo GetWhereCondition sfruttando anche i metodi:

```
function CalcLikeCondition(const PropName, Value: string): string;
function OperatorCondition(const PropName, Operator, Value: string): string; overload;
function OperatorCondition(const PropName, Operator : string; Value: integer): string; overload;
procedure AddAndWhereCond(const Condition: string);
```

presenti nella classi di base della ricerca, per costruire più facilmente la condizione di where.

Ad esempio:

```
function TISSelLocation.GetWhereCondition(TargetClass: TClass): string;
begin
  WhereCond := '';
  if Zip <> '' then
    AddAndWhereCond(CalcLikeCondition('Zip',Zip));
  if City <> '' then
    AddAndWhereCond(CalcLikeCondition('City',City));
  if Province <> nil then
    AddAndWhereCond('Province.Id = '+QuotedValue(Province.Id));

  Result := WhereCond;
end;
```

N.B. È possibile costruire la query IQL anche attraverso l'uso di parametri (es. DataStipula < :Oggi). Il valore del parametro deve essere impostato nella class Procedure SetFilterParamValue della classe.

6.18.5. Il meccanismo legato a SetFilterParamValue

Nella classe di base TCBInstantObject esiste un metodo molto utilizzato per impostare valori di ricerca specificati nelle condizioni di where di una classe di ricerca, oppure di un filtro di pagina o di ruolo.

La classe di basse implementa già l'impostazione di alcuni parametri “standard” come:

```
'No' o 'False' <- False
'Si' o 'Yes' o 'True' <- True
'Oggi''Today' <- Date
'Adesso' o 'Now' <- Now
'DataVuota' o 'EmptyDate' <- 0
'InizioMese' o 'StartOfMonth' EncodeDate(YearOf(Date),MonthOf(Date),1)
'FineMese' o 'EndOfMonth' EncodeDate(YearOf(Date),MonthOf(Date),DaysInMonth(Date));
```

Altri parametri “non standard” vanno gestiti all'interno della singola classe, ereditando SetFilterParamValue.

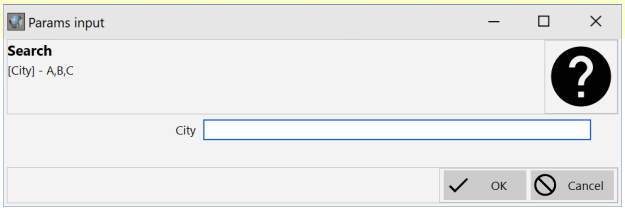
6.18.6. Utilizzare i parametri nelle ricerche/filtri/classi di ricerca

E’ possibile far imputare il valore dei parametri di ricerca direttamente all'utente. Prima era necessario che un parametro fosse sempre “risolto” all'interno del metodo *SetFilterParamValue* della classe sulla quale viene fatta la

ricerca. Per dare la possibilità all'utente di indicare alcuni parametri di ricerca su una certa classe occorre implementare la class function *GetHiddenFilterParam* (di default questa function torna True, perché tutti i parametri non sono editabili dall'utente).

Proviamo a concedere la possibilità di indicare parametri di ricerca alla classe dei contatti. Scriveremo:

```
class function TISContact.GetHiddenFilterParam(const ParamName: string): boolean;
begin
  //Su questa classe posso applicare le ricerche "libere" con parametri "utente"
  Result := False;
end;
```



A questo punto proviamo ad aggiungere all'interno della funzione GSTD_TISCONTACT dei "contatti della rubrica" un nuovo "Ruolo" dal nome "Città" con l'espressione di filtro: (Comune LIKE :Comune).

Quando nella mappa cliccheremo sull'icona della "città" appare la richiesta del valore del parametro (come si vede in figura).

Utilizzando il % è possibile effettuare ricerche parziali.

E' possibile utilizzare questa tecnica anche per i filtri di pagina, oltre che per i ruoli, oppure anche a livello applicativo, quando si usa un componente TCBInstantSelector.

E' necessario impostare l'evento TCBInstantSelector.OnGetParamValues e nell'event-handler chiamare la funzione CBEditParams (l'esempio è disponibile in FData.pas).

La stessa cosa è stata fatta nella form FQueryView (le ricerche libere): se si imposta una ricerca con dei parametri scatta il meccanismo di input.

6.18.7. Utilizzare un meccanismo customizzato di passaggio dei parametri

La form standard di input dei parametri (vedi figura) è solo uno dei modi possibili di fornire alla query che utilizza ricerche generiche per fornire il valore dei parametri. E' possibile agganciare all'event-handler TCBInstantSelector.OnGetParamValues qualsiasi metodo che fornisca direttamente o attraverso la richiesta di input all'utente i parametri necessari.

6.18.8. Utilizzare una classe di ricerca "generica"

Sfruttando lo stesso concetto dei parametri visto per i filtri di pagine e i ruoli, è anche possibile utilizzare una classe di ricerca "generica" per abilitare ricerche senza dover necessariamente creare una classe ad-hoc.

Vantaggi: non è necessario ricompilare l'applicativo per avere a disposizione nuove classi di ricerca (a patto di aver "abilitato" una classe a ricevere parametri di ricerca "utente", come spiegato al punto precedente.

Svantaggi: La gestione dei parametri di ricerca è meno avanzata

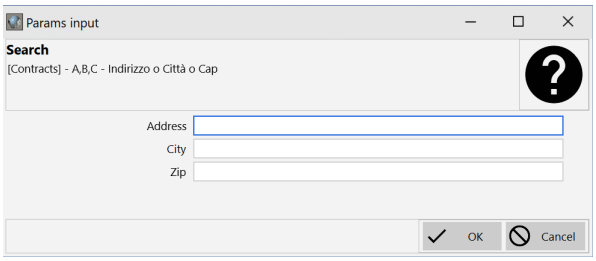
La classe TISSelGeneric (unit SSelGeneric.pas) e la sua classe di dettaglio TISSelGenericDetail, servono per gestire questo tipo di ricerca "generica".

Avviando il menu "Parametri di sistema/Condizioni di ricerca speciali" si entra nella finestra di gestione delle classi di ricerca generiche.

Basta inserire un nuovo record, indicare una descrizione per l'utente, un filtro e la classe sulla quale attivare la ricerca: per esempio:

```
Descrizione: Indirizzo o Città o Cap
Condizione where IQL: (Address = :Address) OR (City LIKE :City) OR (Zip = :Zip)
Oggetto della ricerca: TISContact
```

A questo punto basta avviare la finesta di gestione dei contatti e premere sul pulsante di ricerca/filtro (l'imbuto): scegliere "Condizioni di ricerca generiche/personalizzabili" e selezionare "Comune o provincia o Cap".



Apparirà la stessa mappa di inserimento dei parametri di ricerca, che mostrerà la lista dei campi di ricerca disponibili.

Per azzerare il filtro di ricerca "libera" occorre premere sul pulsante "Ripristina elenco completo" (la casetta).

N.B. La form di richiesta dei parametri è "cumulativa": se ci fossero più parametri relativi a ricerche libere o a filtri di pagina o di ruolo, essi vengono richiesti tutti insieme.

6.18.9. Utilizzare una classe di ricerca "generica" in modo "composito"

Una evoluzione introdotta nella versione 5.8 è la possibilità di comporre la classe di ricerca generica con operatori logici AND o OR a più livelli. I record di dettaglio di tipo TISSelGenericDetail di una classe di ricerca generica possono essere definiti in una struttura gerarchica con operatori e possono anche agganciare record di classi di ricerca già censiti. E' quindi possibile utilizzare la classe di ricerca generica per comporre ricerche complesse e sottoporle a logiche di inclusione o esclusione.

6.18.10. Customizzare i parametri di ricerca

Come già dicevamo, se all'interno di una classe si vogliono fornire dei valori di parametri di ricerca “automatici” (senza che l'utente li possa modificare) occorre fornire il valore del parametro nella class procedure SetFilterParamValue. Ecco come fare:

1) Aggiungiamo un “ruolo” alla funzione GSTD_TISContactPerson.
   1. Description = “In Provincia”
   2. Filter expression = “Provincia = :Provincia_Societa”
   3. Image index = 53

Così facendo, siccome abbiamo abilitato la classe TContact a ricevere parametri in input dall'utente, comparirebbe sempre la finestra di richiesta del parametro Provincia_Societa, ma noi vogliamo fornire a tale parametro un valore fisso, calcolato dall'applicazione.

2) Nella classe TContact ereditare la class procedure SetFilterParamValue e scrivere:

```
class procedure TISContact.SetFilterParamValue(Param: TParam);
var
  Societa : TISSocieta;
begin
  inherited;
  if SameText(Param.name,'Contact_Province') then
  begin
    Societa := RetrieveDefaultSocieta;
    try
      if Assigned(Societa.Provincia) then
        Param.AsString := Societa.Provincia.Id;
    finally
      Societa.Free;
    end;
  end;
end;
```

3) Nella classe TContact modificare la class procedure GetHiddenFilterParam e scrivere:

```
class function TISContact.GetHiddenFilterParam(
  const ParamName: string): boolean;
begin
  if SameText(ParamName,'Provincia_Societa') then
  begin
    //Questo parametro non è modificabile da parte dell'utente
    Result := True;
  end
  else
  begin
    //Di default i contatti consentono ricerche "libere"
    Result := False;
  end;
end;
```

In questo modo forniamo 2 informazioni: la prima è il valore del parametro (preso dalla provincia della società di default censita nel database), la seconda che il parametro è “nascosco” (hidden) e quindi non deve esserne richiesto l'input.

Lo stesso meccanismo funzionerà anche sulla classe TISContactCompany.

6.18.11. Classi di ricerca per form “Native”

Nel caso di form “Native” tutte le logiche legate alle classi di ricerca funzionano ugualmente a patto che la condizione di “Where” della classe di ricerca sia scritta in SQL e non IQL.

6.18.12. Filtrare i dati visibili nell'Explorer

Nella pagina dei dettagli è presente l'InstantExplorer per mostrare i dati di dettaglio che possono essere contenitori o oggetti. Dalla versione 6.4 è possibile organizzare questa pagina in sottopagine, se ci sono molti nodi da mostrare (vedi più avanti il capitolo specifico).

E' anche possibile applicare delle regole di filtro sugli oggetti mostrati sull'explorer (di tipo TCBInstantExplorer), sia a livello di nodo contenitore che di oggetti.

Per nascondere un certo nodo contenitore, occorre ereditare ed implementare:

procedure DoIncludeNode(Sender: TInstantExplorer;
NodeData: TInstantExplorerNodeData; var Include: Boolean); virtual;

Per i nodi degli oggetti non bisogna utilizzare la DoIncludeNode altrimenti gli oggetti sarebbero comunque visibili sulla griglia di un nodo contenitore, bensì occorre proprio filtrare gli oggetti attivando Limited e utilizzando l'evento

OnLimit del componente TCBInstantExplorer.

es. nella form custom FContacts implementare:

```
procedure TfmContacts.InitExposers;
begin
  inherited;
  //Filtro i dati dell'explorer
  ioExplorer.Limited := True;
  ioExplorer.OnLimit := LimitExploperObjects;
end;
```

e

```
procedure TfmContacts.LimitExploperObjects(Sender, AObject: TObject;
  var Accept: Boolean);
begin
  //Non mostro le emails di Gmail
  if (AObject is TISEmail) and (Pos('gmail.com', TISEmail(AObject).Description) > 0) then
    Accept := False;
end;
```

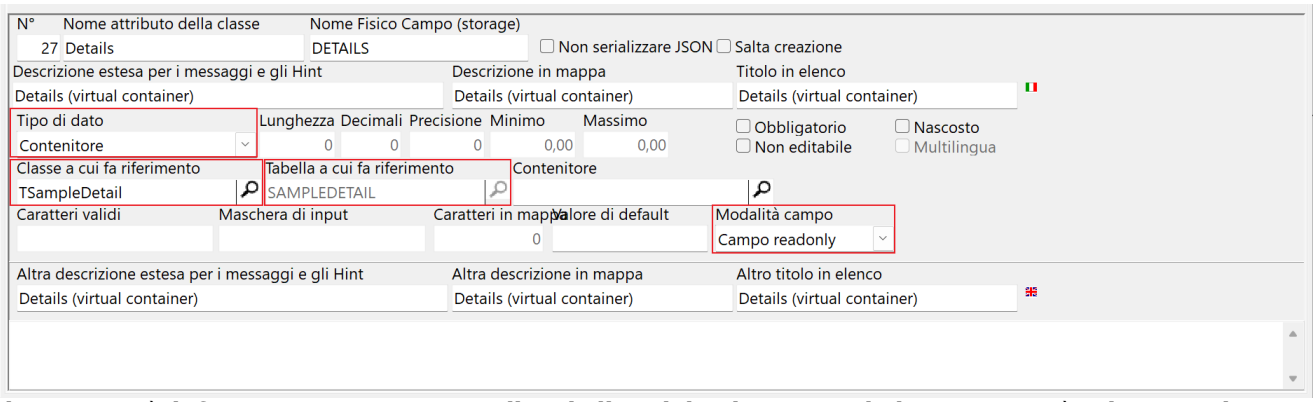
6.19. Attributo contenitore "virtuale"

In InstantSolution è possibile utilizzare classi di dettaglio di una classe master senza necessariamente avere un campo contenitore nell'oggetto master, a livello di Database.

A livello di classe InstantObject invece esisterà comunque un container di tipo "virtuale" che è in grado di accedere agli oggetti di dettaglio attraverso le informazioni di metadati fornitegli.

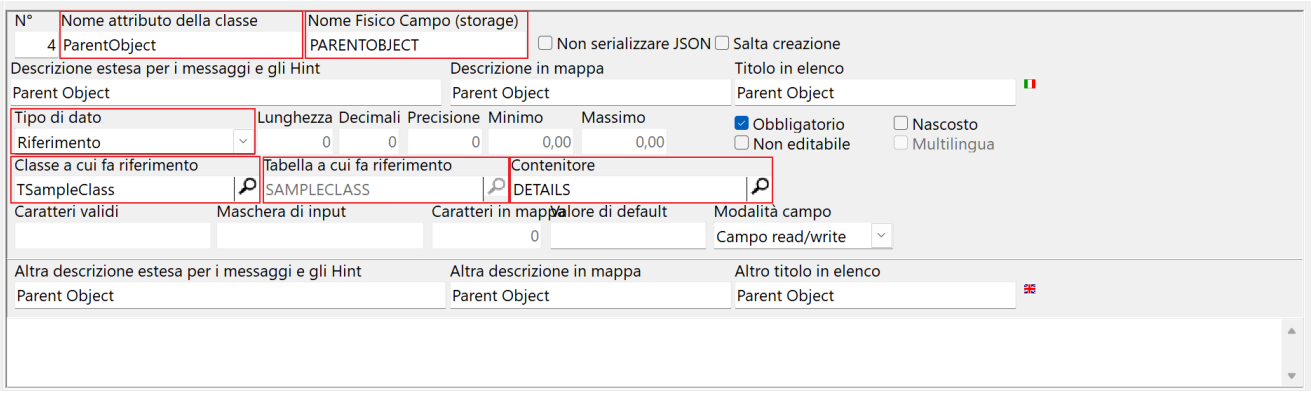
6.19.1. Gestione contenitori "virtuali" in ISWorkbench

Per prima cosa è necessario definire un contenitore virtuale all'interno di ISWorkbench. Tale contenitore è identico alla definizione standard tranne per il fatto di essere definito come campo "Read-Only", come nell'esempio dell'attributo Details della classe TISSampleClass del progetto ISFPrimer:



N.B. il campo così definito non viene creato nella tabella sul database, quindi il contenitore è solo "virtuale" e gestito all'interno della classe InstantObject.

Allo stesso modo, nella classe di dettaglio (TISSampleDetail), occorre definire un reference al master collegato al contenitore "virtuale". Questa configurazione è identica a quella per i contenitori non virtuali.



Da notare che è possibile definire il nome dell'attributo e del campo fisico collegato alla tabella master liberamente. Questa impostazione avrà l'effetto di generare una definizione di classe nuova nella unit Mdictionary.pas di questo tipo:

```
Details: References(TISSampleDetail) virtual 'SAMPLEDETAIL;PARENTOBJECTCLASS;PARENTOBJECTID';
```

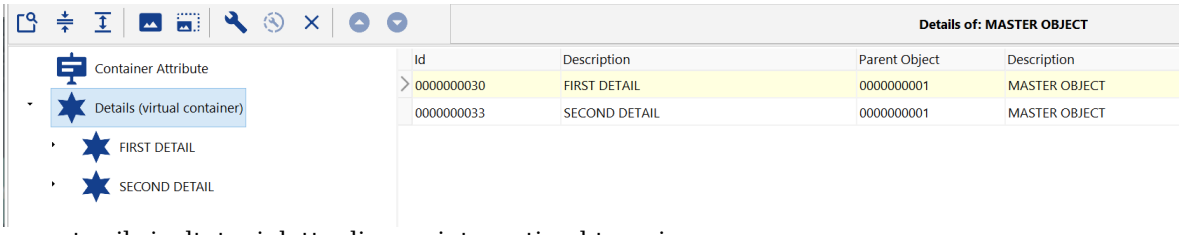
Da notare la nuova keyword "virtual" di un attributo di tipo references e lo storagename che contiene una stringa composta da tabella di dettaglio e dai campi chiave utilizzati per la join master-detail.

6.19.2. Contenitore “virtual” mostrato sull'Explorer

A livello applicativo non occorre fare altro in quanto sulla classe master continua ad esistere il contenitore (TCBInstantReferences) che contiene la lista degli oggetti dei record di dettaglio.

L'effetto immediato è quello di avere a disposizione sull'explorer il contenitore “virtuale” esattamente come se fosse memorizzato nel campo memo del database.

Il meccanismo interno fa in modo di utilizzare una query di accesso ai dettagli per caricare l'elenco degli “id” di dettaglio, anziché andare a leggere il contenuto del campo memo.



L'immagine mostra il risultato, i dettagli sono integrati nel treeview.

6.19.3. Ordinamento dei dettagli di un contenitore “virtual”

Di default il sistema ordina sempre i dettagli in base al campo Description, quindi l'ordine corrisponde al valore visibile sull'albero.

Non è possibile spostare gli oggetti di dettaglio con la Move, ma è possibile definire un ordinamento a piacere anche attraverso la join con altre tabelle.

Occorre implementare a livello di classe che ha il contenitore il metodo GetDetailsStatementValues impostando una o più variabili che vengono passate.

Potendo esserci più contenitori virtuali in una classe è più corretto testare il nome della tabella di dettaglio che viene passata nella FromClause.

Se si vuole implementare solo l'ordinamento basta aggiornare la variabile OrderByClause. Nell'esempio l'ordinamento è la descrizione decrescente:

```
procedure TISSampleClass.GetDetailsStatementValues(
  var FromClause, SequenceNoFieldName, OrderByClause: string);
begin
  inherited;
  if SameText(FromClause,'SAMPLEDETAIL') then
  begin
    OrderByClause := 'COMPANY.DX DESC';
  end;
end;
```

E' possibile anche stabilire un ordinamento in base a un campo appartenente ad una tabella referenziata dalla tabella di dettaglio. Nell'esempio l'ordine è dato dalla descrizione della company referenziata nel dettaglio.

```
procedure TISSampleClass.GetDetailsStatementValues(
  var FromClause, SequenceNoFieldName, OrderByClause: string);
begin
  inherited;
  if SameText(FromClause,'SAMPLEDETAIL') then
  begin
    FromClause := FromClause+sLineBreak+
      'LEFT OUTER JOIN COMPANY ON'+sLineBreak+
      'COMPANY.CLASS = SAMPLEDETAIL.REFERENCEATTRCLASS AND'+sLineBreak+
      'COMPANY.id = SAMPLEDETAIL.REFERENCEATTRID';
    OrderByClause := 'COMPANY.DX DESC';
  end;
end;
```

La variabile SequenceNoFieldName per ora non è gestita: servirà quando si vorrà implementare un ordinamento sequenziale in base ad un campo, ma per ora manca il supporto di update di tale campo sulla tabella di dettaglio.

7. La GUI del MultiFramework

Il MultiFramework è basato su un concetto di “ereditarietà visuale” attraverso il quale sono state create alcune form di base utili allo sviluppo.
Il framework è basato su una form principale, (FmainXP o FMainModern) e da form “secondarie” che vengono invocate all'attivazione di una funzione (TISFunction) collegata ad una voce di menu (TISMenuItem).

7.1. Personalizzazione del layout della Main Form e di altri elementi grafici

Nella stessa cartella (exe) dove è presente l'eseguibile dell'applicazione esiste un file .ini (con lo stesso nome dell'applicazione) che definisce alcuni attributi dell'applicazione: vediamo il significato dei più importanti:

ApplicationStyle=MDI o SDI o TDI (controlla il layout delle finestre dell'applicazione)

ColorMap=Silver,Olive,Silver,Blue (colore delle toolbar)

HideMenuStructure=0 o 1 (nasconde il menu ad albero)

HideActiveFunctions=1 o 0 (nasconde la finestra delle funzioni attive)

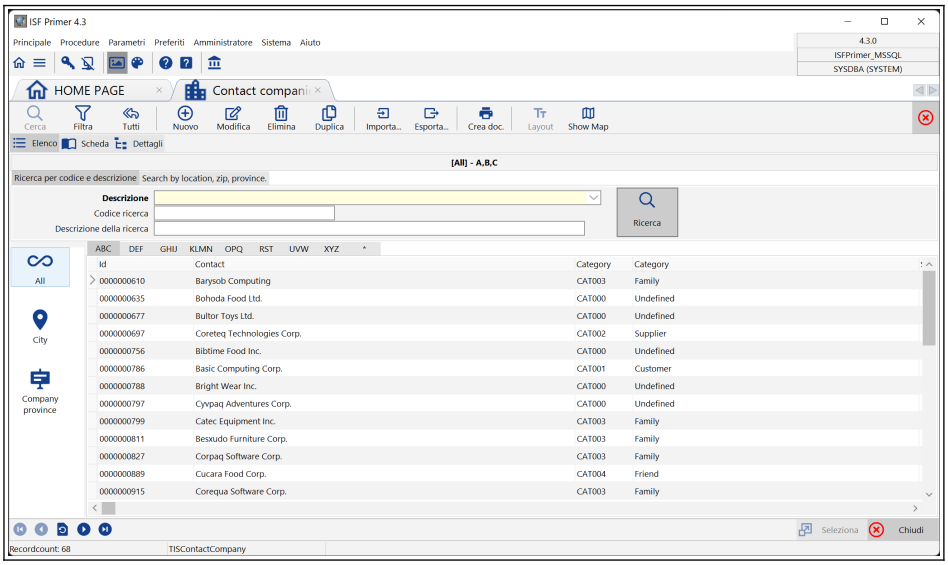
MenuStructureEmbedded=0 o 1 (menu ad albero a comparsa o fisso)

ActiveFunctionsEmbedded=0 o 1 (finestra delle funzioni attive a comparsa o fisso)

SQLMonitorEmbedded=0 o 1 (finestra del monitor SQL a comparsa o fisso)

HideMainToolbar=0 o 1 (nasconde la toolbar della form principale).

Ecco un esempio di applicazione TDI, con le finestre dentro i TAB:



7.2. Personalizzazione del layout di mappe ed elenchi

ISF fornisce un editor integrato per la personalizzazione delle mappe. Esso fa parte dei componenti di CBLib ed è molto comodo per definire facilmente e rapidamente un layout di mappa o di griglia, che viene memorizzatto al livello di dizionario.

- Connettersi come utente “system” e personalizzare le mappe delle persone e delle aziende.
- Andare su una Griglia di “elenco” o su una mappa, compare anche il pulsante “Layout”: premendolo si attiva l'editor del layout (per maggiori informazioni sull'uso dell'editor di layout utilizzate l'help di ISWorkbench alla voce “Layout editor per le mappe” e “Regole layout di mappa” oppure “Layout editor per le griglie” e “Regole layout di griglie”. Il risultato del layout viene memorizzato nel repository di ISWorkbench ed è visibile nella pagina “Layout”.
- Potete anche aggiungere degli sfondi alle mappe: provate con le mappe delle persone e delle aziende: basta aggiungere nella cartella “sfondi” un file jpg o bmp o Gif con il nome della classe (es. TISContact.gif).
- E’ possibile utilizzare i componenti NunmberBox in alternativa dei componenti CurrencyEdit impostando nel file ini nella sezione Framework:
  - NumberBoxEditorOptions=nbSmallInt,nbInteger,nbCurrency,nbFloat

N.B. alcune dimensione di componenti hanno una dimensione di default diversa da ISF7, per agevolare la creazione di layout basati su “colonne” multipli di 4: Currency=12; Date=12; DateTimeHHMM = 16; Time=8; TimeHHMM=8; Float=12, GUID=28, TimeStamp=16)

7.3. Il ruolo di ActiveDataSet, ActiveClass, CurrentObject

E' importante sapere che il framework funziona su un concetto di DataSet attivo in un certo contesto (e di conseguenza di una “ActiveClass” attiva.

Le form di base del framework sono già create in modo tale da impostare correttamente il DataSet attivo quando per esempio si cambia pagina o si cambia nodo sull'explorer della pagina dei dettagli.

Oltre ad ActiveDataSet il sistema agisce anche in base al concetto di ActiveClass (la classe InstantObject attiva in un certo contesto che è quasi sempre la classe controllata dall'ActiveDataSet).

Infine è sempre possibile utilizzare la funzione CurrentObject per sapere qual'è l'oggetto corrente (attenzione al fatto che potrebbe essere nil) della classe attiva.

7.3.1. Sfruttare l'ereditarietà visuale del MultiFramework

Il concetto di ereditarietà visuale può essere molto utile per customizzare funzionalità del framework.

Occorre ricordarsi che nelle form ereditate l'ActiveDataSet/ActiveClass sia sempre coerente al contesto in cui ci si trova: per questo motivo sono stati predisposti dei meccanismi automatici.

7.3.2. Impostare automaticamente ActiveDataSet e ActiveClass

Quando si sviluppano delle form “custom” che ereditano è necessario ricordarsi sempre di fornire al Framework l’indicazione dell’ActiveDataSet (e di conseguenza dell’ActiveClass) attiva in un certo contensto. Dato che spesso la nuova GUI si basa su DbGrid o su DBMultiEdit, vengono forniti due eventi già pronti da agganciare all’event-handler “OnEnter” delle DbGrid e DBMultiEdit che si vanno a creare nella form “custom”:

- DBGridEnterChangeDataSet
- MultiEditEnterChangeDataSet

basterà quindi agganciare tali eventi per cambiare l’ActiveDataSet.

7.3.3. Impostare ActiveDataSet al cambio pagina

Se si crea un nuovo Selector/Exposer per mostrare i suoi dati in una pagina di dettaglio, è importante fare in modo che al cambio di quella pagina, l'ActiveDataSet dovrà essere impostato nell'evento OnChange della pagina.

es. nella form FAccessRole è stata aggiunta una pagina che mostra i profili utilizzati dal ruolo corrente. In questa pagina viene mostrata una griglia collegata al Selector ioslProfiles pertanto è stata implementata:

```
procedure TformAccessRole.pgctDataChange(Sender: TObject);
begin
  inherited;
  if pgctData.ActivePage = tsProfile then
  begin
    OpenProfiles;
    ActiveDataSet := ioslProfiles;
  end;
end;
```

In questo modo cambiando la pagina cambia l'ActiveDataSet (e di conseguenza l'ActiveClass e il CurrentObject) e l'interfaccia utente è coerente con questo cambiamento.

7.3.4. Esempio di form “custom”

Proviamo invece ad ereditare dalla form TformDataSideBar per realizzare la form dei contatti che deve essere in grado in caso di inserimento o modifica di un oggetto di decidere quale form attivare in base alla classe (TISContactPerson o TISContactCompany): salviamo la unit FContact contenete la form TfmContact. Togliamo la “var” globale e registriamo la classe per renderla disponibile a run-time. (N.B. Con questa operazione il file dpr viene completamente riscritto perdendo tutti i commenti utili, conviene riprostinarlo dal backup).

Di default, quando di edita un oggetto viene utilizzata la form stessa, ma è anche possibile abilitare una form di edit alternativa, indicandola nella “funzione” di abilitare la form di modifica “modale”.

Se l'oggetto da editare è di una classe “derivata” il framework è già in grado di abilitare l'editing sulla classe corretta. Ciò che non può fare è sapere in caso di inserimento di quale classe “derivata” si intende inserire l'oggetto.

Nella form customizzata dobbiamo implementare il metodo:

```
uses
  FCBSelectOptions, FChild, MContact, MPerson, MCompany;
{ TfmContacts }
function TfmContacts.GetInsertClassName: string;
var
  Selection : integer;
begin
  //Chiedo di quale classe si vuole creare l'ogetto del contatto
  if ActiveClass = TISContact then
  begin
```

```
    Selection := CBSelectOption(
      'Inserimento Persona'+sLineBreak+
      'Inserimento Azienda'+sLineBreak,
      Font, False, 'Seleziona il tipo di contatto');
    case Selection of
      0 : Result := TISContactPerson.ClassName;
      1 : Result := TISContactCompany.ClassName;
    else
      Abort;
    end;
  end
  else
    Result := inherited GetInsertClassName;
end;
```

7.3.5. Personalizzare una stampa ad-hoc

Proviamo anche a creare una stampa di tutte le province in cui esiste almeno uno dei nostri contatti. Come dicevamo al campitolo precedente sulle stampe è possibile cambiare il meccanismo di default delle stampe.

Nella nostra nuova form customizzata, aggiugiamo un selector dal nome **"ioslProvince"** e impostiamo la proprietò command in questo modo:

```
SELECT * FROM TISProvincia
WHERE EXISTS (SELECT * FROM ANY TISContact WHERE Provincia <> '' USING Provincia)
ORDER BY Descrizione'
```

Questa query estrae tutte le province che sono referenziate da almeno un contatto dalla proprietà "provincia".

Impostiamo anche:

```
ContentDescription = 'Province in cui sono presenti dei contatti'
```

Vogliamo quindi creare un Report con OpenOffice il cui codice sarà: "PROVINCECONTATTI": siccome il "design" del report avviene tutto a run-time, prepariamo la nostra form ad indicare che nel caso in cui il report dovesse essere questo, i dati andranno recuperati dal selector **"ioslProvince"**, sia in fase di "creazione del Layout" che in fase di "stampa".

Per far ciò dobbiamo "ereditare" due metodi e implementarli in questo modo:

```
function TfmContacts.DoCreateDocument(const DocReportId: string;
  DataSet: TDataSet; Mode: TCBDocProducerMode; MultiDocument: boolean): string;
var
  DataSetToPrint: TDataSet;
begin
  if SameText(docReportId,'PROVINCECONTATTI') then
    DataSetToPrint := ioslProvince else
    DataSetToPrint := DataSet;
  Result := inherited DoCreateDocument(DocReportId,
    DataSetToPrint, Mode, MultiDocument);
end;

procedure TfmContacts.DoCreateTemplate(const DocReportId: string; DataSet: TDataSet);
var
  DataSetToPrint: TDataSet;
begin
  if SameText(docReportId,'PROVINCECONTATTI') then
    DataSetToPrint := ioslProvince else
    DataSetToPrint := DataSet;
  inherited DoCreateTemplate(DocReportId, DataSetToPrint);
end;
```

Come si può vedere questi 2 metodi forniscono semplicemente il DataSet appropriato. A Run-time seguire le indicazioni su come creare una stampa (vedi capito **"Come si creano le stampe con ISF"**), ricordando di assegnare questi valori:

Modello: "PROVINCECONTATTI" (è il campo Id)

Descrizione documento: "Province dei contatti"

Nome File: "ProvinceContatti"

Tipo: "D" (Documento di testo Open Document).

Classe di riferimento: **"TISContact"** ***N.B. È la classe della form che ha avviato la stampa!***

Usa sorgente dati: **"SI"** ***N.B. Per evitare che venga creato un Selector/Exposer alternativo***

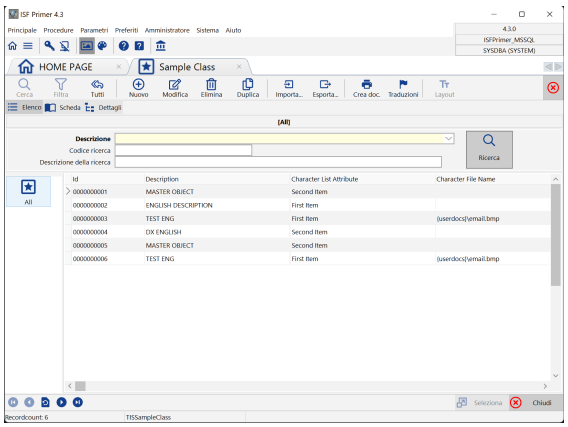
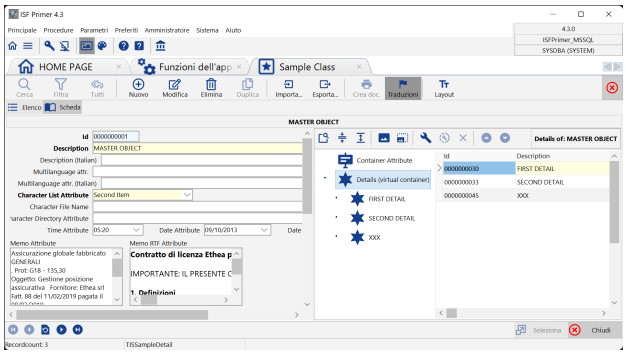
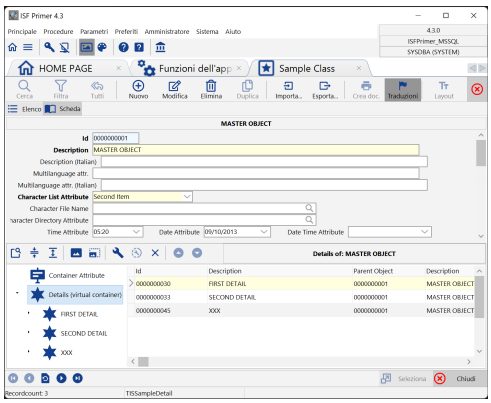
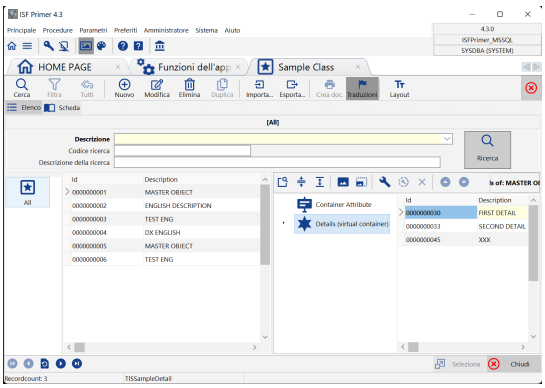
Tabulato: **"SI"** e Numero Records: **-1** ***Per fare un tabulato orientato orizzontalmente.***

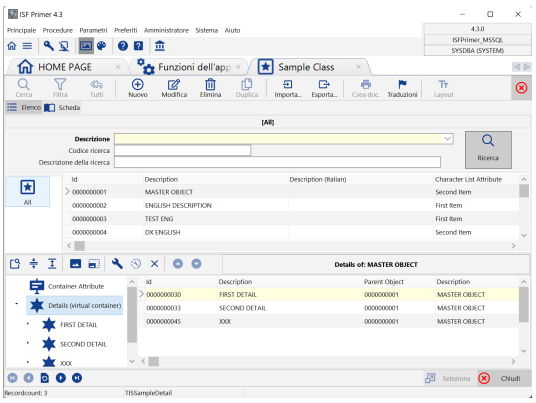
Salvare, e generare il template (premendo sul pulsante "Layout") selezionando tutti i campi della Provincia. Quindi provare a generare il documento OpenOffice di tutte le province dei contatti.

7.3.6. Personalizzare il layout della form standard TformDataStd (unit FDataStd.pas)

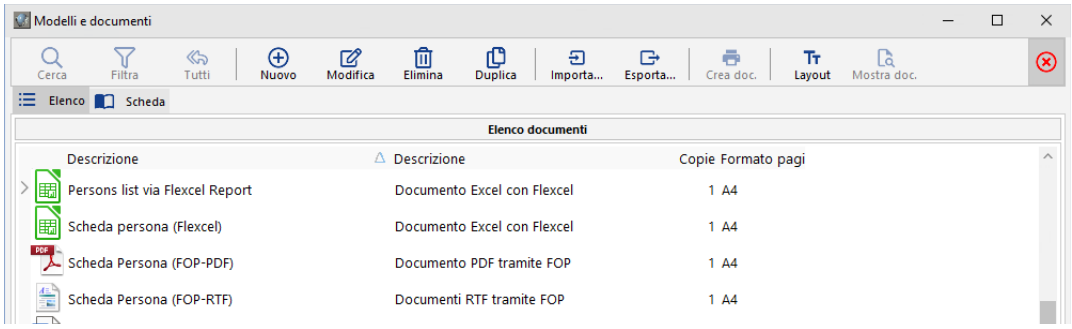
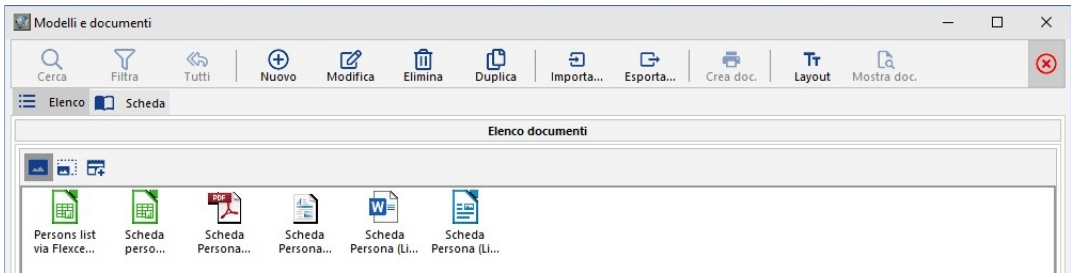
La form TformDataStd inclusa nel Framework espone alcune proprietà published per personalizzarne il “layout” delle varie parti. Normalmente la Form TformDataStd è costituita da 3 pagine: Elenco, Scheda, Dettagli.

EmbeddedExplorerPosition può assumere questi valori:

eepDetailPage: Explorer nella pagina Dettagli (default)	
eepDataRight: Explorer nella pagina dati a destra della scheda	
eepDataBottom: Explorer nella pagina dati sotto la scheda	
eepListRight: Explorer nella pagina elenco a destra dell'elenco	

eepListBottom: Explorer nella pagina elenco sotto l'elenco	

E' possibile anche utilizzare il componente DbListView anziché la DbGrid negli elenchi, ad esempio per la classe GSTD_TISDOCREPORT, utilizzando il paramentro **BrowsingMode** che può assumere questi valori:

BmDbGrid: Elenco tramite la DbGrid (default)	
bmDbListViewReport bmDbListViewSmallIcons bmDbListViewLargeIcons: Elenco tramite DbListView (con ViewStyle diversi)	

Una volta ereditata la form si possono personalizzare praticamente tutti gli aspetti di una applicazione ISF, utilizzando gli appositi metodi virtuali messi a disposizione dalle classi di partenza.

Per cambiare queste impostazioni ci sono 2 modalità:

1. Definire nel file .ini nella sezione [FrameWork] i valori, ad esempio:

```
EmbeddedExplorerPosition=eepDataRight
```

Questa impostazione diventa il default per ogni form del framework.

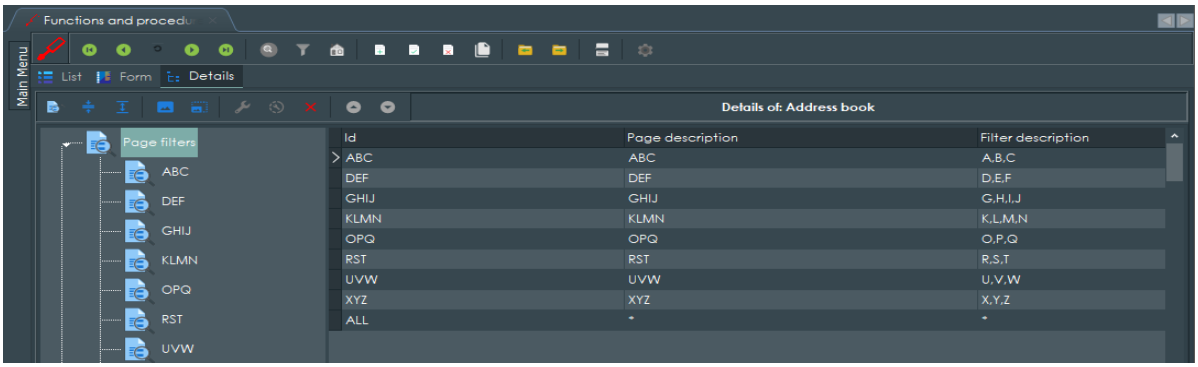
2. Oppure indicare questi valori dentro la ISFunction nel campo: "Parametri opzionali": basta scrivere ad esempio **EmbeddedExplorerPosition=eepDataRight**

7.3.7. Organizzare la pagina di dettaglio in sottopagine

In InstantSolutions è possibile anche definire nelle funzioni che l'Explorer venga suddiviso in diverse pagine, indicando quali nodi di primo livello devono essere visibili per ciascuna pagina.

Occorre utilizzare la gestione delle funzioni e definire il contenitore "DetailPages" (Pagine di dettaglio) andando ad indicare:

Nome della pagina, elenco dei contenitori ammessi (separati da ';') e eventuale immagine da mostrare, come si vede da questo esempio:



7.3.8. Customizzare altri attributi di GUI delle form

E' possibile anche personalizzare moti altri aspetti grafici di una finestra del MultiFramework, utilizzando i “parametri opzionali” della classe TISFunzione, utilizzata per avviare la finestra. Connettersi come utente con privilegi di sistema per poter modificare le classi TISFunzione.

Provare a modificare la funzione GSTD_TISContactPerson indicando in “parametri opzionali”: Font.Color=255 (il numero decimale del colore rosso). Ora la finestra dei contatti di tipo “persona” apparirà con il font rosso.

7.4. Utilizzo della form “nativa” per l'accesso veloce a liste di dati

In alcune situazioni in cui le performances sono particolarmente critiche, è possibile anche accedere ai dati “bypassando” InstantObjects. E' sufficiente utilizzare la form “nativa” fornita con il framework.

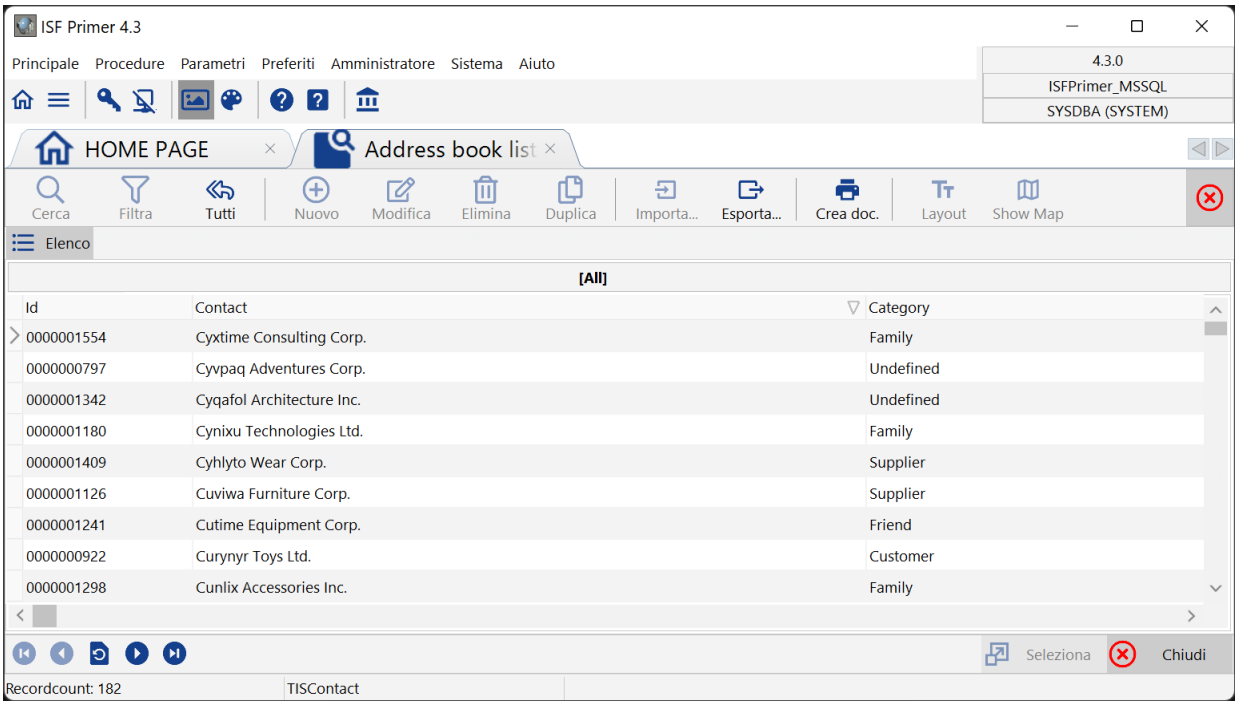
Creiamo una funzione nuova:

```
Function Id=GSTD_CONTACTS_NATIVE
Description=Address book
Management form=TformNativeBrowse
Optional parameters=ClientDataSet.PacketRecords=100
InstantObjectClass=TISContact
ImageIndex=113
```

Agganciamola quindi al menu nella sezione “Address book contacts”: avremo a disposizione una form che accede in modo diretto ai dati attraverso FireDAC, sfruttando la stessa connessione definita a livello di ISWorkBench.

In questa form “nativa” è possibile anche specificare “Filtri di pagina” e “Ruoli” come si fa per le altre “funzioni”, ma bisogna fare attenzione al fatto che le espressioni dei filtri devono essere fatte in “SQL nativo” e non in “IQL”.

Ecco come si presenta la form “nativa” del progetto di esempio con impostati anche i ruoli e i filtri di pagina.



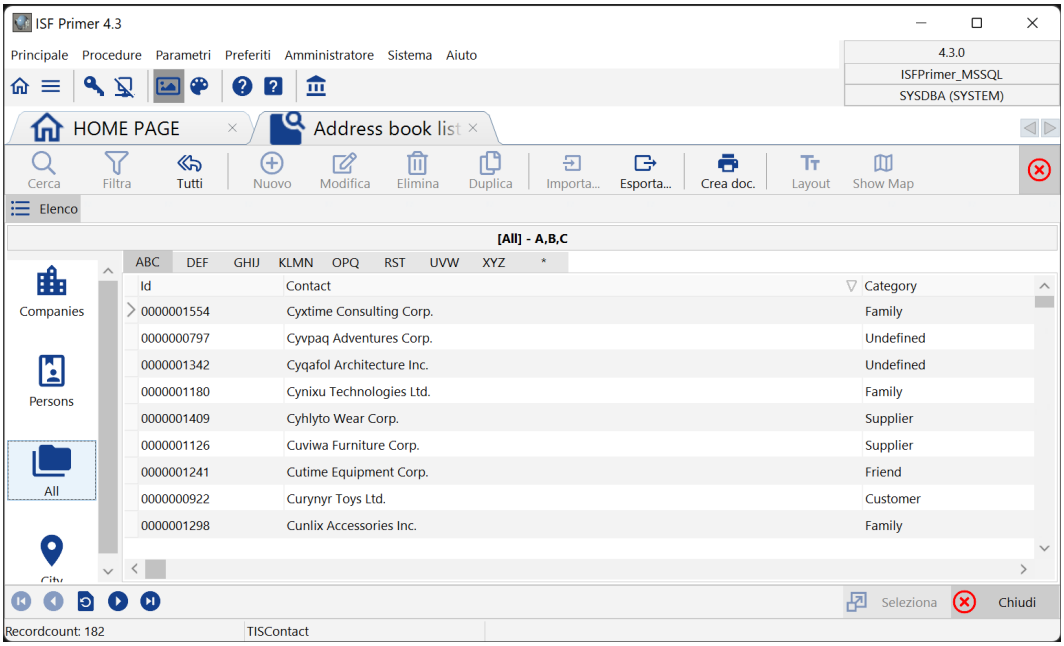
7.5. La modalità QuickSearch

La modalità QuickSearch consiste nella possibilità di utilizzare le classi di ricerca predefinite e/o generiche

direttamente sulla form di “elenco”, sotto forma di diverse pagine (una per ogni classe di ricerca impostata) più eventualmente la pagina delle ricerche “speciali” cioè quelle generiche.

In ogni pagina è possibile sia selezionare una ricerca già salvata (dalla combobox della Descrizione) sia inserire a mano i valori di ricerca e premere il pulsante di Ricerca.

E' anche possibile customizzare il layout del QuickSearch perché il pannello ha come layoutname “QUICK_SEARCH”, perciò si potrebbe anche decidere che a livello di QuickSearch vengono visualizzati solo i campi più utilizzati, oppure spegnerli tutti e lasciare solo la combobox di selezione delle ricerche già memorizzate.



Il meccanismo di Quick può essere abilitato sia nelle form standard che in quelle native in diversi modi:

1) variabile globale: vcAutoQuickSearch è una nuova variabile globale che è possibile impostare nel file .ini dell'applicazione o da codice Delphi. Se viene impostata a True in automatico tutte le classi di ricerca vengono anche mostrate come QuickSearch: vcAutoQuickSearch è di default a False.
2) Tramite class procedure sulla classe “source” di ricerca: è possibile implementare il metodo IsQuickSearch per cambiare il default rispetto al valore di vcAutoQuickSearch (es. vcAutoQuickSearch è false ma ho una classe di ricerca che voglio usare come quicksearch di default).
3) Tramite class procedure sulla classe “target” della ricerca: a prescindere dal valore di vcAutoQuickSearch e del metodo IsQuickSearch si può ereditare il metodo di classe CanQuickSearch e decidere se abilitare/disabilitare il quick search per tutte le classi di ricerca o solo per alcune. Nell'esempio la classe TISContactPerson abilita il QuickSearch a tutte le sue classi tranne alla TISselectId_Desc.

```
class procedure TISContactPerson.CanQuickSearch(
  SearchClass: TInstantObjectClass; var IsQuickSearch: boolean);
begin
  if SearchClass = TISSelectId_Desc then
    IsQuickSearch := False
  else
    IsQuickSearch := True;
end;
```

4) Siccome il meccanismo è legato alla classe potrei aver bisogno di spegnere il QuickSearch su funzioni diverse che usano la stessa classe, pertanto esiste anche una Property a livello di Form (ShowQuickSearchFrame) per poterlo spegnere a impostandolo nei parametri della Funzione.

7.5.1. Condizionare l'utilizzo delle classi di ricerca e QuickSearch

E' possibile condizionare la visibilità e quindi l'utilizzo delle classi di ricerca solo in certi contesti.

Utilizzando i metodi IsQuickSearch e IsFilterSearch sulla classe di ricerca e i rispettivi CanQuickSearch e CanFilterSearch sulle classi target si possono avere diverse combinazioni e stabilire se una classe di ricerca è visible solo come Quick e come filtro a prescindere dalla classe target si utilizza IsQuickSearch o IsfilterSearch. Se invece l'utilizzo è condizionato dalla classe target si utilizza CanQuickSearch e CanFilterSearch sulla classe target.

7.6. Customizzare le azioni sulle classi (CustomActions)

E' possibile evitare di ereditare form se si devono solamente aggiungere delle azioni a certe classi. Esiste un meccanismo di “Custom Actions” che è possibile implementare a livello di classe: l'azione custom viene resa disponibile sulla toolbar delle azioni, di fianco a quelle standard.

Per esempio, se vogliamo implementare una azione “send e-mail” dalla classe TISEMail, dobbiamo procedere a definire 4 nuovi metodi nella unit MEMail.pas:

```
    class function GetCustomActionCount : integer; override;
    class procedure AssignCustomAction(CustomActionId: Integer; Action: TCBAction); override;
    procedure CustomActionOnExecute(CustomActionId : Integer; Action : TCBAction); override;
    procedure CustomActionOnUpdateDataSet(CustomActionId : Integer; Action : TCBAction;
      ActiveDataSet: TDataSet); override;
```

Con questi metodi si informa il framework che è disponibile un certo numero di CustomAction.

E' buona cosa utilizzare una costante (es. ACTION_SEND_EMAIL = 0) per definire un indice relativo alla CustomAction da utilizzare in tutti i metodi così:

```
class procedure TISEmail.AssignCustomAction(CustomActionId: Integer; Action: TCBAction);
begin
  inherited;
  if CustomActionId = ACTION_SEND_EMAIL then
  begin
    Action.Caption := 'Send E-Mail';
    Action.Hint := 'Send E-Mail to current address';
    Action.ImageIndex := IMG_POSTA;
    Action.ShortCut := ShortCut(VK_F12, []);
  end;
end;
```

Nel metodo di Update si attiva l’azione in base ad alcune condizioni: in questo caso si utilizza il metodo CustomActionOnUpdateDataSet che fornisce anche l’ActiveDataSet:

```
procedure TISEmail.CustomActionOnUpdateDataSet(CustomActionId: Integer;
  Action: TCBAction; ActiveDataSet: TDataSet);
begin
  inherited;
  if (CustomActionId = ACTION_SEND_EMAIL) then
  begin
    //Disabilito l'azione se non c'è l'indirizzo e-mail o se il dataset è in editing
    Action.Enabled := (Description <> '') and (ActiveDataSet.State = dsBrowse);
  end;
end;
```

Allo stesso modo il metodo di Execute sarà:

```
procedure TISEmail.CustomActionOnExecute(CustomActionId: Integer; Action: TCBAction);
```



```
var
  Msg: string;
begin
  inherited;
  //Apre il client di posta
  if CustomActionId = ACTION_SEND_EMAIL then
  begin
  Msg := 'mailto:'+Description;
    ShellExecute(0, 'open' , pChar(Msg), nil, nil, SW_SHOW );
  end;
end;
```

Il framework mostrerà la CustomAction come pulsante sulla toolbar in modo dinamico, ogni volta che cambia l'ActiveClass o l'oggetto corrente.

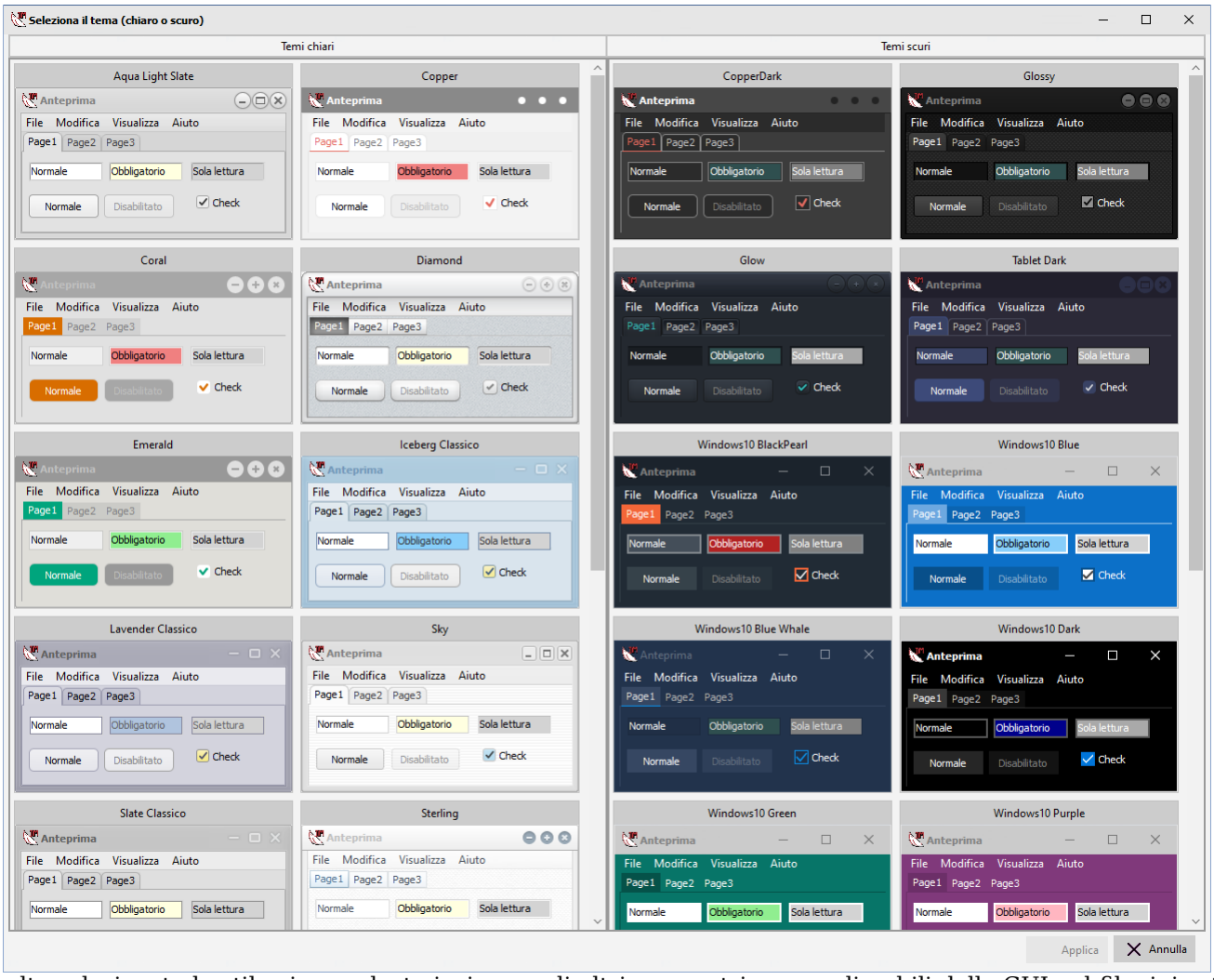
7.7. Gestione dei lock sui dati

Dalla versione 6.5.6 è stata introdotta la possibilità di gestire i lock sui dati, sfruttando la classe TISCBObjectLocker: fare riferimento al documento **Migrazione_da_ISF_6.5.3_a_ISF_6.5.6.pdf** per capire come funziona.

7.8. Utilizzo degli "stili grafici": tema chiaro o scuro

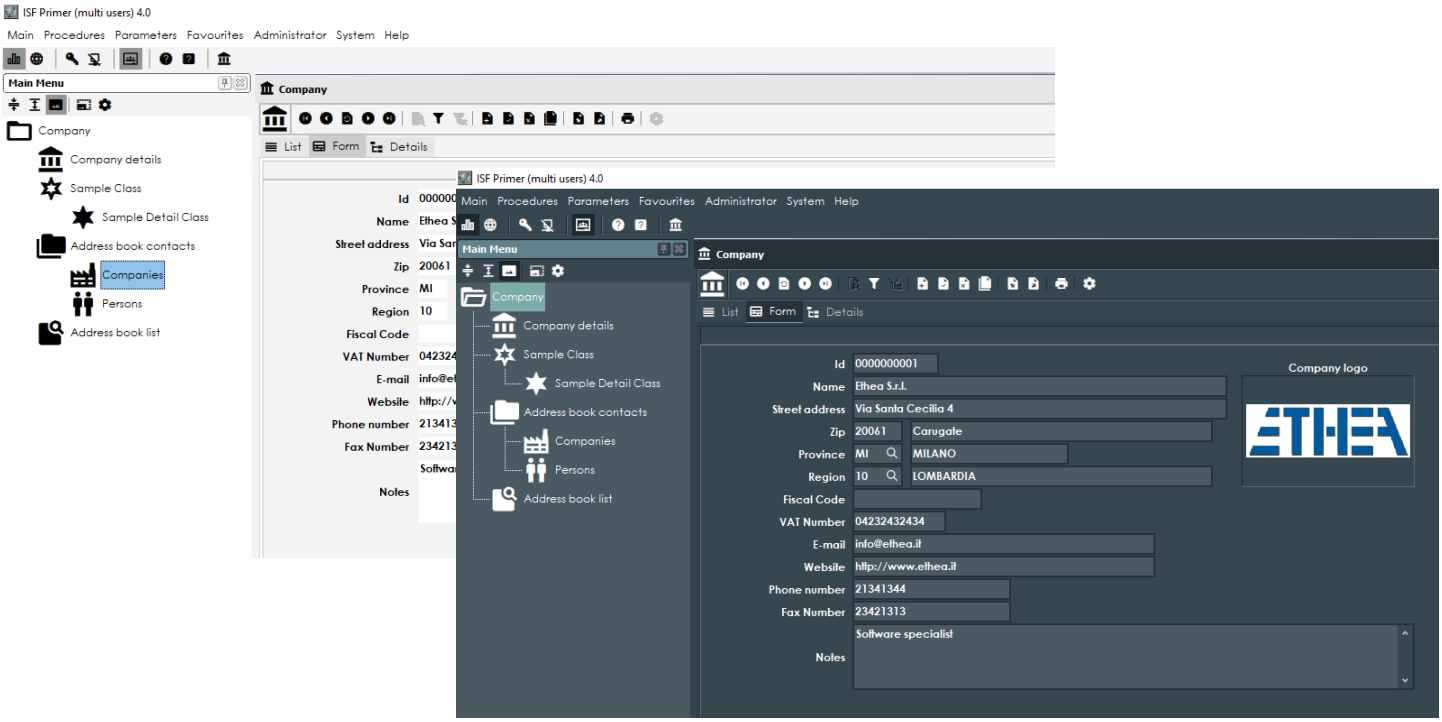
Con la versione 8 e il supporto alle icone attraverso il componente SVGIconImageList, è possibile utilizzare al meglio gli "stili grafici" per una applicazione. (il supporto alle icone tramite Font che era disponibile con ISF7 è stato soppresso).

Per sfruttare questa nuova caratteristica nel framework MultiWindow occorre abilitare l'applicazione all'uso degli stili (Project/Options/Application/Appereance) e selezionare gli stili che si vogliono rendere disponibili all'utente finale. Automaticamente l'applicazione fornisce la selezione dello stile nella finestra di Configurazione dell'applicazione, oppure attraverso la funzione di scelta dello stile disponibile nella barra del menu principale (pulsante della tavolozza colori):



Una volta selezionato lo stile viene salvato insieme agli altri parametri personalizzabili della GUI nel file .ini sotto la voce VCLStyle della sezione [Framework]. Ecco come appare ISFPrimer con 2 stili diversi e l'uso delle icone

IconFonts che si adattano diventando bianche o nere:



7.9. Gestione Icone SVG e IconName nei dfm

Con la versione 8 è stato raggiunto il pieno supporto High-DPI multimonitor.

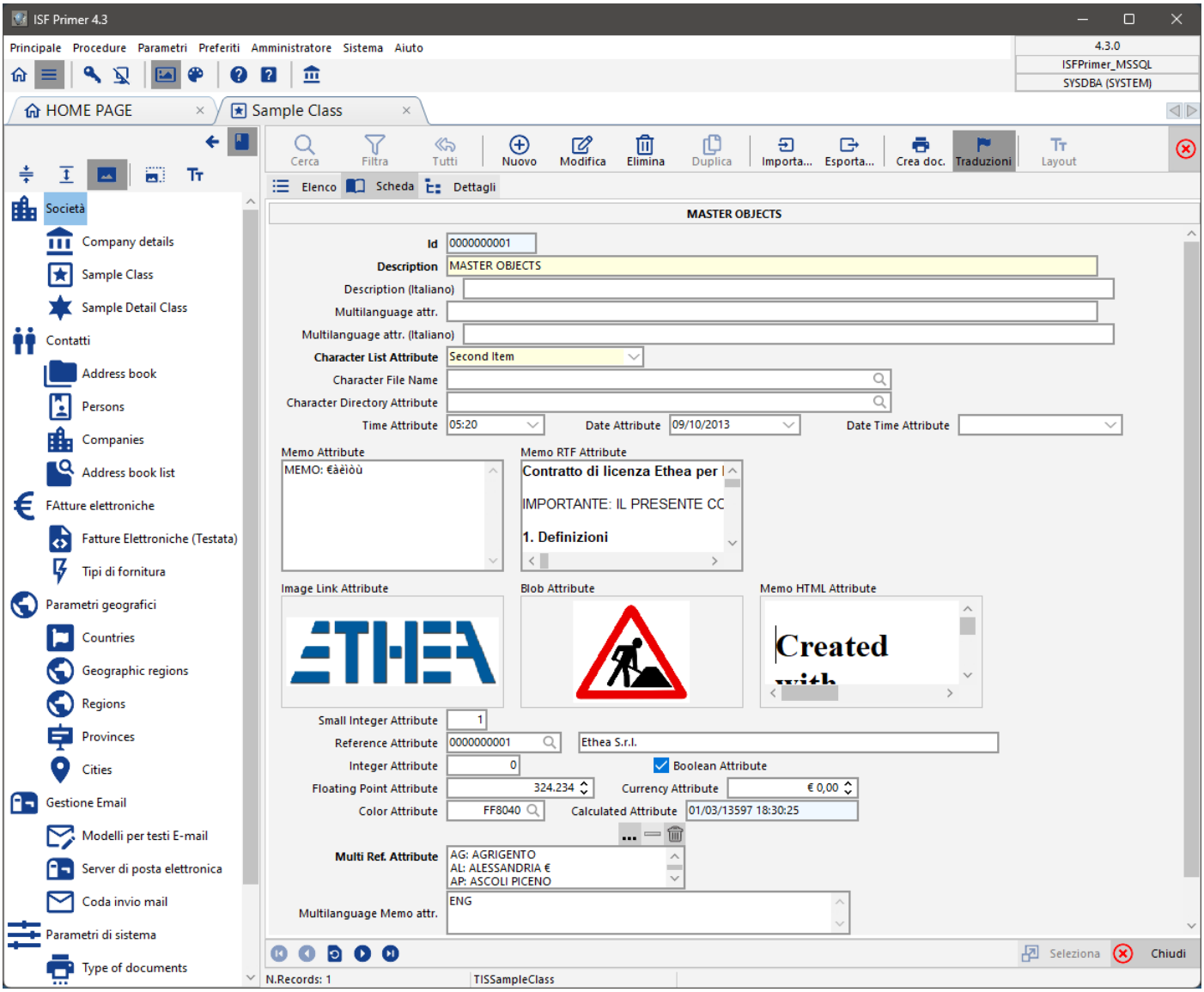
Questo aspetto ha obbligato a cambiare tutta la logica delle Icone che ora sono collezionate nel datamodule TdmISFResources (DISFEResources.pas) attraverso un componente TSVGIconImageCollection.

N.B. con la versione 8 non si possono più utilizzare i Font di Icone, perciò il supporto a TiconFontImageList è stato rimosso!

Per consentire di spostare le form anche in monitor con DPI diversi si è reso necessario, in tutte le form del framework e della libreria CBLib, introdurre il concetto di “VirtualImageList”, che raccoglie solo le icone necessarie alle Action di quella form dalla collection contenuta nel datamodule TdmISFResources.

Allo stesso tempo, tutte le Action e gli elementi della GUI non puntano più alle Icone per “ImageIndex” ma per “ImageName”, e questo concetto è stato esteso anche alle classi del framework (es. TISMenuItem e TISFunction) che prima utilizzavano ImageIndex.

Esempio di maschera con le Icone SVG (monocromatiche)



7.10. Utilizzo della form Wizard del MultiFramework

7.10.1. Derivare da TFormWizard

Il form Wizard presente nel MultiFramework è una form pensata per gestire operazioni in “sequenza”. Occorre ereditare dalla form TformWizard ed implementare alcuni metodi virtuali come:

```
procedure NextPage; override;
procedure PriorPage; override;
procedure ExecuteWizard; override;
procedure OnPageChange; override;
procedure ExecuteWizard; override;
procedure ExecuteOK; override;
```

Creare un Wizard non è comunque una operazione semplicissima ed occorre sviluppare un po' più del solito ;-)

La cosa importante da sapere è che si devono utilizzare componenti TInstantSelector e TInstantExposer per manipolare gli oggetti da presentare sull'interfaccia utente, oltre all'uso di CBXDbMultiEdit e CBXDbGrid che offrono tutta la flessibilità dell'editor di mappe e griglie già vista.

7.10.2. Un esempio di Wizard: la form di esportazione

Possiamo analizzare la form di esportazione dati TformExportWizard per vedere come è stata utilizzata la form di base del Wizard.

Ogni Tabsheet creato deve avere Caption e Hint valorizzati, che vengono mostrati sulla parte alta del wizard.

Con i metodi NextPage, PriorPage e OnPageChange si controllano gli spostamenti tra le pagine.

Con il metodo ExecuteWizard si esegue l'elaborazione.

Questo Wizard utilizza le tecnologie di esportazione di un dataset in vari formati contenute in CbLib (vedi capitolo

relativo all'esportazione dei dati).

7.11. Utilizzo della HomePage e Modifica dati profilo

La versione 7.6 si differenzia dalla versione 7.5 per alcuni aspetti:

- Aggiunto il supporto all'autenticazione Windows per connettersi a MS-SQL
- Aggiunta la gestione della HomePage per organizzare meglio il menu
- Aggiunta form di modifica dati del proprio account

7.11.1. Aggiunta form di editing del proprio Account

Con la versione 7.6 è stata aggiunta una voce di menu "Account/Profile" (Profilo/Account) nel menu "Main" (Principale) per permettere all'utente di modificare alcuni dati del suo account. Per questo motivo va aggiunta questa unit al progetto:

```
  FAccount in '..\..\..\src\GUIMulti\FAccount.pas' {FormAccount},
```

e va copiato il file:

```
{ISF7_6}\ISFPrimer\Config\ISFUNCTION\TISFunction.GSTD_TISACCOUNT.1.xml
```

facendo attenzione a modificare le descrizioni della funzione in italiano (perché in ISFPrimer sono in inglese).

7.11.2. Utilizzo di una schermata di HomePage

Dalla versione 7.6 è disponibile una nuova form del framework contenuta nella unit FHomePage.pas che va aggiunta al dpr per poter compilare:

```
  FHomePage in '..\..\..\src\GUIMulti\FHomePage.pas' {fmHomePage},
```

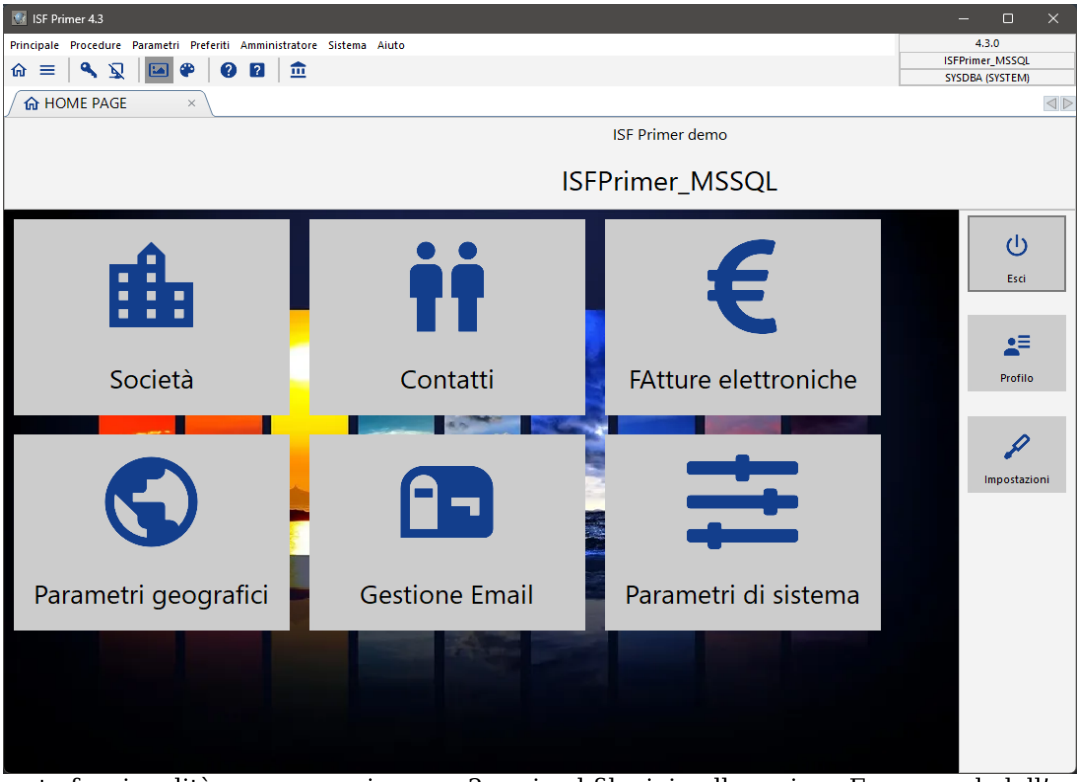
e va copiato il file:

```
{ISF}\ISFPrimer\Config\ISFUNCTION\TISFunction.GSTD_TISHOME.1.xml
```

facendo attenzione a modificare le descrizioni della funzione in italiano (perché in ISFPrimer sono in inglese).

Con questa form è possibile mostrare una serie di pulsanti, uno per ogni "nodo" principale del menu ad albero: cliccando su di essi è possibile aprire il menu filtrato con le sole voci di questo menu: nell'esempio si vede il menu filtrato quando si clicca su "Company".

Il pulsante "Profilo" avvia la nuova maschera di modifica dati del proprio account, mentre "Impostazioni" lancia la finestra di configurazione (se l'utente è amministratore).



Per abilitare questa funzionalità occorre aggiungere 2 voci nel file .ini nella sezione Framework dell'applicazione:

```
[FrameWork]
UseHomePage=1
HomePageButtonWidth=200
```

La dimensione dei pulsanti va "calibrata" in base a quale voci di menu di primo livello ci sono nel vostro menu.

Il logo da mostrare in alto a sinistra della HomePage va configurato nella sezione:

```
[Application]
```

```
HomePageBanner=NomeFile.png
HomePageWallPaper=NomeFile.png
```

7.12. Autenticazione di Windows (single signon)

Ora è possibile attivare l'autenticazione di Windows quando ci si connette ad un database MS-SQL.

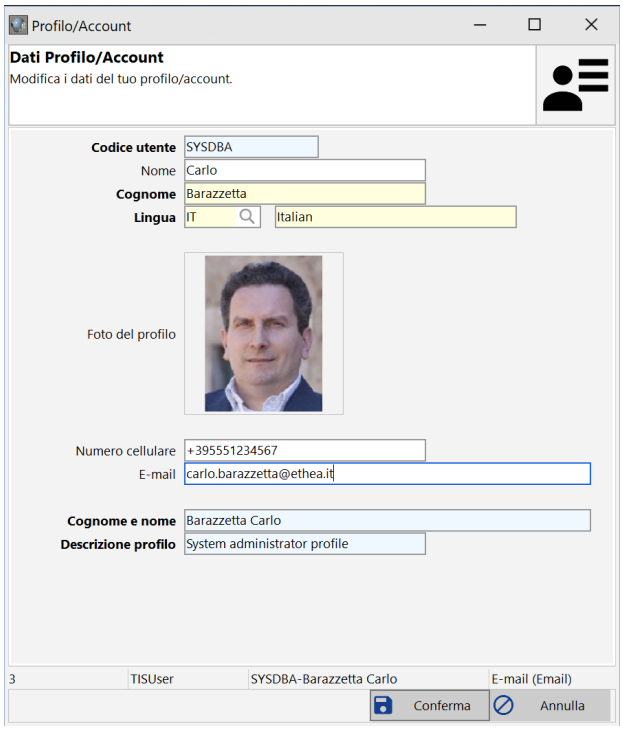
La nuova form di login prevede un checkbox che si abilita quando si sceglie un database MS-SQL: cliccando su "Windows Authentication" si disabilita la possibilità di indicare lo user name (che viene recuperato da quello di windows) e la password.

Il sistema accede al database con Autenticazione Windows, dopodiché cerca comunque l'utente Domain\User nella tabella degli utenti, ignorando la password indicata nella tabella medesima.

7.13. Nuova maschera "Profilo Account"

E' stata aggiunta nel menu principale (sotto "cambia password") la possibilità di modificare i dati del proprio profilo utente come Nome, Cognome, Lingua, Foto, Numero di telefono, email.

La stessa maschera si attiva dal pulsante "Profilo" della HomePage.



7.14. Utilizzo di moduli forniti con il Framework

All'interno del framework ISF si trovano anche dei moduli applicativi già pronti, con una serie di classi e di librerie utili per funzionalità generiche di largo uso.

Alcuni esempi sono i moduli:

- MGeographic: contiene una serie di classi di dati geografici (città, province, regioni, paesi, ecc…) ultili per la gestione degli indirizzi

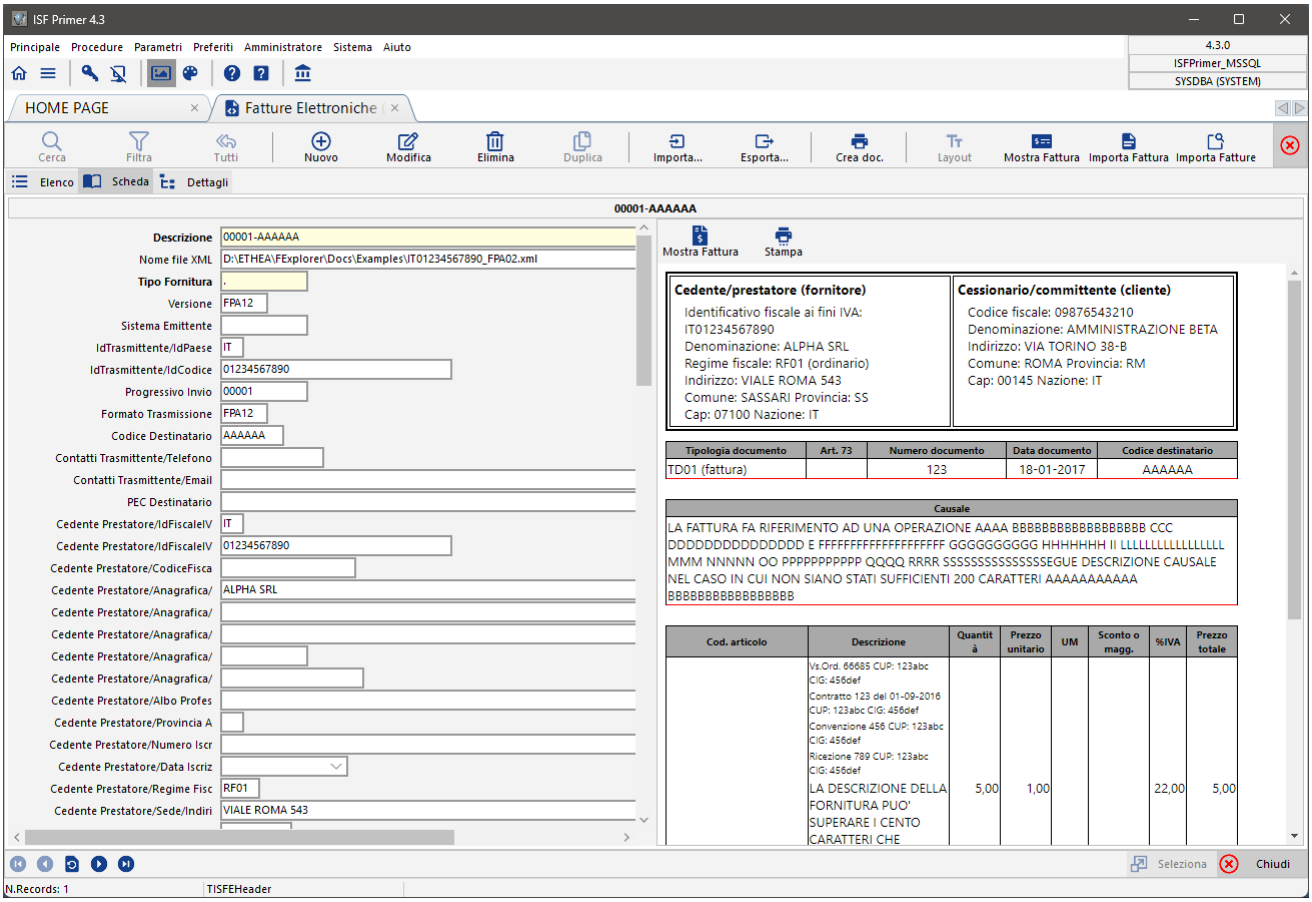
- MMultiLanguage: contiene una classe utile per la traduzione di messaggi applicativi

- MFatturaElettronica: contiene una serie di classi e di librerie per la gestione delle fatture elettroniche, rilasciato con la versione 8.5.

7.15. Nuovo Modulo FatturaElettronica (versione 8.5 di ISF)

La gestione dei dati di fatture elettroniche ormai in Italia è trasversale a diversi applicativi che si scambiano i file xml o pm7 che contengono le fatture elettroniche e gli allegati. Dalla versione 8.5 di ISF è disponibile il nuovo modulo "Fatture Elettroniche" completo di una libreria di importazione ed esportazione dati, oltre alla mappatura sulle classi di InstantObjects e la visualizzazione nel Browser "Embedded" grazie alla nuova form del framework: FNativeBrowseForm.

Nell'immagine il modulo con la gestione della Fattura Elettronica integrato in ISFPrimer:



Per aggiungere il modulo “Fatture Elettroniche” nella propria applicazione occorre farlo in diversi ambiti:

1) Aggiungere il modulo in ISWorkBench

2) Aggiungere tutte le unit di modello nel .dpr

3) Aggiungere le dll di deploy nella cartella dell’eseguibile (libeay32.dll, ssleay32.dll, webview2loader.dll) e installare il WebViewer di Microsoft (https://developer.microsoft.com/it-it/microsoft-edge/webview2/#download-section)

4) Aggiungere la path ..\..\..\ext\FatturaElettronica\Src alle search path di progetto

5) Aggiungere le classi e le unit di progetto (vedere sezione “Fatture elettroniche Models” e “Form e datamodules della libreria Fattura Elettronica” nel file di progetto di ISFPrimer

6) Aggiungere la classe TpathConfiguration (e la unit MPathConfiguration) per la gestione delle path “custom”.

7.16. Nuovo Modulo InvioMail (versione 8.5 di ISF)

Un’altro nuovo modulo aggiunto alla versione 8.5 del framework riguarda il supporto all’invio mail, compresa la gestione delle code in uscita e dei template per definire i “corpi” dei messaggi delle e-mail.

Per aggiungere il modulo “Invio Mail” nella propria applicazione occorre farlo in diversi ambiti:

1) Aggiungere il modulo in ISWorkBench

2) Aggiungere tutte le unit di modello nel .dpr

3) Aggiungere le dll di deploy nella cartella dell’eseguibile (libeay32.dll, ssleay32.dll)

4) Aggiungere le classi e le unit di progetto (vedere sezione “InvioMail Models”

8. Configurazione dell'applicazione e accesso ai dati

8.1. Il file di configurazione dell'applicazione

Attraverso il file **NomeApplicazione.ini** dell'applicazione è possibile intervenire su diversi aspetti del framework:

```
[Application]
Wallpaper=Wallpaper.bmp
Splash=Splash.jpg
LoginSplash=Splash.jpg
HelpFile={app}\..\WebHelp\Index.htm
DictionaryPath={app}\..\Dictionary\
CacheDictionaryClasses=1

[MultiLanguage]
;If you have Ethea Translation Tools for ISF applications, set your application languages
Languages=ENG;ITA;ESP
;UpdateRepository=yes

[FrameWork]
ApplicationStyle=TDI
FlatControls=0
NoBorderControls=1
ColorMap=Classic
HideMenuStructure=0
HideActiveFunctions=0
MenuStructureEmbedded=0
MenuStructureExpanded=1
ActiveFunctionsEmbedded=0
SQLMonitorEmbedded=0
HideMainToolbar=0
LargeMenuIcons=1
MenuHotTrack=1
LargeActiveFunctionIcons=0
LargeExplorerIcons=1
AutoEdit=1
AutoQuickSearch=0
AutoApplyChanges=1
EditColor=$FFFFC0
RequiredColor=$80FFFF
ReadOnlyColor=$E0E0E0
DecimalSeparator=,
DateSeparator=/
DateFormat=DMY
LoadFormBGImage=1
GridOddColor=$F0F0F0
ShowHintBalloon=1
DefaultLoadMode='lmFullBurst'
UseReferenceButtons=1
VCLStyle=Windows10
WindowMinWidth=500
WindowMinHeight=400
FontName=Segoe UI
FontSize=9
FontColor=clWindowText
ExcelDefaultExt=.xlsx
FontHeight=-12

[CustomColors]
ColorA=FFFFFF
ColorB=49B9BC
ColorC=2737C9
ColorD=DD2067
ColorE=60AAFD
ColorF=32ABA5
ColorG=4D5259
ColorH=B5A2B7
```

```
ColorI=FF00FF
ColorJ=C47D60
ColorK=23F8F2
ColorL=983D96
ColorM=B8ACF7
ColorN=9D2F55
ColorO=8CFDA3
ColorP=576391

[FOPEngine]
EnginePath={app}\..\..\FOPEngineNew
BatchFileName=fop.bat
TemplateFileName=FOPDocProducerTemplate.xsl
Debug=1

[CodeSite]
;LogPath=$(ISFPrimerLogPath)
;LogFileName=ISFPrimer.csl
;TCPHost=192.168.1.102
;TCPPort=3434
```

Un altro file modificabile e personalizzabile è la classe TISConfigurazione, nella unit Mconfigurazione che deve mantenere la struttura di base come quella fornita dal framework ma può essere estesa per gestire altri aspetti di configurazione modificabili dall'utente e memorizzabili nel database.

8.2. Accesso al database e gestione/protezione delle password

Per accedere al database in una applicazione ISF occorre configurare 2 file diversi: uno è utilizzato dalla tecnologia InstantObjects, l'altro dalla tecnologia DataDictionary di ISF per la verifica e l'aggiornamento del database.

8.2.1. I files di accesso ai dati

I due files che si occupano di gestione l'accesso ai dati in una applicazione InstantSolutions sono:

\Exe\NomeApp.xml: contiene la definizione dei connectionbrokers di InstantObjects

esempio di file di connessione:

```
<TInstantConnectionDefs>
  <TInstantFireDACConnectionDef>
      <Name>ISFPrimer_FB_UTF8</Name>
    <Database>{app}\..\Database\ISFPrimer-UTF8.fdb</Database>
    <Port>-1</Port>
    <Properties></Properties>
    <DriverID>FB</DriverID>
    <UseDelimitedIdents>FALSE</UseDelimitedIdents>
    <User_Name>SYSDBA</User_Name>
    <EncryptedPassword>yekretsam</EncryptedPassword>
      <BlobStreamFormat>sfXML</BlobStreamFormat>
      <UseUnicode>True</UseUnicode>
  </TInstantFireDACConnectionDef>
  <TInstantFireDACConnectionDef>
      <Name>ISFPrimer_FB</Name>
    <Database>{app}\..\Database\ISFPrimer.fdb</Database>
    <Port>-1</Port>
    <Properties></Properties>
    <DriverID>FB</DriverID>
    <UseDelimitedIdents>FALSE</UseDelimitedIdents>
    <User_Name>SYSDBA</User_Name>
    <EncryptedPassword>yekretsam</EncryptedPassword>
      <BlobStreamFormat>sfXML</BlobStreamFormat>
    <DefaultStatementCacheCapacity>100</DefaultStatementCacheCapacity>
    <ReadObjectListWithNoLock>TRUE</ReadObjectListWithNoLock>
  </TInstantFireDACConnectionDef>
  <TInstantFireDACConnectionDef>
      <Name>ISFPrimer_MSSQL</Name>
      <DriverID>MSSQL</DriverID>
      <Server>127.0.0.1, 1434</Server>
      <OsAuthent>True</OsAuthent>
    <Database>ISFPrimer</Database>
      <BlobStreamFormat>sfXML</BlobStreamFormat>
```

```
  </TInstantFireDACConnectionDef>
</TInstantConnectionDefs>
```

Dictionary\FDConnections.ini: contiene la definizione di accesso ai dati tramite FireDAC e i servizi di ISWorkbench. N.B. se non si vuole integrare nell'applicazione la verifica/aggiornamento della struttura del database questo file non è necessario.

Esempio di file di connessione:

```
[ISFPrimer_FB_UTF8]
DriverID=FB
Database={DictionaryPath}\..\Database\ISFPrimer-UTF8.fdb
User_Name=SYSDBA
Password=masterkey
UseAlterTable=Yes
SQLScriptSeparator=;
BlobStreamFormat=sfXML
UseUnicode=True

[ISFPrimer_FB]
DriverID=FB
Database={DictionaryPath}\..\Database\ISFPrimer.fdb
User_Name=SYSDBA
Password=masterkey
UseAlterTable=Yes
SQLScriptSeparator=;
BlobStreamFormat=sfXML

[ISFPrimer_MSSQL]
DriverID=MSSQL
Server=127.0.0.1, 1434
Database=ISFPrimer
OSAuthent=yes
UseAlterTable=Yes
SQLScriptSeparator=GO
```

Questi due files condividono i nomi di accesso (Alias) e se all'interno dell'applicazione si vuole sfruttare l'accesso nativo al database occorre che vengano configurati in parallelo e mantenuti allineati.

8.2.2. Protezione delle password di accesso al database

E' possibile proteggere "nascondere" le password di accesso al database salvate dentro i files di configurazione NomeApp.xml e FDConnections.ini

Per evitare di scrivere in chiaro la password nel file NomeApp.xml è possibile definire la voce EncryptedPassword (e non indicare la voce Password). In tal caso il framework cerca una funzione per decrittare tale password che deve essere registrata in questo modo (si consiglia nell'inizializzazione della unit MConfiguration:

```
  //Registro una procedura custom di decrittazione della password di accesso
  ISRegisterDecryptAuthProc(MyDecryptAuthProc);
```

MyDecryptAuthProc deve essere una procedura definita con questi parametri:

```
procedure MyDecryptAuthProc(var AUserName, APassword: string);
```

Il framework invocherà la procedura di decrittazione solo se trova definito EncryptedPassword e non trova Password nel file xml di configurazione.

All'interno della procedura è necessario decrittare la password, che quindi verrà utilizzata per la connessione al database. E' possibile anche decrittare o cambiare il nome utente che accede al database. L'algoritmo di decrittazione che si vuole adottare è a carico dell'utilizzatore del framework, che dovrà preoccuparsi anche di avere un tool che gli permetta di crittare la password da indicare in EncryptedPassword.

Sempre dalla versione 6.6.5 è possibile non indicare alcun utente e password nel file FDConnections.ini. In tal caso il framework è in grado ora di fornire lo stesso user e password (eventualmente decrittata) utilizzato per la connessione a InstantObjects.

8.2.3. Protezione delle password di accesso degli utenti

Dalla versione 6.6.5 è possibile proteggere la password degli utente che viene salvata nel DB.

Dato che di default (per retrocompatibilità) la password non viene protetta, è necessario registrare la procedura di protezione, utilizzando quella già messa a disposizione dal framework stesso (che applica l'hash alla password) oppure fornendone una propria.

Inserire nella sezione initialization di MConfiguration:

```
  //Registra una procedura custom di protezione della password utente
```

```
  //utilizzando quella già fornita che applica l'hash
  ISRegisterProtectPasswordProc(ProtectPasswordWithHash);
```

Sia nel momento in cui la password viene aggiornata dall'utente stesso nella finestra di "modifica password", che quando viene aggiornata da parte dell'amministratore nella finestra di gestione utenti, la password **viene protetta richiamando questa procedura**. In una situazione in cui le password non sono ancora protette, è possibile proteggere tutte le password degli utenti chiedendo loro di modificare la propria password oppure attraverso la maschera di gestione utenti da parte dell'amministratore. Il Framework è in grado di accedere sia tramite la password utente in chiaro che utilizzando la password protetta: in questo modo la protezione delle password utenti può essere fatta in modo progressivo, senza bloccare l'operatività.

8.2.4. Regole di validazione della password utente

E' possibile stabilire delle regole specifiche per l'inserimento della password da parte dell'utente. Occorre registrare la propria procedura che verifica la validità della password e che fornisce il messaggio con la regola che sta applicando, ad esempio:

```
//Registra una procedura custom di verifica della Password
ISRegisterPasswordCheckProc(MyCheckUserPassword);

procedure MyCheckUserPassword(const AUserName, APassword: string;
  out APasswordIsCorrect: Boolean; out ARulesMsg: string);
var
  LLength: Integer;
begin
  LLength := Length(APassword);
  APasswordIsCorrect := LLength >= 8;
  if LLength < 5 then
    ARulesMsg := 'Indicare una password di almeno 8 caratteri'
  else if LLength < 8 then
    ARulesMsg := 'Password debole'
  else
    ARulesMsg := 'Password forte';
end;
```

N.B. il messaggio ARulesMsg viene visualizzato sulla finestra di modifica password per indicare all'utente se sta inserendo una password corretta ed eventualmente anche il livello di sicurezza.

8.3. Performance e Isolation Level

8.3.1. Isolation "DirtyReads"

Con l'utilizzo del broker FireDAC è possibile anche controllare l'isolation-level della connessione attraverso il parametro Isolation contenuto nel file xml di connessione. Di default, se non è specificato, viene usato l'isolation level "unspecified" che a livello di FireDAC significa utilizzare il default per il tipo di connessione (quindi in caso di MS-SQL o FireDAC sarà ReadCommitted).

Se si imposta il parametro **isolation** a "xiDirtyReads" la connessione leggerà i dati aggiornati sul DB ma che sono sottoposti a transazione attiva.

8.3.2. Letture "dirty" WITH(NOLOCK)

Solo per le connessioni con MS-SQL è possibile un approcio misto, ovvero mantenere l'isolation level a Read-Committed (per evitare in fase di update di leggere dati non ancora aggiornati da un'altra transazione) ma utilizzare le letture "sporche" solo in caso di select di chiavi di oggetti. Per far questo occorre abilitare il parametro **ReadObjectListWithNoLock=True** in modo tale che InstantObjects aggiunga nelle letture delle chiavi la parola chiave WITH(NOLOCK) su tutte le tabelle.

8.3.3. Aumentare le performance con Statement Cache

Utilizzando il parametro DefaultStatementCacheCapacity=N (dove n può essere ad esempio 100), si chiede a InstantObjects di non "buttare" le query di accesso ai dati ma di riutilizzarle a parità di "statement SQL": questo permette di riutilizzare query già create (cioè oggetti TFDQuery) che sono già state "preparate", alle quali vengono solo cambiati i valori parametri: questa tecnica migliora le prestazioni, ma introduce anche un rallentamento nella "ricerca" dello statement identico pertanto il valore di N va definito con cautela per evitare che il tempo necessario a cercare la query già utilizzata non incida maggiormente sui vantaggi prestazionali.

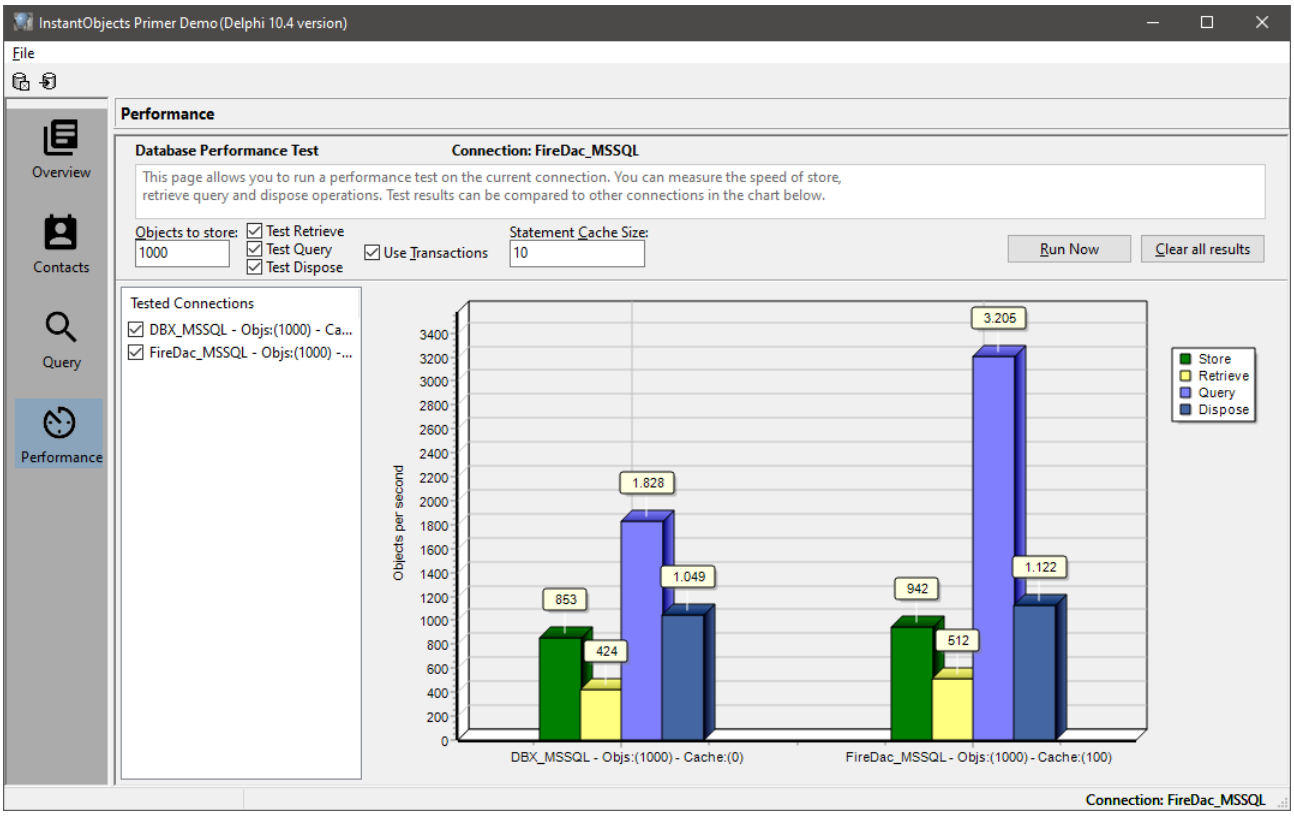
8.3.4. Misurare le prestazioni con InstantObject Primer demo

Nella cartella InstantObjects del framework esiste una famosa applicazione demo dal nome "Primer" che mostra l'uso di InstantObjects. (la potete trovare sotto {ISF_Install_Dir}\InstantObjects\Demos\PrimerCross e non è da confondere con ISFPrimer.

In questa applicazione esiste una form utile per misurare le prestazioni delle connessioni, sia per vedere lo stesso

broker come si comporta con connessioni locali o remote, sia per confrontare broker di tecnologie diverse (es. DbExpress e FireDAC), anche se DbExpress non è più utilizzato come Broker di ISF8, se ci sono prestazioni migliori in DbExpress rispetto a FireDAC connessi allo stesso database significa che il broker FireDAC non è efficiente.

Questa è una schermata di esempio delle prestazioni:



8.4. Aggiornamento struttura del database

E' possibile abilitare l'applicazione ad aggiornare automaticamente la struttura del database quando viene rilasciata una nuova versione del prodotto. Il check viene fatto tramite il campo CONFIGURATION.DBVERSION e la versione dell'esegubile.

Per fare il check automatico occorre scrivere queste righe nel datamodule DFunctions:

```
procedure TdtmDFunctions.BeforeConnect(ConnectionDef : TInstantConnectionDef);
begin
  CheckDatabase(ConnectionDef, dtmdMain.dictMain, dtmdMain.LoginEvent, '%d.%d.%d');
end;
```

L'ultimo parametro può essere '%d.%d oppure %d.%d.%d: questo serve per decidere se si vuole fare il check solo con Major Version e Minor Version o anche per Release: se le informazioni di versione dell'applicazione e quelle dentro il DB differiscono parte il processo di aggiornamento automatico del database.

9. Altre configurazioni/aspetti applicativi

9.1. Mostrare uno sfondo dell'applicazione dinamico

Di default il framework carica lo sfondo dell'applicazione e della Homepage configurato nel file .ini nella voce:

```
[Application]
Wallpaper=Wallpaper.png
HomePageWallPaper=Wallpaper.png
```

andando a inizializzare le variabili globale vcApSfondo e vcApHomeWallpaper.

E' possibile modificare tali sfondi una volta che ci si è collegati al database, sulla base del nome dell'ambiente scelto oppure di un qualsiasi valore contenuto in una classe del database a cui si è connessi.

Ad esempio, se si vuole cambiare lo sfondo in base al nome della connessione, occorre modificare Dfunction.pas in alcuni punti.

1) definire una variabile privata per salvare il nome di default del file dello sfondo:

```
  private
    vcSfondoLogin: string;
```

2) Aggiungere queste righe di codice nel metodo:

```
procedure TdtmDFunctions.AfterConnect(ConnectionDef : TinstantConnectionDef);
```

per cambiare le variabili globali dello sfondo dell'applicazione e della HomePage:

```
  //Change wallpaper based on database name
  vcSfondoLogin := vcApSfondo;
  vcApSfondo := ExtractFilePath(Application.ExeName)+
    ConnectionDef.Name+'.jpg';
  vcApHomeWallpaper := vcApSfondo;
```

3) Ripristinare lo sfondo quando si effettua il logout:

```
procedure TdtmDFunctions.AfterDisconnect(ConnectionDef : TInstantConnectionDef);
begin
  vcApSfondo := vcSfondoLogin;
end;
```

Con queste semplici modifiche è possibile personalizzare lo sfondo in qualsiasi modo, anche leggendo valori contenuti nel database.

9.2. Mostrare il contratto di licenza del software

E' possibile mostrare il contratto di licenza del software all'interno della form di informazioni. Occorre fornire l'informazione del file da mostrare e del titolo per la finestra:

```
procedure TdtmDFunctions.AfterConnect(ConnectionDef : TInstantConnectionDef);
var
  LicenseFileName: string;
  LicenseTitle: string;
  FileDate: Integer;
begin
{$IFNDEF ISFCONSOLE}
  //Registration of license file
  LicenseFileName := Format('%s\Licenza_ISFPrimer.rtf',
    [ExtractFilePath(ParamStr(0))+'..\Setup\']);
  if FileExists(LicenseFileName) then
  begin
    FileDate := FileAge(LicenseFileName);
    LicenseTitle := Format('Approved at %s',[DateToStr(FileDateToDateTime(FileDate))]);
    RegisterLicenseFile(LicenseTitle, LicenseFileName);
  end;
{$ENDIF}
end;
```

Il programma prevede che il file di licenza si chiami Licenza_ISFPrimer.rtf e che sia presente nella cartella ..\Setup rispetto alla posizione dell'eseguibile.

9.3. Ruolo di UmultiAppSpecific

Ciascun progetto ha una unit specifica per definirne le peculiarità, oltre al file .ini dell'applicazione.

Questa unit è importante per la configurazione di componenti di terze parti supportate da ISF ma che sono a pagamento, come ad esempio ReportBuilder, WPDF e WPView, EDocEngine, ecc... In queste unit sono presenti le uses per utilizzare questi componenti, sottoposte a direttive di compilazione specifiche.

9.4. *Utilizzo avanzato della CBLib8*

La libreria CBLib 8 fornita insieme a InstantSolutions contiene molti componenti di utilizzo avanzato. Vediamo alcuni esempi.

9.4.1. Utilizzare il componente CBGoogleMapViewer

Nella CBLib è disponibile il componente CBGoogleMapViewer che si basa sul TEdgeBrowser e consente di mostrare una mappa di Google, indicando una location o un percorso.

N.B. Per poter compilare è necessario scaricare da GetIt il componente: EdgeView2 SDK.

Il nuovo componente va aggiunto ad una pagina scheda di una form custom, allineato a destra, e va implementato il metodo:

```
procedure TfmContacts.RefreshGoogleMap;
var
  Address: string;
  Contact: TISContact;
begin
  if (CurrentObject is TISContact) and (pgctData.ActivePage = tsScheda) then
  begin
    Contact := TISContact(CurrentObject);
    Address := format('%s, %s', [Contact.Address, Contact.City, Contact.Province]);
    MapViewer.GotoAddress(Address);
  end;
end;
```

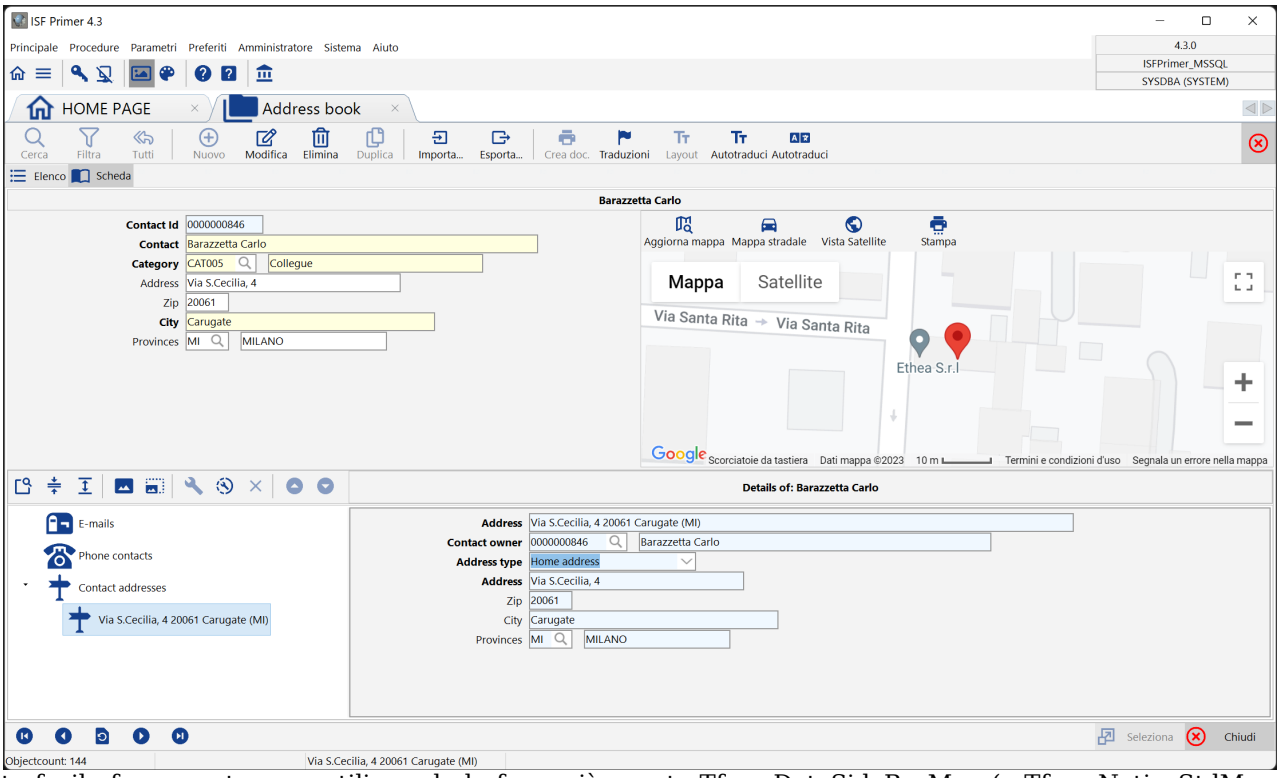
E' sufficiente calcolare l'indirizzo completo e chiamare il metodo “GotoAddress” del viewer per mostrare la mappa sincronizzata con il dato che sto visualizzando: la sincronizzazione è garantita dall'aver chiamato RefreshGoogleMap in questi 2 metodi:

```
TfmContacts.ListDataSetAfterScroll e TfmContacts.pgctDataChange(Sender: TObject);
```

L'effetto che si ottiene è quello di mostrare la mappa con la possibilità di cambiare zoom, tipo di visualizzazione, ecc... Questa tecnica può essere utilizzata in qualsiasi form custom, basta associare i dati di un indirizzo al componente che lo mostra.

N.B. occorre acquisire la licenza per le API di Google Maps: è un servizio a pagamento di Google. Una volta acquisito il proprio codice va registrato nella unit UmultiAppSpecific nella sezione initialization:

```
CBRegisterGoogleMapsApiKey('AIzaSyBNY0ARa4GdRU4LrOK……..');
```



E’ molto facile fare questa cosa utilizzando la form già pronta TformDataSideBarMap (o TformNativeStdMap per le form native): basta ereditare da questa form e implementare un metodo nell'oggetto che restituisce l'indirizzo:

```
function TISContact.GetMapAddress: string;
begin
  Result := format('%s, %s', [Address, City, Province]);
```

```
end;
```

9.4.2. Utilizzare il componente CBBrowser

Dalla versione 8.4.0 di InstantSolutions è disponibile il componente CBBrowser che si basa sul TEdgeBrowser e consente di mostrare un Browser “Edge” all’interno di una form.

N.B. Per poter compilare è necessario scaricare da GetIt il componente: EdgeView2 SDK.

Il nuovo componente è già disponibile nella form FnativeWEBBrowser e permette di visualizzare qualsiasi contenuto HTML.

Nella classe di base di CBInstantObject è ora presente un nuovo metodo di oggetto che restituisce il contenuto HTML dell’oggetto da mostrare dentro il browser: questa cosa è molto utile ad esempio per mostrare una fattura elettronica, come mostrato in questo esempio.

```
function TISFEHeader.GetBrowserContent(out ACaption, AHint, AImageName: string): string;
begin
  ACaption := SHOW_INVOICE;
  AHint := SHOW_INVOICE_WEB_HINT;
  AImageName := 'file-invoice-dollar-solid';
  if NomeFileXML <> '' then
  begin
    if FileExists(Self.NomeFileXML) then
    begin
      FInvoice := dmFEResources.LoadLegalInvoice(NomeFileXML);
      Result := dmFEResources.RenderInvoiceAsHTML(FInvoice, '');
    end
    else
    Result := Format('<html><p>%s</p></html>',
      [Format(FILE_NOT_FOUND,[NomeFileXML])]);
  end
  else
  Result := Format('<html><p>%s</p></html>', [FE_NOT_ASSIGNED]);
end;
```

10. Applicazioni “Console” con InstantSolutions

10.1. Principi sulle applicazioni console

Quando è necessario svolgere delle operazioni che non richiedono interfaccia utente (per la gestione, ad esempio, di processi schedulati) è possibile con Delphi sviluppare applicazioni “console” che possono essere schedulate a livello di sistema ed avviate. Con InstantSolutions è possibile utilizzare tutta la logica di business sviluppata a livello di classi in una applicazione “Console” seguendo alcune linee guida:

- non fare mai riferimenti all'oggetto Application della uses Forms
- non chiamare mai “dialog boxes” di messaggi, ma sollevare Eccezioni o scrivere in un file di log
- utilizzare i parametri dalla linea di comando per gestire le diverse operazioni

Nella cartella ISFTemplate\Projects è presente il template ISConsoleTemplate che mostra un piccolo esempio di come gestire una applicazione Console.

IS fornisce alcuni servizi di base nella unit UConsoleFrameworkLib, quali:

- ReadIniFile
- ConsoleLogin
- WriteToLog

N.B. in una applicazione CONSOLE, per non includere componenti visuali della VCL, occorre specificare alcune direttive di compilazioni speciali:

IO_CONSOLE (per compilare InstantObjects senza la VCL)

CB_CONSOLE (per compilare la CBLib senza la VCL)

ISF_CONSOLE (per compilare ISF senza la VCL)

11. Server “REST” con InstantSolutions e M.A.R.S

11.1. Principi sulle applicazioni REST

Un server REST permette di fornire “risorse” di dati in formato JSON (generalmente).

Con ISF e l’utilizzo della libreria MARS Curiosity (di Andrea Magni) è possibile costruire in modo semplice una applicazione REST basata su ISF.

L’applicazione dovrà contenere tutte le classi del datamodel necessarie ed è sufficiente implementare una classe di base di risorse per poter accedere ad un InstantConnector di tipo FireDAC per accedere agli oggetti.

Per ulteriori dettagli consultare i consulenti di Ethea.

12. Applicazione Web con InstantSolutions e KITTO

12.1. L'utilizzo con Kitto

Dalla versione 7.3.7 è possibile integrare le classi contenute in ISWorkBench per generare e manutenere i modelli di una applicazione Kitto.

Occorre innanzitutto configurare il file ISWorkBench.ini per aggiungere questa sezione:

```
[KITTO]
HOMEFOLDER={DictionaryPath}\..\Home
```

in cui specificare qual'è la Home path dell'applicazione Kitto, dentro la quale ci sono i "model".

A livello di menu di ISWorkBench è quindi disponibile una nuova funzione di generazione/aggiornamento Model di Kitto da avviare.

Per ulteriori dettagli su come si costruisce una applicazione WEB con Kitto si rimanda alle informazioni del prodotto presenti sul sito www.ethea.it

12.2. La generazione automatica dei model di Kitto

Dalla versione 7.3.8 è possibile utilizzare ISWorkBench anche per creare e aggiornare i model di una applicazione di Kitto prelevando tutte le informazioni contenute nelle classi di ISWorkBench.

Avviare la procedura dalla toolbar di ISWorkBench "Generazione automatica model Kitto".

N.B. se nel model di Kitto ci sono dei campi aggiuntivi non presenti in ISWorkBench essi non vengono tolti per evitare di eliminare i campi di tipo "expression". Questo però significa che se si tolgono dei campi "fisici" a delle classi occorre eliminarli a mano nei model di Kitto.

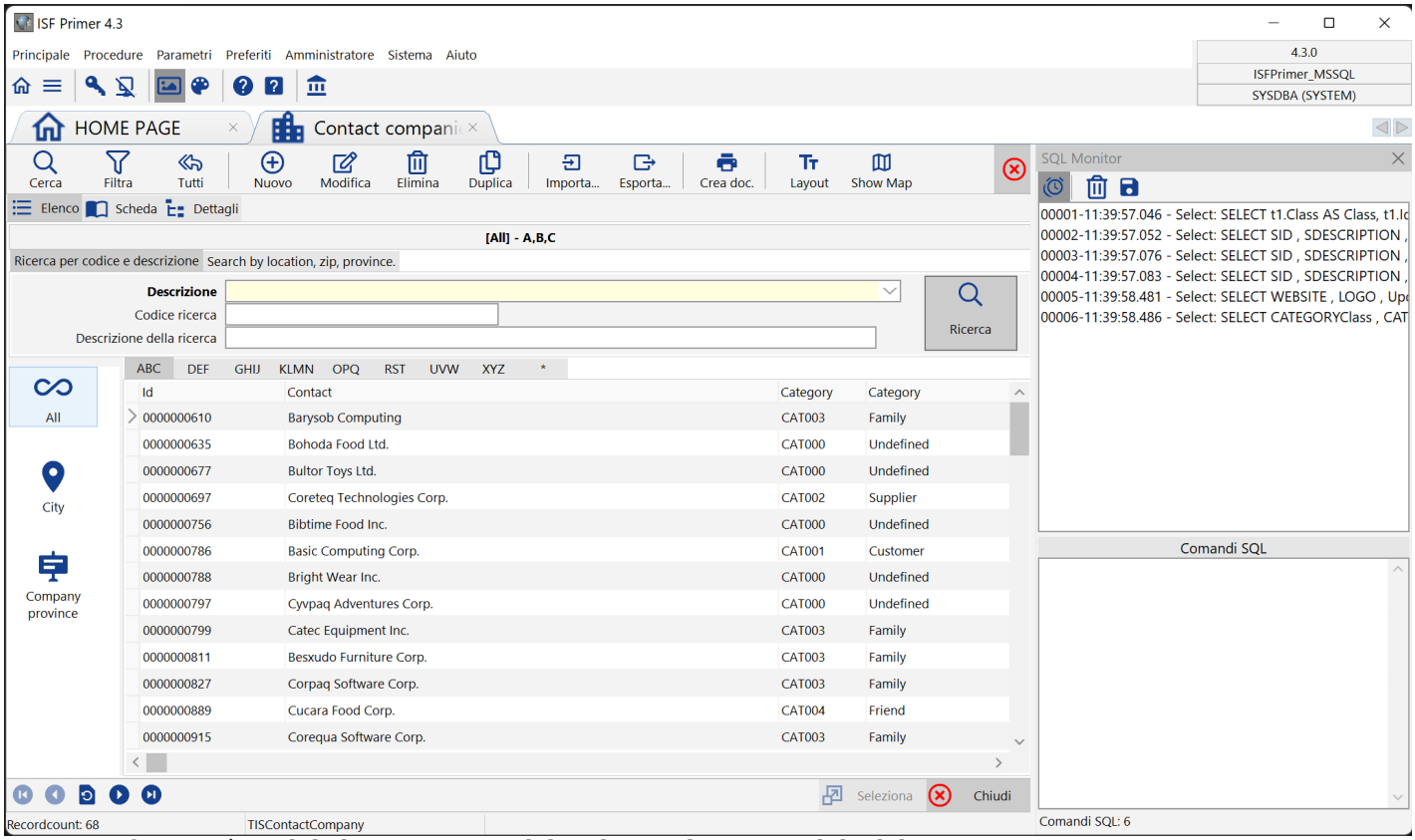
13. Tuning dell'applicazione e verifica integrità dei dati

Una applicazione sviluppata con ISF, che quindi si basa su InstantObjects, necessita di un tuning accurato, in quanto le “vere” istruzioni SQL che vengono create e lanciate all'interno dell'applicazione sono “invisibili” al programmatore che usa classi e broker di accesso ai dati. Sapere però che cosa succede “dietro le quinte” è fondamentale per assicurarsi buone prestazioni dell'applicazione.

Per questo motivo, all'interno del multiframework è disponibile una form di “SQL monitor”.

13.1. Attivazione e utilizzo dell'SQL monitor

Per avere a disposizione questa form occorre compilare l'applicazione con la direttiva: IO_STATEMENT_LOGGING ed effettuare il login con un utente che ha i privilegi di sistema (nei database di esempio tale utente è: SYSDBA - password: masterkey). Richiamare la form dal menu “sistema”.



In questa finestra è visibile l'SQL monitor del multitemplate, attivabile dal menu “sistema”.

All'interno della finestra di SQL Monitor è possibile vedere quanti e quali sono le istruzioni SQL native e tradotte dal broker che si sta utilizzando durante l'utilizzo dell'applicazione (In basso è anche visibile il contatore delle query eseguite dall'applicazione).

Come già spiegato in precedenza occorre ricordare che quando si esegue una query con il comando IQL, come ad esempio:

```
SELECT * FROM TISContactCompany ORDER BY Descrizione
```

il meccanismo che utilizza InstantObjects è quello di tradurre la query in formato compatibile con il broker/database SQL sottostante:

```
SELECT t1.Class AS Class, t1.Id AS Id FROM CONTACTS t1 WHERE (t1.Class = 'TISContactCompany') ORDER BY t1.DX
```

attraverso la quale scaricare una lista di chiavi.

Quindi ad ogni “fetch” di oggetto viene eseguita una “retrieve” dell'oggetto che viene a sua volta tradotta in una o più istruzioni SQL, a seconda che la classe abbia più livelli di ereditarietà.

In questo esempio vengono prima presi gli attributi della classe TISContactCompany:

```
SELECT WEBSITE , UpdateCount  FROM COMPANIES WHERE Class = :Class AND Id = :Id
```

Parametri:

```
Class: ftString = TISContactCompany
Id: ftString = 0000001542
```

quindi vengono presi gli attributi della classe TISContact dalla quale la TISContactCompany :

```
SELECT CATEGORYClass, CATEGORYId , ADDRESS , ZIP , CITY , PROVINCEClass, PROVINCEId , PhoneContacts ,
```

```
Addresses , DX , UPDTIMESTAMP , UpdateCount  FROM CONTACTS WHERE Class = :Class AND Id = :Id
```

Parametri:

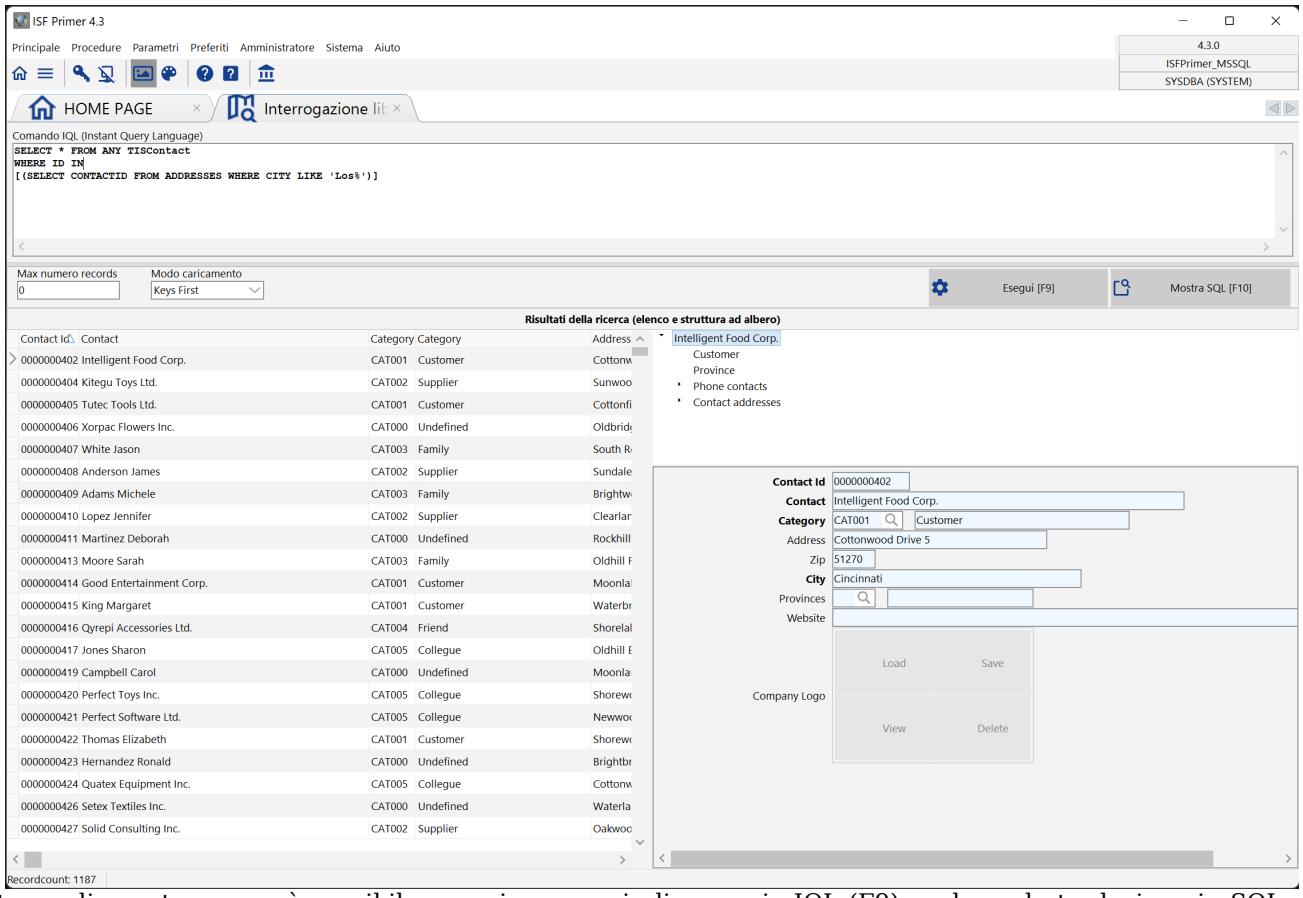
```
  Class: ftString = TISContactCompany
  Id: ftString = 0000001542
```

... e così via, per ogni oggetto la cui chiave è stata scaricata con la prima query.

Come si può notare InstantObjects esegue molte query per accedere agli oggetti, quindi si potrebbe pensare che le performaces generali dell'applicazione siano molto basse. In realtà, sfruttando il meccanismo di reference-counting degli oggetti, quando deve fare una retrieve di un oggetto che è già in memoria esso non viene più riletto dal database (a meno di non forzargli un refresh), quindi ci sono performances migliori.

13.2. Utilizzo della finestra di "interrogazioni libere"

Per imparare ad utilizzare il linguaggio IQL, provare a realizzare query complesse e capire come vengono tradotte in formato SQL, esiste la funzione di "interrogazione libera", avviabile dal menu sistema, disponibile se si effettua il login con un utente che ha i privilegi di sistema (nei database di esempio tale utente è: SYSDBA - password: masterkey).



All'interno di questa mappa è possibile eseguire query in linguaggio IQL (F9), vederne la traduzione in SQL nativo (F10) e osservare/navigare sui risultati nella parte sottostante.

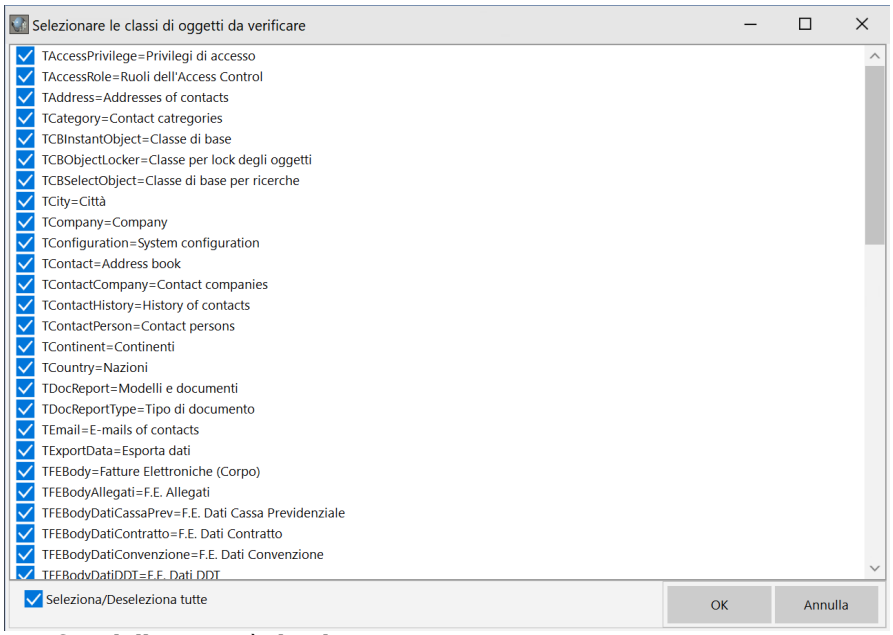
N.B. La query tradotta è mostrabile solo se si compila con la direttiva: IO_STATEMENT_LOGGING

13.3. Verifica dell'integrità dei dati

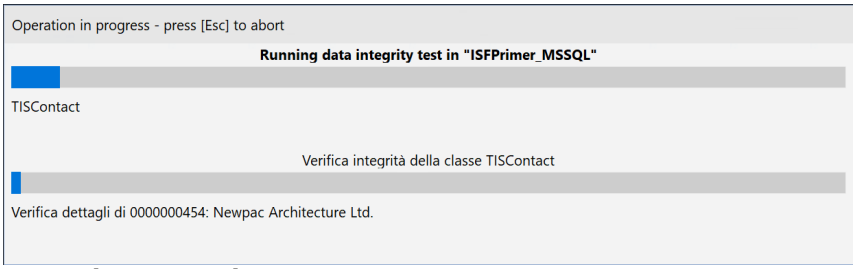
Lavorando per la prima volta con InstantSolutions (ma in particolare con InstantObjects) è possibile scrivere del codice che generi degli oggetti non "congruenti", in particolare dettagli "orfani" dell'oggetto padre.

Basta dimenticarsi di fare una operazione di "store" sull'oggetto padre, dopo aver ad esempio "agganciato" un oggetto figlio al proprio oggetto padre, che il database non è più "integro. E' buona cosa, durante lo sviluppo, controllare periodicamente che il database sia integro utilizzando la funzione "Verifica integrità" sempre dal menu "sistema" (sempre dopo aver fatto il login con privilegi system).

Apparirà la finestra di selezione delle classi sulle quali avviare la verifica di integrità dei dati:

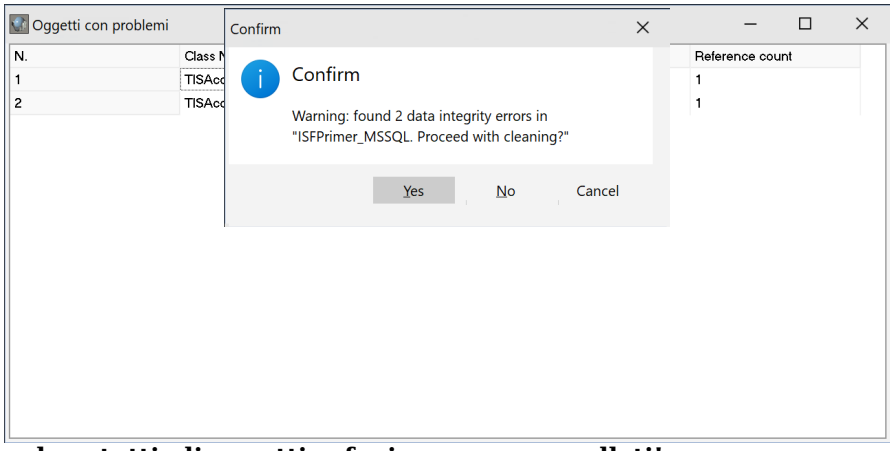


Premendo OK parte la verifica dell'integrità dei dati:



Se ci sono problemi viene visualizzata una lista:

Chiudendo questa finestra si può procedere in automatico alla rimozione di questi oggetti “orfani”:



Al termine della procedura tutti gli oggetti orfani saranno <u>cancellati</u>!

N.B. È possibile che un oggetto orfano di un oggetto padre sia di fatto semplicemente “staccato” dall'oggetto padre e quindi necessita solo di essere “riattaccato”. Fare attenzione a non cancellare oggetti orfani se non si è certi che sono dati incongruenti.

14. Struttura di un progetto ISF: riepilogo

14.1. Organizzazione e posizione dei files

- Un progetto ISF deve risiedere in una cartella sotto {ISFInstallDir}\MioProgetto.
- Un progetto ISF lavora attraverso i percorsi relativi: ecco un'esempio delle path definite

```
..\..\..\ext\CBLib\src;..\..\..\src\ISFLib\;
..\..\..\InstantObjects\Source\Core;
..\..\..\InstantObjects\Source\Brokers\FireDAC;
..\..\..\InstantObjects\Source\Catalogs\IbFb;*
..\..\..\InstantObjects\Source\Catalogs\MsSql;*
..\..\..\InstantObjects\Source\Catalogs\MySql;*
..\..\..\InstantObjects\Source\Catalogs\Oracle;*
..\..\..\InstantObjects\Source\Brokers\XML;
..\..\..\ext\HTMLViewer\source;
..\..\..\ext\SynEdit\source;
..\..\..\ext\ChromeTabs\Lib;
..\..\..\ext\ChromeTabs\GDIPlus;
..\..\..\ext\IconFontsImageList\Source;
..\..\..\ext\VCLStyleUtils\DDetours\Source;
..\..\..\ext\VCLStyleUtils\Common;
..\..\..\ext\OnGuard\source;
..\..\..\ext\OnGuard\Tools\LicenseManager;
```

* in base al database che si intende utilizzare

14.2. Le direttive di compilazione

La CBLib 8.X fornita con ISF 8.X ha la doppia “lingua base”: italiano e inglese. Di default i package sono in inglese.

Per avere la libreria (packages) e la propria applicazione con la CbLib in italiano occorre aggiungere la direttiva CBLIB_ITA ai progetti. Fate attenzione anche ai files delle forms contenute nella cartella Src di ISF: esistono 2 cartelle per i dfm del framework: ITA_dfm e ENG_dfm: l'installer copia i dfm di tali form nella cartella src.

Se si vuole sviluppare una applicazione con un'altra lingua di base, occorre copiare manualmente i files dfm corretti nella cartella src.

Verificare di avere queste direttive di compilazione:

```
NO_REPORTBUILDER;
NO_ISF_FLEXCEL;
OPENOFFICE;
FOPENGINE;
REGVERSION;
EFONGUARD;
VCLSTYLEUTILS;
CBLIB_ITA;
VCLSTYLEUTILS;
MAINXP;
```

Di default OpenOffice e FOP sono abilitati (e compilano), mentre ReportBuilder e Flexcel no: se si possiede ReportBuilder o Flexcel basta attivare la direttiva di compilazione.

15. Deploy di una applicazione

Normalmente il deploy di una applicazione ISF è simile a qualsiasi altro deploy di una applicazione Client/Server attraverso un processo di Build che va fatto esternamente all'IDE di Delphi e un file di setup.

15.1. Deploy dell'applicazione: compilazione via batch

In InstantSolutions sono stati predisposti alcuni file batch per la compilazione di un progetto all'esterno dell'IDE, molto utili in fase di deployment definitivo.

All'interno dell'IDE spesso si modificano direttive di compilazione (tipico è l'uso della direttiva DEBUG), oppure si compila utilizzando package run-time per velocizzare la compilazione... In fase di deployment la modalità di compilazione deve essere sempre univoca, pertanto è consigliabile utilizzare questi batch.

15.2. Come compilare una applicazione ISF da linea di comando

A questo scopo gli esempi forniti contengono anche una cartella BAT che contiene BuildProject.bat: questo batch è repsonsabile della compilazione del progetto, infatti contiene una informazione molto importante: le direttive di compilazione es:

```
set Defines=OPENOFFICE;FOPENGINE;NO_REGVERSION;NO_EFONGUARD;IBFB_SUPPORT;VCLSTYLEUTILS;MAINXP
```

All'interno della cartella Project\{DelphiVer} c'è il batch che serve per compilare il progetto per quella specifica versione di Delphi che si chiama BuildAllProjects.bat: ecco quello del progetto ISFPrimer per Delphi 11contenuto in Projects\D11:

```
@echo off
call ..\..\..\bat\DelphiEnvironment
call ..\..\bat\BuildProject ISFPrimer D11 MONOUTENTE
echo.
echo End compilation with Delphi 11
pause
```

15.3. Utilizzo di InnoSetup 6 per creare l'installer dell'applicazione

Nell'esempio ISFPrimer è presente anche una cartella di esempio di Setup realizzato con InnoSetup 6: il file è SetupISFPrimer.iss

Questo setup mostra come è possibile generare setup diversi in base ad una condizione, in questo esempio "MONOUTENTE" o "MULTIUTENTE".

Per compilare il setup direttamente dall'IDE di InnoSetup occorre impostare la riga:

```
#define TipoProgetto = "MONOUTENTE"
```

togliendo il punto e virgola davanti.

Questa variabile poi viene utilizzata in alcuni punti del progetto .iss per condizionare la creazione del package di installazione, es:

```
;Files del gruppo "EmbeddedFirebird" (tranne versione Multiutente)
#if TipoProgetto = "MONOUTENTE"
Source: .\FB2embedded\*; DestDir: {app}\ISF\ISFPrimer\Exe;
Source: .\FB2embedded\intl\*; DestDir: {app}\ISF\ISFPrimer\Exe\intl;
Source: .\FB2embedded\udf\*; DestDir: {app}\ISF\ISFPrimer\Exe\udf;
#endif
```

N.B. occorre prevedere anche l'installazione dei "client" di accesso ai dati, come ad esempio il client SQLNCLI11.dll per accedere a MS-SQL oppure FbClient.dll per accedere a Firebird SQL, ecc...

15.4. Avviare la generazione del Setup da linea di comando

Come per i batch di compilazione, allo stesso modo è possibile anche lanciare da linea di comando il setup realizzato con InnoSetup, attraverso il batch Build_Setup.bat sempre presente nella cartella ISFPrimer\Setup.

Questo batch prevede che gli venga passata la direttiva "MONOUTENTE" o "MULTIUTENTE" in modo da passarla a sua volta all'InnoSetup in questo modo:

```
"c:\program files (x86)\Inno Setup 6\iscc.exe" %1 /dTipoProgetto="%2"
```

N.B. Il batch assume che Inno Setup 6 sia installato sotto c:\program files (x86)\Inno Setup 6

Questo batch viene chiamato dal batch di compilazione ISFPrimer\Bat\BuildProject.bat in questo modo:

```
echo.
CHOICE /C SN /M "Vuoi generare anche il Setup (S/N)?"
IF ERRORLEVEL == 2 goto Exit
IF ERRORLEVEL == 1 goto BuildSetup
goto Exit
:BuildSetup
call ..\..\Setup\Build_Setup ..\..\Setup\Setup_ISFPrimer.iss %3
pause
```

```
goto exit
```

Siccome a sua volta questo batch viene chiamato da ISFPrimer\Projects\D2007\BuildAllProjects.bat come abbiamo visto sopra il risultato sarà quello di ricompilare l'applicazione da linea di comando lanciare il setup passando la tipologia di progetto (in questo esempio MONOUTENTE).

Questa tecnica può essere utilizzata per compilare e creare un setup condizionato per clienti diversi.

15.5. Deploy di una applicazione in cloud

Per poter fare il Deploy di una applicazione InstantSolutions in un ambiente cloud o terminal server, volendo separare l'applicazione dai file di configurazione e di memorizzazione dei dati degli utenti è possibile utilizzare una feature per gestire gli “Storage” locali in cartelle diverse sullo stesso server.

15.5.1. Step di Deploy da seguire

- Modificare il file MiaApplicazione.ini per aggiungere, nella sezione [FrameWork] la riga:

```
StoragePath=D:\MiaApplicazioneConfig
```

Tale cartella dovrà contenere delle sottocartelle, una per ogni società cliente, configurate per essere accessibili solo agli utenti di quella società.

Per ipotesi avremo:

```
D:\DossierManagerConfig\SocietaCliente
```

In questa cartella spostare il file FDConnections.ini e il file MiaApplicazione.xml chiamandolo SocietaCliente.xml e configurare al suo interno i parametri di connessione al database di quel cliente.

Avviare il programma MiaApplicazione con un parametro a linea di comando (in questo esempio sarà SocietaCliente).

15.6. Licensing con EfOnGuard:

la licenza può essere floating, cioè può esistere il file Registration.ini nella cartella SocietaCliente con indicato il numero massimo di utenti che possono accedere contemporaneamente.

Oppure può essere named, cioè esisteranno tanti file .ini (es. SA.ini, USER.ini) con la licenza per singolo utente di quella società.

Configurazione Path dell'applicazione:

Dal menu Amministratore / Configurazione impostare le seguenti cartelle:

```
Cartella condivisa documenti: {storagepath}\Documents\
Cartella condivisa immagini: {storagepath}\Images\
Cartella condivisa modelli: {app}\..\DocTemplates\
Cartella guida in linea: {app}\..\Help\
Cartella importazione doc.: {storagepath}\Import\
Cartella esportazione doc.: {storagepath}\Export\
```

In questo modo sia le immagini, che i documenti prodotti, che quelli importati o esportati, verranno salvati sotto la cartella D:\MiaApplicazioneConfig\SocietaCliente (che corrisponde all'alias {storagepath}).

16. Supporto Multilingua (Database e GUI)

Dalla versione 6 è previsto il supporto multilingua di ISF (già presente nella versione 5), per gestire in modo corretto tutti gli aspetti, in particolare la logica della lingua di base dell'applicativo, introdotta dalla versione 5.

Tutte le funzionalità per lo sviluppo di applicazioni multilingua con InstantSolutions sono integrate con il sistema di traduzione delle applicazioni "Ethea Translation Tools".

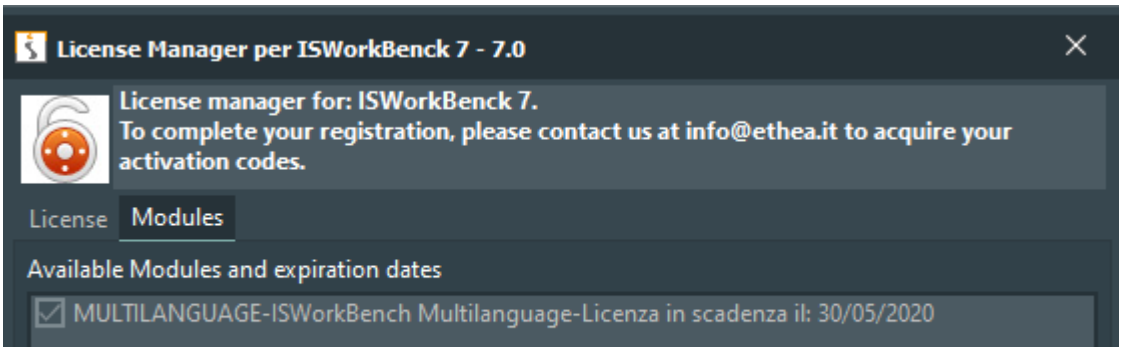
Con il supporto multilingua è possibile, all'interno di ISWorkbench, identificare alcuni attributi delle classi come "multilingua": a questi attributi è possibile fornire anche una descrizione in lingua alternativa a quella di default (solo se tra le lingue utilizzate ci sono l'italiano e l'inglese). Questa opzione è necessaria solo se si intende avere una GUI multilingua. Il concetto di multilingua si riferisce a 2 aspetti complementare ma diversi:

16.1. Gestione in lingua dei dati

Il sistema multilingua permette di avere una applicazione con una lingua di base, più tutti i campi in lingua.

A livello applicativo non è necessario modificare alcun programma, layout di mappa o report: la lingua che pilota il meccanismo di lettura/scrittura degli attributi è inizializzata in base alla lingua dell'utente, ma è possibile cambiarla al volo, in base al contesto (es. prima di stampare un report in lingua).

1) Bisogna richiedere la registrazione del modulo Multilingua di ISWorkbench per gestire applicazioni multilingua

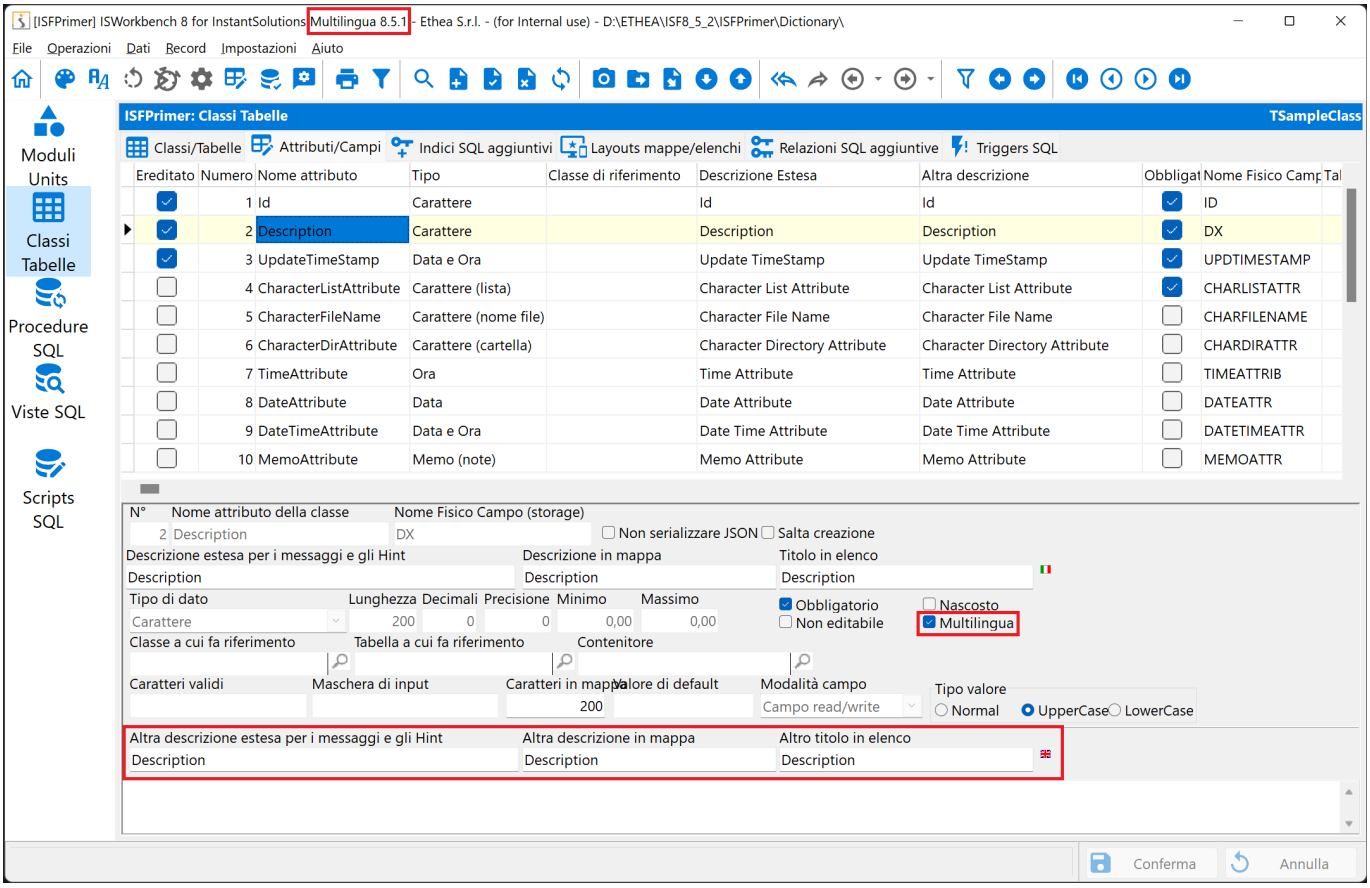


2) Per configurare una applicazione Multilingua occorre modificare il file ISWorkbench.ini impostando i valori della sezione MULTILANGUAGE:

```
[MULTILANGUAGE]
;If you have Ethea Translation Tools for ISF applications, set your application languages:
;For an English based application
LANGUAGES=ENG;ITA;ESP
```

In questo esempio l'applicazione ha come lingua di base l'inglese e deve gestire anche dati in lingua italiana e spagnola. Allo stesso modo deve essere modificato il file .ini dell'applicazione sempre nella sezione MULTILANGUAGE. La lingua di base (variable IODbLanguage) è inizializzata di default dalla direttiva di compilazione CBLIB_ITA, che determina la lingua di default con la quale si sta sviluppando l'applicazione, ma nulla vieta di avere una lingua di sviluppo della GUI italiana (CBLIB_ITA abilitata) e una lingua di default del database in inglese: il parametro da indicare nel file .INI dell'applicazione è **DB_LANGUAGE.**

```
[MULTILANGUAGE]
LANGUAGES=ENG;ITA;ESP
DB_LANGUAGE=ENG
```



3) Infine, per rendere un attributo multilingua occorre spuntare la voce “Multi-language” dell'attributo.

ISWorkbench si occupa automaticamente di:

a) generare le classi di InstantObjects con un meccanismo automatico di lettura/scrittura degli attributi dal database in base alla lingua attiva: se non trova il dato, utilizza il campo di base (nella lingua di default del database) e lo indica tra parentesi quadre.

b) Aggiornare la struttura del database aggiungendo i campi multilingua (preceduti dal prefisso della lingua es. ITA_MULTILANGDESC, ENG_MULTILANGDESC, ESP_MULTILANGDESC).

16.2. Gestione in lingua della GUI (interfaccia utente)

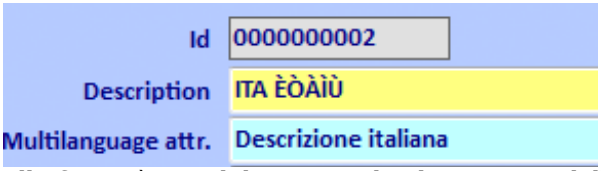
Complementare, ma non necessario, è la possibilità di avere una iterfaccia utente in lingua.

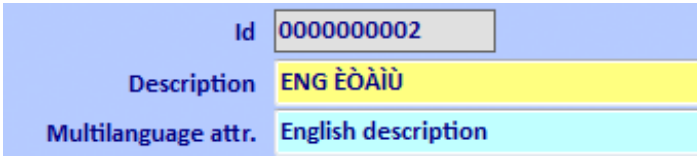
Se in ISWorkbench si definiscono anche le descrizioni estesa/mappa/elenco nella lingua alternativa rispetto a quella base dell'applicativo (nella figura si vedono questi valori in lingua italiana). In fase di generazione delle units delle classi, ISWorkbench si occupa automaticamente di generare tali descrizioni in modo compatibile con "Ethea Translation Tools". Seguendo poi le linee guida definite nel documento **EtheaTranslationTools.pdf** e utilizzando Il TranslationEditor e il RepositoryEditor sarà possibile creare una interfaccia utente multilingua dell'applicazione.

N.B. Se si sviluppa con la stessa versione di ISF su 2 progetti diversi, uno multilingua e l'altro no, quando si passa da un progetto all'altro è sempre necessario rigenerare con ISWorkbench le classi, perché le units di Framework sono diverse e ovviamente incompatibili.

16.3. Impatto sull'applicazione: visibilità dei dati

Come già detto a livello applicativo non è necessaria alcuna modifica. Dato che ISWorkbench genera un meccanismo intelligente di lettura e scrittura dell'attributo multilingua, sfruttando le potenzialità della programmazione a oggetti, l'effetto finale è quello di avere una applicazione che in modo dinamico e trasparente mostra i dati in lingua.





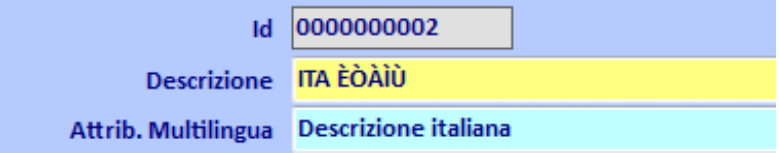
Nella figura è possibile notare che il contenuto dello stesso oggetto è diverso a seconda della lingua attiva.

In questo caso l'effetto è quello solamente del dato in lingua diversa.

N.B. Quando il valore in lingua non è presente nel database, il sistema propone comunque la descrizione in “lingua

base", ma tra parentesi quadrate, stando ad indicare che il valore deve ancora essere tradotto.

Se invece si cambia la lingua della GUI dell'applicazione l'effetto è quello di avere anche in automatico le descrizioni degli attributi:

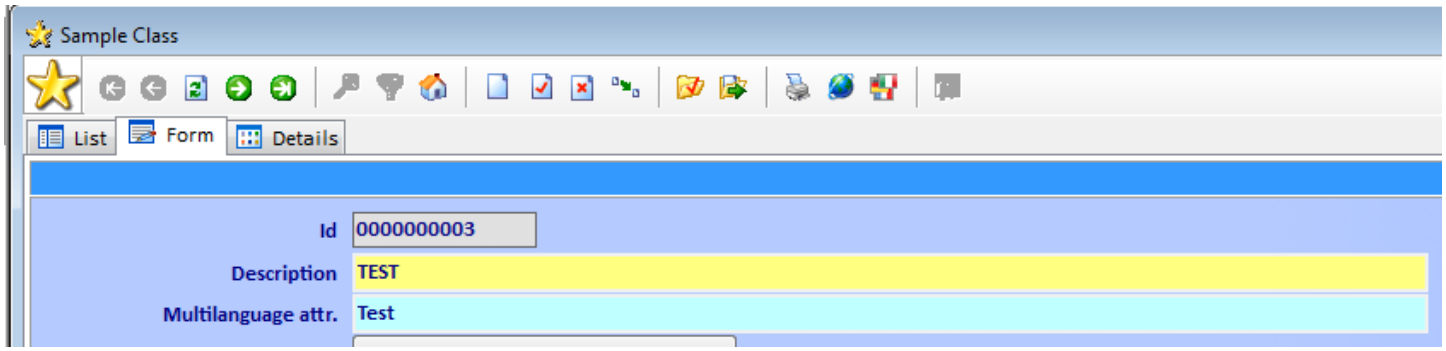


Ovviamente anche il resto della GUI dell'applicazione deve essere tradotta, ma dato che InstantSolutions sfrutta moltissimo il riutilizzo di codice e un meccanismo intelligente di GUI a più livelli di ereditarietà, il porting in lingua della GUI di una applicazione risulta molto veloce e semplice da realizzare, attraverso i tools forniti da Ethea stessa.

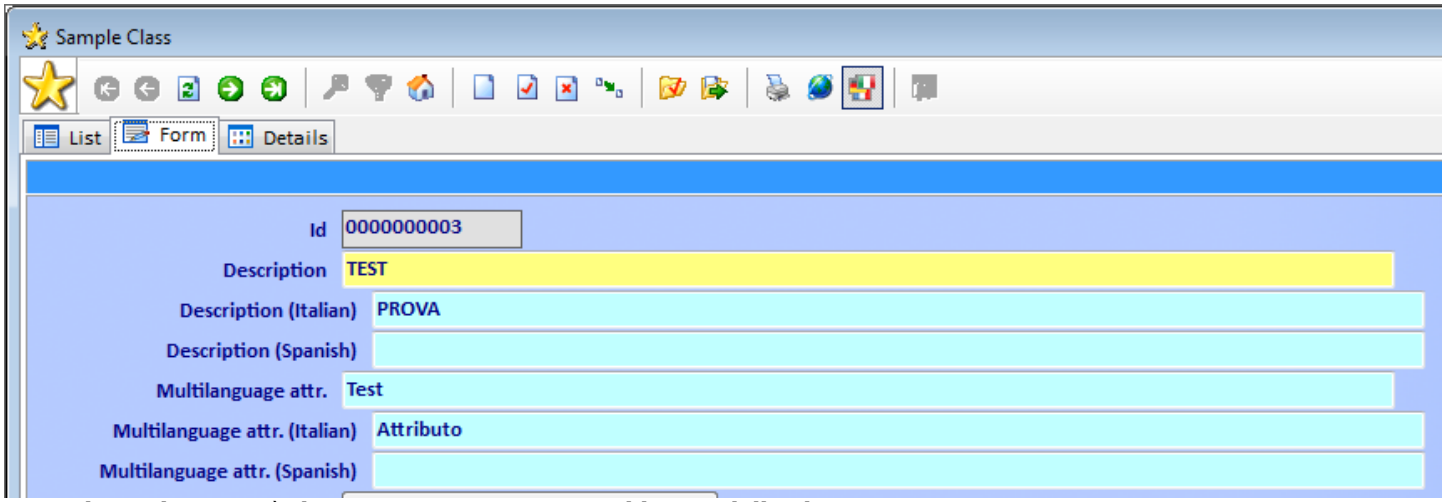
16.4. Nuova azione per mostrare/nascondere gli attributi della traduzione

Dalla ver.5.10 è disponibile su tutte le form del framework una nuova azione che si abilita automaticamente se la classe su cui si sta lavorando ha almeno un attributo multilingua.

Di default gli altri attributi nelle lingue da tradurre non sono visibili (come si vede da questa immagine):



Se si clicca sull'azione "Show/Hide Translation attributes" si ottiene la visualizzazione di tutti le traduzioni:



N.B. Gli attributi in più devo esistere comunque nel layout della classe:

```
Id|-1|+0|20|-1|"EditCaption"|LeftMiddle|20
Description|-1|+0|20|-1|"EditCaption"|LeftMiddle|20
ITA_Description|-1|+0|22|-1|"EditCaption"|LeftMiddle|20
ENG_Description|-1|+0|22|-1|"EditCaption"|LeftMiddle|20
ESP_Description|-1|+0|22|-1|"EditCaption"|LeftMiddle|20
```

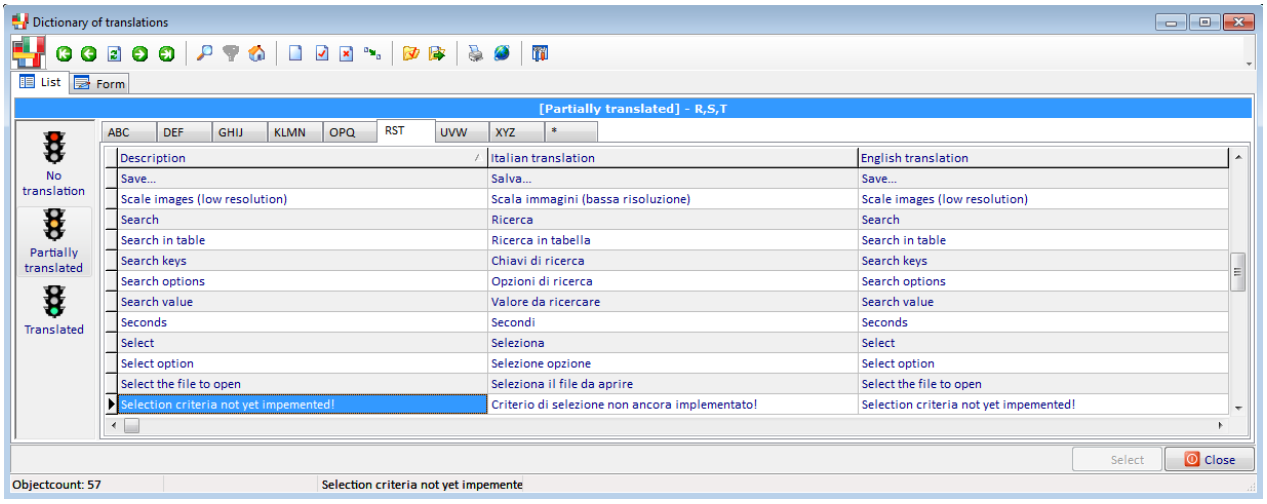
Il pulsante fuziona come un checkbox: quando mostra le traduzioni rimane premuto, se si ripreme ritorna in posizione normale.

Per cambiare il default di visualizzazione (se per esempio per una classe si vuole che gli attributi in lingua siano subito visibili) utilizzare il metodo di classe DefaultShowLanguageAttributes.

```
class function TISSampleClass.DefaultShowLanguageAttributes: boolean;
begin
  Result := True;
end;
```

16.5. Nuova classe di Dizionario per le traduzioni

La classe TISTranslation serve per collezionare un dizionario di stringhe tradotte in lingue diverse, per agevolare l'inserimento di valori multilingua. Per aggiungere la classe al proprio progetto seguire le istruzioni nel documento: [ISF]\Doc\Migrazione_da_ISF_5.3_a_ISF_5.10.pdf



Questa classe si registra automaticamente al sistema multilingua per fornire un meccanismo di traduzione.

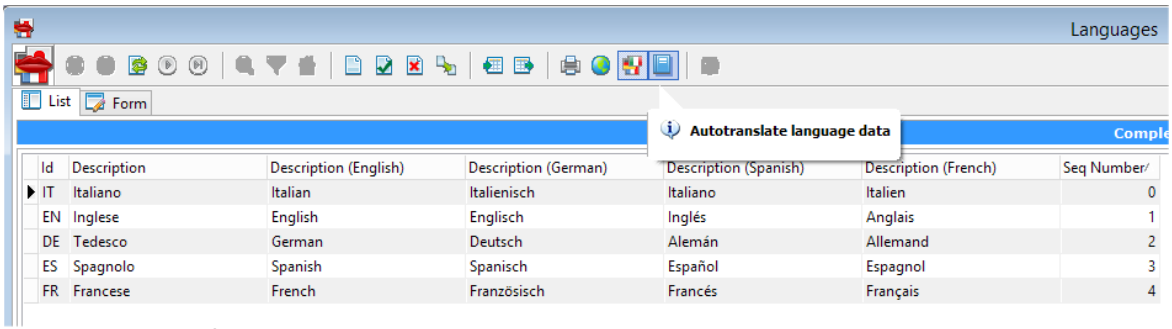
In questo modo è possibile gestire le traduzioni in automatico chiamando il servizio di traduzione TranslateValue:

```
procedure TISSampleClass.SetENG_Description(const Value: string);
var
  TranslatedValue: string;
begin
  inherited;
  if (Value <> '') then
  begin
    if TranslateValue(Value,mlEnglish,mlItalian,TranslatedValue) then
      _ITA_Description.Value := TranslatedValue;
    if TranslateValue(Value,mlEnglish,mlSpanish,TranslatedValue) then
      _ESP_Description.Value := TranslatedValue;
  end;
end;
```

Nell'esempio è mostrato come tentare la traduzione automatica in italiano e/o in spagnolo di una stringa in inglese che è stata impostata nell'attributo Description.

16.6. Nuova azione per tradurre automaticamente i dati in lingua

Su tutte le form del framework è disponibile una azione per la traduzione automatica tramite Microsoft Translation Services, che si abilita automaticamente se la classe su cui si sta lavorando ha almeno un attributo multilingua. Per poter abilitare tale azione occorre anche configurare il file MSTranslation.ini seguendo le istruzioni riportate nel documento: **{ISF}\Doc\Configurazione_Microsoft_Translation_Services.pdf.** All'interno dell'applicazione multilingua, comparirà l'icona per la traduzione dei dati:



Cliccando sul pulsante verrà richiesto se si vuole tradurre solo il record corrente o tutti i record della tabella. La traduzione avrà effetto solo sui dati non ancora tradotti. Nella mappa è visibile già il risultato della traduzione automatica.

16.7. Abilitazione dinamica delle lingue dell'applicazione

Mentre il file ISWorkbench.ini stabilisce le lingue supportate a livello di database e classi, il file .ini dell'applicazione può contenere un subset di lingue attive minore, rispetto a quelle disponibili, attraverso la variabile globale

AppLanguages. Impostando tale variabile (ad es. sottoponendola all'acquisto del modulo in lingua) è possibile abilitare le lingue attive in modo dinamico.

17. Supporto CodeSite

Dalla versione 6.3.4 di InstantSolutions è stato introdotto il supporto all'utilizzo di CodeSite per la gestione avanzata del logging di una applicazione. E' possibile utilizzare sia CodeSiteExpress che CodeSiteStudio.

Per abilitare una applicazione è necessario aggiungere la direttiva di compilazione CODESITE: con questa operazione si abilita di default il Live Viewer di CodeSite. InstantSolutions è stato abilitato per tracciare gli eventi:

```
AbstractData.SetActiveDataSet (cambio del dataset attivo)
TformChild.SetActiveClass (cambio della classe attiva)
TCBInstantObject.BeforeDispose (prima della cancellazione di un oggetto)
TCBInstantObject.BeforeStore (prima della memorizzazione di un oggetto)
TCBInstantExposer.DoAfterOpen (dopo l'apertura di un exposer in modalità content)
TCBInstantSelector.DoBeforeOpen (prima dell'apertura di un selector)
TCBInstantSelector.StoreAllObjects (
TCBAction.Execute (EnterMethod e ExitMethod: esecuzione di una azione)
TCBXPageControl.Change (cambio pagina di un pagecontrol)
```

Questi sono gli eventi più importanti che determinano il funzionamento di una applicazione.

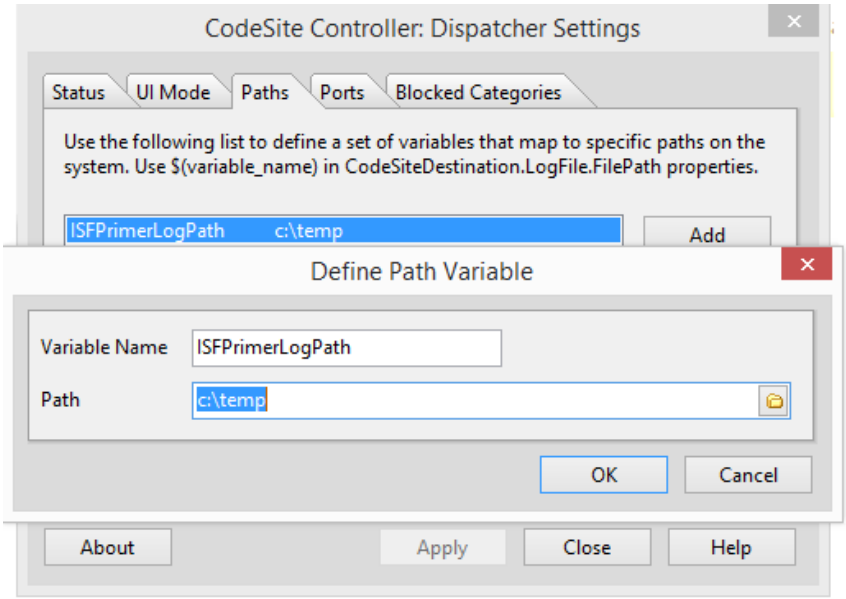
Se si vuole dirottare l'output di CodeSite su un file, è necessario configurare il file ini dell'applicazione, come in questo esempio:

```
[CodeSite]
LogPath=$(ISFPrimerLogPath)
LogFileName=ISFPrimer.csl
```

Se si vuole dirottare l'output di CodeSite verso un host remoto via TCP-IP aggiungere anche le righe:

```
TCPHost=192.168.1.102
TCPPort=3434
```

N.B. se si usa la sintassi $(ISFPrimerLogPath) significa che si si vuole demandare al Dispatcher la definizione della posizione del file di log, utilizzando il CodeSite Controller:



La nuova unit Ethea.CodeSite prevede anche la possibilità di filtrare i dati che si vogliono mandare a CodeSite, attraverso 3 eventi. Es.

```
    CodeSite_RegisterIncludeFilter(IncludeFieldToLog);
    CodeSite_RegisterIncludeObject(IncludeObjectToLog);

procedure TdmFunction.IncludeFieldToLog(Sender: TObject; const MasterField: string;
  Field: TField; var Include: Boolean);
begin
  Include := SameText(Field.FieldName,FLD_ID) or SameText(Field.FieldName,FLD_DX) or
    (Field.DataType = ftTimeStamp);
end;

procedure TdmFunction.IncludeObjectToLog(Obj: TObject; var Include: Boolean);
begin
  Include := Obj.ClassName = 'TISContantPerson';
end;
```

Questi 2 esempi mostrano come filtrari i campi del dataset oppure indica che l'unico oggetto che deve essere salvato nel log deve essere della classe TISContactPerson.

18. Integrazione di madExcept

Il sistema avanzato di gestione delle eccezioni (madExcept) fornisce un supporto alla soluzione degli errori inattesi che si verificano all'interno di una applicazione, con la possibilità di contattare il supporto tecnico in modo semplice ed efficace.

Ethea lo ha selezionato come strumento efficace per risolvere questo tipo di problemi e lo ha quindi integrato dentro InstantSolutions.

18.1. Acquisto e installazione di madExcept

MadExcept non è fornito insieme al framework di sviluppo InstantSolutions. Va quindi acquistato separatamente e installato. MadExcept si installa dentro l'IDE di Delphi nel menu “projects”.

Sito web: http://madshi.net/madExceptDescription.htm

18.2. Configurazione di madExcept in una applicazione Delphi

Per l'integrazione di madExcept all'interno di una applicazione InstantSolutions riferirsi al documento: Integrazione_madExcept_in_ISF.pdf.

18.3. Vantaggi sull'utilizzo di madExcept

L'utilizzo di madExcept integrato dentro InstantSolutions offre diversi vantaggi:

1) una gestione avanzata degli errori

2) la possibilità di inviare in automatico report di dettaglio sull'errore e lo screenshot senza che l'utente debba fare alcun tipo di operazione manuale

3) la possibilità di personalizzare la gestione degli errori: invece di registrare dtmdMain.ShowMadExcept è possibile registrare una propria procedura personalizzata.

4) la possibilità di avere madExcept installato su una sola macchina di sviluppo in team

19. Funzioni avanzate: l'uso di Trigger

Con ISF è possibile integrare l'uso dei Trigger in una applicazione. Va detto che i Trigger “sfuggono” al modello e alla logica di business implementata attraverso InstantObject con codice Delphi, ma quando sono utili è possibile sfruttare questa potenzialità dei motori SQL.

19.1. Trigger di “storicizzazione”

Immaginiamo di volere tenere una storia dei record di un oggetto, attraverso un meccanismo di trigger. Ogni volta che lato server SQL si aggiorna un record i dati precedenti vengono salvati in una tabella di “storico” in modo da poter risalire ai dati precedenti.

- Per far questo creiamo una tabella SQL che conterrà i dati storicizzati con la stessa struttura di CONTACTS, e la chiamiamo STCONTACTS.
- Posizioniamoci sulla tabella CONTACTS e andiamo sulla pagina Triggers.
- Creiamo il nuovo trigger indicando il nome ST_CONTACTS e definiamo il trigger Body: ogni database ha una sintassi diversa. Nel programma Demo abbiamo provato lo stesso trigger implementato in 3 modi diversi, in base al tipo di database utilizzato:

Corpo trigger Firebird

```
CREATE TRIGGER ST_CONTACTS FOR CONTACTS
ACTIVE AFTER UPDATE POSITION 0
AS
begin
  if (OLD.dx <> NEW.DX) then
  begin
    INSERT INTO STCONTACTS(
      CLASS,
      ID,
      UPDATECOUNT,
      DX,
      CATEGORIACLASS,
      CATEGORIAID,
      INDIRIZZO,
      CAP,
      COMUNE,
      PROVINCIACLASS,
      PROVINCIAID,
      ADDRESSES,
      CONTATTITELEFONICI
    )
    VALUES
    (
      OLD.CLASS,
    OLD.ID || '_' || CAST(OLD.UPDATECOUNT AS
VARCHAR(10)),
      OLD.UPDATECOUNT,
      OLD.DX,
      OLD.CATEGORIACLASS,
      OLD.CATEGORIAID,
      OLD.INDIRIZZO,
      OLD.CAP,
      OLD.COMUNE,
      OLD.PROVINCIACLASS,
      OLD.PROVINCIAID,
      OLD.ADDRESSES,
      OLD.CONTATTITELEFONICI
      );
  end
end
```

Corpo Trigger MS-SQL

```
CREATE TRIGGER ST_CONTACTS ON CONTACTS
FOR UPDATE
AS
BEGIN
  IF UPDATE(DX)
  BEGIN
    INSERT INTO STCONTACTS(
      CLASS,
      ID,
      UPDATECOUNT,
      DX,
      CATEGORIACLASS,
      CATEGORIAID,
      INDIRIZZO,
      CAP,
      COMUNE,
      PROVINCIACLASS,
      PROVINCIAID
      )
      SELECT
        DELETED.CLASS,

DELETED.ID+'_'+CAST(DELETED.UPDATECOUNT AS
VARCHAR(10)),
        DELETED.UPDATECOUNT,
        DELETED.DX,
        DELETED.CATEGORIACLASS,
        DELETED.CATEGORIAID,
        DELETED.INDIRIZZO,
        DELETED.CAP,
        DELETED.COMUNE,
        DELETED.PROVINCIACLASS,
        DELETED.PROVINCIAID
      FROM CONTACTS
      INNER JOIN DELETED ON
        (CONTACTS.CLASS = DELETED.CLASS) AND
        (CONTACTS.ID = DELETED.ID)
  END
END
```

Corpo Trigger Oracle

```
CREATE TRIGGER ST_CONTACTS
AFTER UPDATE ON CONTACTS
REFERENCING OLD AS OLD NEW AS NEW
FOR EACH ROW
  WHEN (OLD.DX <> NEW.DX)
BEGIN
    INSERT INTO STCONTACTS(
      CLASS,
      ID,
      UPDATECOUNT,
      DX,
      CATEGORIACLASS,
      CATEGORIAID,
      INDIRIZZO,
      CAP,
      COMUNE,
      PROVINCIACLASS,
      PROVINCIAID,
      ADDRESSES,
      CONTATTITELEFONICI
    )
    VALUES
    (
      :OLD.CLASS,
      :OLD.ID || '_' || CAST(:OLD.UPDATECOUNT
AS VARCHAR(10)),
      :OLD.UPDATECOUNT,
      :OLD.DX,
      :OLD.CATEGORIACLASS,
      :OLD.CATEGORIAID,
      :OLD.INDIRIZZO,
      :OLD.CAP,
      :OLD.COMUNE,
      :OLD.PROVINCIACLASS,
      :OLD.PROVINCIAID,
      :OLD.ADDRESSES,
      :OLD.CONTATTITELEFONICI
);
END;
```

Questo trigger storicizza il record del contatto solo se cambia la sua descrizione.

Da notare che per evitare conflitti di primary-key sulla tabella di storicizzazione abbiamo calcolato l'ID con l'ID del record da storicizzare + il campo UpdateCount.

20. Supporto allo sviluppo di Web-Services

Dalla versione 7.2.0 di ISF è possibile facilmente sviluppare Web-Services con il supporto alla libreria **MARS Curiosity**, un server REST open-source scritto da ***Andrea Magni*** (https://github.com/andrea-magni/MARS) e il supporto alla serializzazione/deserializzazione di oggetti InstantObjects basato sulla libreria **delphi-neon** scritta da ***Paolo Rossi*** (https://github.com/paolo-rossi/delphi-neon).

- A livello di InstantObjects, oltre al supporto "nativo" ObjectToJSON (solo per la serializzazione) è possibile ora utilizzare **delphi-neon** sia per la serializzazione che per la de-serializzazione di ogetti InstantObjects, attraverso un serializzatore/deserializzatore custom per InstantObjects:

ISF8\InstantObjects\Source\Core**Instant.Neon.Serializers.pas**

- A livello di ISFLib è stato creato un serializzatore/deserializzatore custom per CBInstantObjects, gli oggetti utilizzati da ISF:

ISF8\src\ISFLib**CBInstant.Neon.Serializers.pas**

N.B. dato che tale supporto è specifico e richiede una serie di passaggi per la configurazione occorre contattare il personale tecnico di Ethea per includere tale supporto in una propria applicazione.

21. Conclusioni

Iniziare a lavorare con ISF può sembrare molto semplice, ma non bisogna sottovalutare alcune complessità, che man mano che si incrementa il livello di complessità di una applicazione posso emergere.

Innanzitutto occorre conoscere le basi di InstantObjects che è un framework OPF molto potente ma che ha ancora alcune limitazioni, in particolare di performances sui dati in formato “elenco”.

Poi bisogna cercare di imparare a scrivere del codice “pulito”, dando a ciascusa classe l'onere dei suoi compiti e soprattutto bisogna imparare a separare ciò che è “logica di business” da ciò che è interfaccia utente.

ISF è lo strumento ideale per non trovarsi di fronte subito alle difficoltà di un approcio di sviluppo basato su un OPF, anche se per poterlo utilizzare al meglio occorre approfondire molti altri aspetti che in questo breve corso introduttivo non si è potuto toccare.

Lo staff di Ethea è sempre pronto e a disposizione ad intervenire sul framework stesso quando le esigenze degli sviluppatori convergono verso esigenze comuni, nell'ottica di offrire uno strumento flessibile e sempre aggiornato.

BUON LAVORO CON INSTANTSOLUTIONS!