



Modern Delphi: language and RTL

Tutte le novità su linguaggio e RTL nelle ultime versioni di Delphi



LUCA MINUTI



luca.minuti.it



luca.minuti@gmail.com



dev.to/lminuti



github.com/lminuti



linkedin.com/in/luca.minuti/



DelphiDay
italian conference

19 Novembre 2025
Padova



wintech
italia

OPEN-SOURCE PROJECTS

github.com/Iminuti

WiRL

github.com/delphi-blocks/WiRL

Delphi SAML

github.com/EtheaDev/Delphi-SAML

OpenSSL

github.com/Iminuti/Delphi-OpenSSL

MCPCConnect

github.com/delphi-blocks/MCPCConnect



19 Novembre 2025
Padova





AGENDA

1. What's new in Delphi 13
2. Perché l'RTL è importante
3. L'RTL di Delphi 13
4. Novità nelle versioni più vecchie
5. Cosa ci siamo persi?



WHAT'S NEW!

- New Features in RAD Studio 13
 - Ternary operator
 - 64-bit IDE
 - New constraints for generic type
 - VCL Enhancements (TitleBar, ControlList, EdgeBrowser, WinUI 3, ...)
 - SmartCore AI Component Pack
 - JSON enhancements
 - WebBroker and WebStencil update

https://docwiki.embarcadero.com/RADStudio/Florence/en/What%27s_New



IL LINGUAGGIO

- Coerente e con una quasi totale compatibilità col passato
- Dalla prima versione ci sono stati numerose modifiche: interfacce, array dinamici, operator overloading, default parameters, method overloading, nested class, record helper, generici, metodi anonimi, unicode, inline variables, supporto win64, linux, MacOS, Android, iOS, ecc.



RTL - RUNTIME LIBRARY

- Al centro dell'esperienza d'uso insieme con il linguaggio
- Centinaia di classi per i problemi più comuni
- Permette di scrivere meno codice e di aderire a pratiche più moderne
- Miglioramenti di sicurezza



DELPHI 13 RTL

1



TERNARY OPERATOR

- Delphi utilizza una forma speciale della parola chiave **if**, ad esempio: `X := if Left < 100 then 22 else 45`
- Le due espressioni del ramo then e else devono essere dello stesso tipo
- L'operatore if-then-else ha una priorità bassa rispetto ad altri operatori (come +), quindi è consigliato l'uso di parentesi per evitare risultati inattesi.



IS NOT

- Adesso è possibile usare “is not” al posto di “not (x is y)”

```
if not (Obj is TFoo) then ... // classic syntax
```

```
if Obj is not TFoo then ... // new syntax
```

```
if not (x in [Apple, Orange]) then ... // classic syntax
```

```
if x not in [Apple, Orange] then ... // new syntax
```



NORETURN

- Serve per indicare che una procedura non termina l'esecuzione normale, ma genera un'eccezione
- Aiuta il compilatore a evitare avvisi su risultati non definiti e consente migliori ottimizzazioni del codice
- È utile nelle funzioni che sollevano eccezioni e non restituiscono un valore



GENERIC CONSTRAINTS

- Nuovi constraint per i generici: **interface** e **unmanaged**
 - unmanage sostituisce record
 - interface controlla che sia effettivamente una interfaccia

```
TMyClass1<T:IFoo> = class  
end; // Old
```

```
TMyClass2<T:IFoo, interface> = class  
end; // New
```

```
C1: TMyClass1<IFoo>; // ok
```

```
C2: TMyClass2<IFoo>; // ok
```

```
C3: TMyClass1<TFoo>; // ok
```

```
C4: TMyClass2<TFoo>; // Error E2667
```



NAMEOF FUNCTION

- Restituisce la stringa che rappresenta il nome dell'identificatore

```
Memo1.Lines.Add (NameOf (Sender));  
Memo1.Lines.Add (NameOf (self));  
Memo1.Lines.Add (NameOf (TForm31));  
Memo1.Lines.Add (NameOf (X));  
Memo1.Lines.Add (NameOf (Integer));
```



IMPLICIT “SELF” FOR RECORD

- Per le funzioni Initialize e Finalize dei record non è più obbligatorio il parametro Self

```
TRec = record  
  FIntData: Integer;  
  class operator Initialize; // equiv. Initialize(out Self: TRec);  
  class operator Finalize; // equiv. Finalize(var Self: TRec);  
  procedure Test;  
end;
```



SMARTCORE AI COMPONENTS

- Nuova libreria disponibile su **GetIt** per il supporto AI:
 - *OpenAI, Gemini, Claude, Ollama*
- Concetto simile ai database:
 - Driver, Connection, Chat/Image
- Risposte asincrone (tramite eventi)



JSON ENHANCEMENTS

- JSON Processing improvements
- New TJsonTimestamp
- New converter for Serialization: TList, TDictionary, TStack
- TJSONNumber doesn't inherit from TJSONValue (TJSONString)
- TJSONBool.Create a TJSONTrue or TJSONFalse
- TJsonReferenceHandling, [JsonValueSerialize]



ALTRO

- Added a case-insensitive version of the TStringHelper.Contains function
- IOUtils: TMatchCasing, TFile.GetEncoding, TPath.Exists
- NetHttpClient support HTTP3
- [https://docwiki.embarcadero.com/RADStudio/Florence/en/What%27s New](https://docwiki.embarcadero.com/RADStudio/Florence/en/What%27s%20New)



DELPHI 12 RTL

1



STRING LITERAL

- Prima di Delphi 12: 255 caratteri (come le short string)
- Long string: supporto fino a 4k per linea
- Multi line string:
 - le stringhe multilinea, racchiuse tra virgolette triple (“”).
Possono anche essere di più (ma sempre dispari 5, 7, ecc.)
 - Non è più necessario unirle con il '+'.



STRING LITERAL

- Il carattere di chiusura `` deve trovarsi in una riga a sé stante
- Il rientro del carattere di chiusura `` determina il rientro di base dell'intera stringa.
- Ogni spazio vuoto prima di tale livello di rientro viene rimosso
- Nessuna riga può essere meno rientrata del rientro di base
- L'ultima nuova riga prima del carattere di chiusura `` viene omessa



STRING LITERAL

- Nuova direttiva di compilazione **TEXTBLOCK**
 - `{$TEXTBLOCK NATIVE/CR/LF/CRLF [<ident>]}`
- Il parametro aggiuntivo <ident> è una decorazione, ignorata dal compilatore ma eventualmente utilizzata da strumenti esterni. Può essere HTML, XML, JSON, SQL, ecc.



STRINGS E STREAM

- TStrings:
 - Contains, ContainsName, ContainsObject
- TStream
 - TProxySubrangeStream
 - TProxyAggregateStream



DICTIONARY

- I dictionary non garantiscono il mantenimento dell'ordine. Se vengono letti, per esempio con una `for...` in l'ordine può essere diverso da come sono stati inseriti
- Due nuovi dictionary
 - `TOrderedDictionary`
 - `TObjectOrderedDictionary`



RTTI HELPER

- Nuovo TRttiTypeHelper:
 - GetMethodsInDeclOrder, GetFieldsInDeclOrder, GetPropertiesInDeclOrder, GetIndexedPropertiesInDeclOrder



IOUTILS - DIRECTORY

- TDirectory.Copy
- TDirectory.GetDirectoriesEnumerator
- TDirectory.GetFilesEnumerator
- GetFileSystemEntriesEnumerator



IOUTILS - PATH

- TPath.Combine (more overload)
- TPath.GetDesktopPath
- TPath.GetAppPath



IOUTILS - FILE

- Adesso è possibile leggere un file composto da righe separate da invio con una `for..in` senza caricarlo tutti in memoria:
 - `TFile.GetLinesEnumerator`



ENUMERATOR

- Nuovi enumerator su Components e Controls:
 - `for var MyComponent in Form1 do`
 - `for var MyControl in Panel.GetControl(...);`
- Possibilità di filtrare con un metodo anonimo



INTERFACCE

- Alcune volte può essere utile implementare un'interfaccia ma senza usare il reference count.
 - deprecated TSingletonImplementation use TNoRefCountObject
- **ATTENZIONE:** viene comunque chiamato il metodo Release quando esce dallo scope



FNV - HASH

- Fowler–Noll–Vo (o FNV, dai cognomi degli ideatori)
- FNV è un algoritmo di hashing non crittografico (no password e simili)
- È ampiamente utilizzato in tabelle hash, checksum e identificatori
- Supporta diverse dimensioni di hash come 32, 64, 128, 256, 512 e 1024 bit
 - THashFNV1a64
 - THashFNV1a32 (già in Delphi 11)



IN GENERALE

- Molti indici da integer a NativeInt:
 - TList, TObjectList, TStringList, ecc.
- Molte funzioni inline
 - es. TStream
- IsEmpty su molte liste:
 - TList, TStringList, TObjectList, ecc.

demo time





DELPHI 10 e 11

2



UTILITY

- IsValidIdent: returns true if the given string is a valid identifier
- TCollection.Sort, TZipFile Delete (by index or name)
- new TPointerStream (like memory stream)
- DateUtils.HttpToDate
- TArray.Concat
- TDirectoryHelper.Copy
- TNoRefCountObject



CREATE ORDER

- Eliminata OldCreateOrder da Form e DataModule
- Con OldCreateOrder (default fino a Delphi 3) il codice dell'evento **OnCreate** partiva prima del codice inserito dopo inherited sul **costruttore**.



CLX?

```
{ Call CheckSynchronize periodically within the main thread in order for  
background threads to synchronize execution with the main thread. This  
is mainly for applications that have an event driven UI such as Windows  
or XWindows (Qt/CLX). The best place this can be called is during Idle  
processing. This guarantees that the main thread is in a known "good"  
state so that method calls can be safely made. Returns True if a method  
was synchronized. Returns False if there was nothing done.  
,
```

```
{ Call CheckSynchronize periodically within the main thread in order for  
background threads to synchronize execution with the main thread. This  
is mainly for applications that have an event driven GUI such as Windows  
VCL or multi-platform FMX ones. The best place this can be called is durin  
Idle processing. This guarantees that the main thread is in a known "good"  
state so that method calls can be safely made. Returns True if a method  
was synchronized. Returns False if there was nothing done.  
}
```



FNV - HASH

- Fowler–Noll–Vo (o FNV, dai cognomi degli ideatori)
- FNV è un algoritmo di hashing non crittografico (no password e simili)
- È ampiamente utilizzato in tabelle hash, checksum e identificatori
- Veloce e con rare collisioni
- Diverse dimensioni di hash come 32, 64, 128, 256, 512 e 1024 bit
 - THashFNV1a64 (introdotto in Delphi 12)
 - THashFNV1a32 (introdotto in Delphi 11)



JSON

- Encoding options (EncodeBelow32, EncodeAbove127)
 - Permettono di avere un controllo fine di come comportarsi con i caratteri con codifica sotto 32 (caratteri di controllo) e sopra 127 (caratteri speciali: accentate, simboli, ecc.)
- TJSONPair diversi costruttori che permettono di creare il “Pair” senza dove prima creare un JSONValue
- TJSONObject diversi override di AddPair (come sopra)



NET ENCODING

- Due nuovi encoding predefiniti. Prima era necessario creare una nuova istanza specifica.
 - Base64String: base64 ma senza ritorni a capo (76)
 - Base64URL: base64 senza ritorni a capo inoltre '+' e '/' sono sostituiti da '-' e '_',) e usato per esempio da YouTube quando è necessario mettere una stringa in base64 in un URL



RTTI OBJECT

- Nuovi metodi per TRttiObject:
 - `function GetAttribute(AAttrClass: TCustomAttributeClass): TCustomAttribute;`
 - `function GetAttribute<T: TCustomAttribute>: T;`
 - `function HasAttribute(AAttrClass: TCustomAttributeClass): Boolean;`
 - `function HasAttribute<T: TCustomAttribute>: Boolean;`



TLS 1.3

- Aggiunto il supporto per TLS 1.3 a THttpClient
 - THttpClient è il nuovo componente HTTP che usa le API del sistema operativo e non necessita di OpenSSL per il supporto TLS
- I componenti Indy non supportano ancora TLS 1.3 nativamente ma:
 - Separazione package con il supporto a OpenSSL in modo da poterlo aggiornare indipendentemente dal resto
 - Progetto indipendente: TaurusTLS (disponibile su GitHub e su GetIt)



CURRENCY HELPER

```
function ToString(const AFormatSettings: TFormatSettings): string; overload; inline;  
function ToString: string; overload; inline;
```

```
function Frac: Currency; inline;  
function Ceil: Int64; inline;  
function Floor: Int64; inline;  
function Trunc: Int64; inline;
```

```
class function Size: Integer; inline; static;  
class function ToString(...): string; overload; inline; static;  
class function Parse(...): Currency; overload; static;  
class function TryParse(...): Boolean; overload; inline; static;
```

demo time





MODERN DELPHI

3



STREAM / READER / WRITER

- Sintassi pascal classica:

```
AssignFile(F, NomeFile);  
Rewrite(F);  
WriteLn(F, 'Hello, World!');  
CloseFile(F);
```

- FileStream, StringStream, MemoryStream, ResourceStream



UNIT: DATE UTILS

- Contiene centinaia di funzioni per le date:

```
function YearOf(const AValue: TDateTime): Word;  
function MonthOf(const AValue: TDateTime): Word;  
function EndOfTheMonth(const AValue: TDateTime):  
TDateTime;  
function WeekOfTheYear(const AValue: TDateTime): Word;  
overload;  
function MonthsBetween(const ANow, AThen: TDateTime):  
Integer;
```



HTTP CLIENT

- THttpClient è il nuovo componente HTTP che usa le API del sistema operativo e non necessita di OpenSSL per il supporto TLS
- Utile specialmente su mobile (installare OpenSSL non è sempre semplice)
- Sposta la responsabilità dell'aggiornamento al SO
- Sintassi moderna (Interfacce, TArray, ...)



REGULAR EXPRESSION

- Strumento di ricerca e pattern matching
- Usano simboli speciali come *, +, ?, [] per definire regole di corrispondenza (es. `\d+` per numeri)
- Validazione email/telefoni, estrazione dati, ricerca e sostituzione di testo, parsing di log
- Disponibili in quasi tutti i linguaggi di programmazione e editor di testo moderni



ITERATORS

- Gli iteratori permettono di eseguire dei loop senza usare un indice
- Di solito per array o liste ma non solo
- Sono presenti in molte classi standard ma possono anche essere implementati in classi custom
- Si usano con la sintassi **for..in**



GENERICI

- Tramite i generici è possibile creare una classe che manipola dati di diverso tipo senza doverne scrivere una per tipo
- L'uso più comune è con le liste:
 - TList, TObjectList, TThreadList
 - TDictionary, TObjectDictionary, **TOrderedDictionary**
 - TQueue, TStack (object e threaded version)
 - TArray, THashSet

demo time





DELPHI HISTORY

4



Delphi 2005

- **for ... in loops**
- inline keyword
- nested types
- nested constants
- {\$REGION}/{\$ENDREGION} directives



Delphi 2007

- Enhanced records
- Operator overloading
- Static methods and properties
- **FastMM** is the default memory manager
- Strict private/protected visibility keyword
- Final keyword for virtual methods



Delphi 2009

- Generics
- Anonymous methods
- Support for nested exceptions and exception tracing
- **Exit function can take a parameter specifying a result**
- Deprecated keyword can now have additional text



Delphi 2010

- New Delphi RTTI (Run Time Type Information)
- Attributes
- The as and is operators can be used on interfaces
- Casting can be performed on an interface: `TObject(MyInterface)`
- New delayed directive to load external libraries at first call
- Class Constructor/Destructor



Delphi XE

- **Regular Expressions Library**
- TTimeZone, TBinaryReader & TBinaryWriter
- Refactoring and improvements on TThread and Sync classes
 - TThread.Suspend and TThread.Resume deprecated
 - Proper thread synchronization based on TEvent and TMutex
 - New classes for thread synchronization
 - TInterlocked, TSpinWait, TSpinLock



Delphi XE2

- FireMonkey
- Support for Win64
- LiveBindings
- **Full unit scope names are now required in your uses clause**
- New System.Zip unit



Delphi XE3

- **Record helpers for built-in types**
- Atomic intrinsic functions:
 - AtomicExchange()
 - AtomicIncrement()
 - AtomicCmpExchange()
 - AtomicDecrement()
- Introduction of the [ref] attribute



Delphi XE4

- New dependency directive
 - Specifying the dependencies of an external library



Delphi XE5

- New REST Client Library added
- System.Types
- Operator overloading for classes
 - Only for the NextGen compilers Android/iOS



Delphi XE6

- JSON framework refactored
 - System.JSON introduced
 - REST.Json introduced



Delphi XE7

- **String-Like Operations Supported on Dynamic Arrays**
- Parallel Library added to the RTL
- TNetEncoding



Delphi XE8

- **New HTTP Client API**
- New Hash API
- Asynchronous Programming Library
- New integer types
 - FixedInt, FixedUInt => 32-bit integer types on all platforms
- New platform dependent integer types
 - LongInt, LongWord => 64-bits on iOS-64, 32-bits on all other platforms



Delphi 10

- Support for Android 5.1.1 and iOS 8.4
- Improved OSX exception handling



Delphi 10.1

- Native support for Utf8String and RawByteString type on all platforms
- The [weak], [unsafe] and [volatile] attributes are supported on all compilers
- Class and record helpers cannot access private members of the classes or records they extend



Delphi 10.2

- Support for Linux server apps (Intel 64-bit using LLVM and ARC).
- Assigning a dynamic arrays to a pointer using the @ operator is only allowed when hard-casting the array.
- More flexible namespace resolution of unit names



Delphi 10.3

- **Inline Variable Declaration**
- **Type inference**
- AnsiString / AnsiChar on Linux
- Traditional Memory Reference Counting in the Linux compiler



Delphi 10.4

- **Custom managed records**
- Unified memory management on all platforms - full ARC compiler has been removed and all compilers
- Support for Android, macOS 64-bit



Delphi 11

- **Binary Literals and Digit Separators**
- **New record helpers: TDateTimeHelper and TCurrencyHelper**
- Inline assembler support for AVX instructions (AVX-512)
- macOS ARM 64-bit target platform



Delphi 12

- **Long and multiline string literals**
- **NativeInt/NativeUInt have been downgraded to weak aliases**
- The Delphi32 compiler now handles NaN comparisons according to
- IEEE754

demo time





THANK YOU