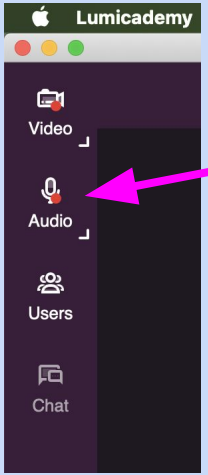
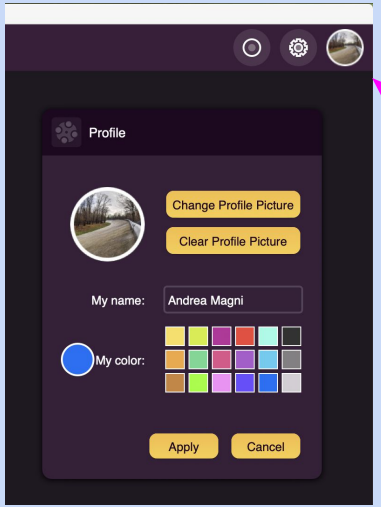
- La piattaforma permette ai partecipanti di condividere la vostra webcam e il microfono
  - Per cortesia, aprite il microfono solo per fare domande
- Per accedere alla chat Slack:
  - https://andreamagni.eu/live.html



Pallino rosso = spento, click per attivare (vale per Video e Audio)



Potete impostare il vostro nome utente, se volete

Delphi Live - Italia
03 Giugno 2020

Andrea Magni
me@andreamagni.eu

Andrea Magni

Ingegnere informatico, Monza

Sviluppatore, Formatore, Consulente

Desktop, Server, Web, Mobile e IoT

embarcadero MVP Delphi

TFrameStand, TFormStand, FMXER (FMX)

MARS-Curiosity (REST Library)

FMX Book (coming)

https://andreamagni.eu

Piattaforma Lumicademy

- https://www.lumicademy.com/
- https://blog.grijjy.com/
  - Allen Drennan
  - Erik van Bilsen
- Applicazioni FMX
  - Windows e OS X
  - con integrazioni a basso livello (GPU, video)

Delphi Live Events

Alessandro Bruzzone

Sviluppatore Delphi sin dalla versione 1

Responsabile Sviluppo Software presso

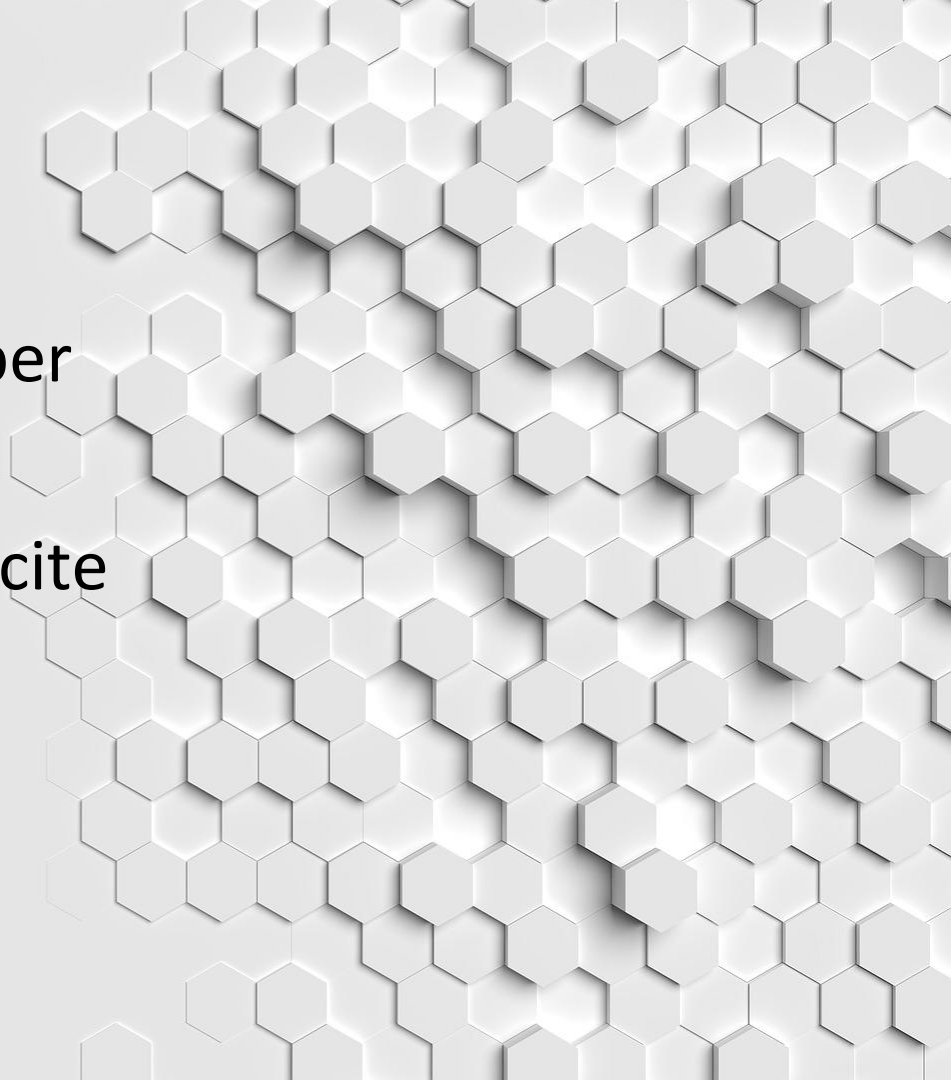
Uakari Software S.r.l.

Soluzioni con Delphi per la GDO livello nazionale

Tra i Clienti principali: Conad, Crai, Sigma e altri

Agenda

- Codice più leggibile ?
- Class helper & Record helper
- Type system
- Record e conversioni implicite
- Record Vs Class
- Domande e saluti



Codice più leggibile ?

Partiamo dai commenti…

Il commento utile è quello che spiega il **“perché”**

e non il **“cosa”** si sta facendo

“Il Codice si deve commentare da solo “

Capire il codice a colpo d’occhio

Codice facilmente leggibile

Migliore manutenzione ed efficienza nella sua estensione/modifica

Rischio minore di introdurre nuovi bug

(solo) tre regole fondamentali...

Capire il codice a colpo d’occhio

Regola 1

Usare un nome sensato per tutti gli identificatori

```
procedure Sel(par1, par2: string);
```

❓❓❓

```
procedure FindCustomer(aName, aSurname: string);
```

✔️

Capire il codice a colpo d’occhio

Regola 2

parametri Boolean ?

```
Sostituisci(‘abcd’, ‘a’, True);
```

❓ ❓ ❓

meglio... parametri Enum ?

```
Sostituisci(‘abcd’,’a’, stCaseSensitive)
```

✔️

Migliore lettura, maggiore praticità di estensione

Capire il codice a colpo d’occhio

Regola 3

Copia & Incolla ?

NO!

E’ uno dei principali responsabili di funzioni composte da “migliaia” di righe di codice

meglio puntare su procedure atomiche

Per iniziare… i Class Helper

Possono essere utili ?

un esempio...

Caratteristiche base dei Class helper

- Ci permettono di estendere qualsiasi classe senza derivare
- Non occorre modificare il tipo nelle dichiarazioni esistenti
- Accesso agli elementi public e protected (no private)
- Non si possono dichiarare variabili
- La visibilità di un helper è a livello di unit
- Può essere usato un solo helper per ogni classe (vince l'ultimo)

Un passo avanti… i Record helper

Stessa logica dei Class helper

Si associano ai record ovviamente…

ma non solo

I tipi di dato “nativi”

Cosa intendiamo per “nativi” ?

Tipi di dato che non richiedono di essere esplicitamente allocati in memoria

ci pensa il compilatore…

Qualche esempio:

- Integer
- Double
- Boolean
- TDateTime
- string (...?!?!)

Record helper sui tipi “nativi”

Definire metodi su dati nativi = OOP

Migliore stile nella scrittura del codice

Migliore leggibilità del codice

Record helper sui tipi “nativi”

Chiamate annidate Vs Chiamate a cascata *(Fluent)*

Chiamate annidate

```
newValue := Uppercase(AnsiReplaceStr(Trim(myValue), ‘search’ , ‘repl’)));
```

Chiamate a cascata

```
newValue := myValue.ToUpper.Replace(‘search’,’repl’).Trim;
```

Trova la differenza....

Record helper sui tipi “nativi”

Chiamate annidate Vs Chiamate a cascata *(Fluent)*

Chiamate annidate

```
newValue := Uppercase(AnsiReplaceStr(Trim(myValue), ‘search’ , ‘repl’)));
```

Chiamate a cascata con stesso ordine di esecuzione

```
newValue := myValue.Trim.Replace(‘search’,’repl’).ToUpper;
```

l’ordine di apparizione dei metodi coincide con l’ordine delle chiamate

Un “nuovo” modo di usare i record

Definizione di nuovi tipi “nativi” personalizzati

Andiamo “oltre” la lettura a colpo d’occhio del codice…
Andiamo “oltre” i commenti…

E’ il codice stesso che ci guida!

Un “nuovo” modo di usare i record

La chiave di tutto è l’overload degli operatori

Definire dati anche complessi con normali assegnazioni (:=) di “valori semplici”

Scrivere condizioni semplici ed intuitive (= , < , >, <= , >=)
coinvolgendo in realtà funzioni anche complesse

Definire conversioni implicite “ad-hoc” da e verso tutti i tipi di dato che ci occorrono
senza dover chiamare metodi espliciti di conversione
(occhio però a non farci prendere troppo la mano)

`http://docwiki.embarcadero.com/RADStudio/Sydney/en/Operator_Overloading_(Delphi)`

Record Vs Class

- Allocazione nello stack = maggiori prestazioni
- Overload operatori = maggiore leggibilità e usabilità
- Ottimo per i tipi di dato “primitivi” del modello dati
- Property e parametri con dichiarazioni specifiche
- Conversioni, formattazioni e altro … impliciti
- Con Delphi 10.4 anche inizializzazioni, finalizzazioni a assegnazioni (copia) personalizzate

`http://docwiki.embarcadero.com/RADStudio/Sydney/en/Custom_Managed_Records`

Record Helper

- E infine con i record helper: librerie general purpose per favorire chiamate a cascata (Fluent)

un esempio per giocare un pò...

`https://github.com/bruzzoneale/delphi-scl`

Grazie