



DelphiDay
italian conference

MMX Code Explorer & DocInsight

Un'accoppiata vincente per mettere il turbo alla produttività



PAOLO ROSSI
WINTech ITALIA - CTO
SENCHA & EMB. MVP



blog.paolorossi.net



paolo@paolorossi.net



twitter.com/awebguy



github.com/paolo-rossi



linkedin.com/in/paolo-rossi-pc



DelphiDay
italian conference

19 Novembre 2025
Padova



wintech
italia



GITHUB PROJECTS



github.com/paolo-rossi



Delphi JWT

JSON Web Token Library



WiRL

REST Library for Delphi



Linux Daemon

Real Linux daemons



Delphi Neon

JSON Serialization Library



OpenAPI-Delphi

OpenAPI 3.0 Library



NATS Delphi

NATS Client Library for Delphi



AGENDA

1. Documentare i sorgenti
 - a. DocInsight
2. Aumentare la produttività
 - a. MMX
 - b. CnPack
 - c. BeyondCompare



Introduction

1



Documentazione sorgente

- Dove?
 - All'interno delle procedure (codice)
 - Descrizione di classi, metodi, parametri
- A chi è rivolta?
 - Sviluppatore
 - Manager
 -nessuno



Come commentare

- Punti dove c'è una condizione pregressa e non chiara dal contesto
- Codice che dipende logicamente da altro codice
- Includere link a referenze esterne (se esistono)
- Usare commenti per marcare implementazioni non complete

Commenta classi, metodi e parametri



Come non commentare

- I commenti non devono duplicare il codice
- Se non riesci a scrivere un commento chiaro forse c'è un problema nel codice
- I commenti devono fugare dubbi non crearli
- Non scrivere le modifiche come commento (con autore, data, ecc...)
 - Esistono i version control!



Come non commentare



```
// Se condizione è vera esegui scelta1 altrimenti esegui scelta 2
if Condizione = True then
    EseguiScelta1()
else
    EseguiScelta2()

i := i + 1; // aggiungi 1 a i

// crea un ciclo for da 1 a 10
for var i := 0 to 10 do
```



XMLDoc

- Standard di Microsoft introdotto in Visual Studio
- Adottato da Delphi all'epoca di Delphi.NET
- L'IDE di Delphi “capisce” nativamente il formato XMLDoc
 - Converte al volo in documentazione (HelpInsight) i commenti
- Sia per sviluppatori (immediato) sia per creare documentazione “persistente”
- Prolisso rispetto ad altre soluzioni (JavaDoc)



Commenti XMLDoc

```
/// <summary>
///   A resource that the server is capable of reading, included in a prompt or tool call result.
/// </summary>
/// <remarks>
///   Note: resource links returned by tools are not guaranteed to appear in the results of
///   `resources/list` requests.
/// </remarks>
TResourceLink = class(TBaseContent)
public

    /// <summary>
    ///   The URI of this resource.
    /// </summary>
    Uri: string;
```



DocInsight

- Plug-in (expert) per l'IDE di Delphi
 - A pagamento
- Versione 3, 4, 5 fino a Delphi 12
- Versione 6 (ora in beta) per Delphi 13
- DevJet Software
 - <https://devjetsoftware.com/>
- Altro prodotto interessante
 - JetPascal (beta)



Perchè usare un tool esterno?

Teorema della pigrizia del programmatore

Se un'operazione è un minimo complessa o
lunga (ed è opzionale) ...**non la si fa!**



Quindi perchè usare un tool?

- Tempo impiegato (a commentare)
- Accuratezza
 - Un tool controlla l'input e minimizza gli errori
- Standardizzazione formato di scrittura
 - Tag apertura, chiusura a capo, ecc...
- In definitiva, ricordando il teorema suddetto

**L'operazione risulta veloce e "piacevole" quindi
... viene eseguita**



Demo:

Demo DocInsight

- ★ Uso basico
- ★ Uso di tutti i tag
- ★ Produzione documentazione



MMX

2



MMX Code Explorer

- Plugin (expert) per l'IDE
- <https://www.mmx-delphi.de/>
- Prima era a pagamento, adesso è free
- Disponibile su GetIt
- Disponibile un normale Windows Installer
- Compatibile con D13 e precedenti



Copia e spostamento entità

- IMHO la funzionalità più utile
 - **"Ctrl-Alt-X"** (Taglia)
 - **"Ctrl-Alt-C"** (Copia)
 - **"Ctrl-Alt-V"** (Incolla)
- Applicabile a... quasi tutto
 - Classi, metodi, proprietà, campi, interfacce, ecc...



Sync e cancellazione entità

- Altre funzionalità essenziali
- Una classe è sempre “divisa” tra interface ed implementation
- Comoda visualizzazione della classe ma richiede una sincronizzazione tra interface ed implementation
- Sincronizzare i parametri dei metodi
 - `Ctrl-Alt-Y`
- Cancellare completamente un metodo/proprietà
 - `Ctrl-Shift-Canc`



Edit... everything

- Funzionalità perfetta per i neofiti ma utile a tutti
 - **Ctrl-E**
- Si può applicare a... quasi tutto



Variabili locali

- Utile per non spostarsi dal contesto
 - **Ctrl-L**
- Possibile creare ed inizializzare variabili
 - Locali
 - Inline
 - Costanti inline



Extract Methods

- Utile per rifattorizzare metodi lunghi e complessi quindi...molto importante!
 - **MMX** -> **Extract Method**
- Finestra di dialogo con tante opzioni per personalizzare al meglio il nuovo metodo



Aggiunta entità

- Altra funzionalità essenziale per chi è alle prime armi sull'Object Pascal e per chi ha la memoria corta :-)
 - **Ctrl-Alt-F** -> aggiunta campo
 - **Ctrl-Alt-M** -> aggiunta metodo
 - **Ctrl-Alt-P** -> aggiunta proprietà
 - **Ctrl-Alt-I** -> aggiunta indexer
 - **Ctrl-Alt-E** -> aggiunta evento



Code Explorer

- Il centro di comando del vostro codice!
- Da qui potete fare tutto
- Le viste sono molto... “piene”, bisogna abituarsi
- Come consiglio potete sostituire la “Structure View” con “Code Explorer”



Live metrics

- Le trovate nel “Code Explorer”
- Informazioni molto utili sulle classi
- Si possono (dis)abilitare tutti gli hint
- Dopo averle lette... bisogna fissarle!



Miscellaneous

- `Ctrl-Alt-J` - Surround with try-finally
- `Ctrl-Alt-Up/Down` - Next/Previous
- `Alt-Shift-R` - Reverse Assignment
- Menu: Text Tools -> Align Code
- Menu: Conversions -> Module String Wizard
- Menu: Tools -> Unit Dependencies
- Tanti altre...



THANK YOU