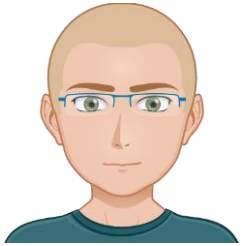
Crea una «CLI»

che i tuoi utenti ameranno

Mi sentite?



Marco Breveglieri

Software and Web Developer, Trainer and Consultant

@ABLS Team – Reggio Emilia

Progetto e sviluppo software.
Organizzo corsi e consulenze.
Mangio sushi. 😉

👉 **https://www.breveglieri.it**

DelphiDay italian conference

ONLINE EDITION

wintech italia

Delphi Podcast

Il primo podcast italiano dedicato a **Delphi**

a cura di **Marco Breveglieri**

👉 **https://www.delphipodcast.com**

DelphiDay italian conference

ONLINE EDITION

wintech italia

Syncfusion
Deliver innovation with ease

Delphi in
100 pagine

https://www.syncfusion.com/ebooks/delphi

DelphiDay
italian conference
ONLINE EDITION

wintech
italia

AGENDA

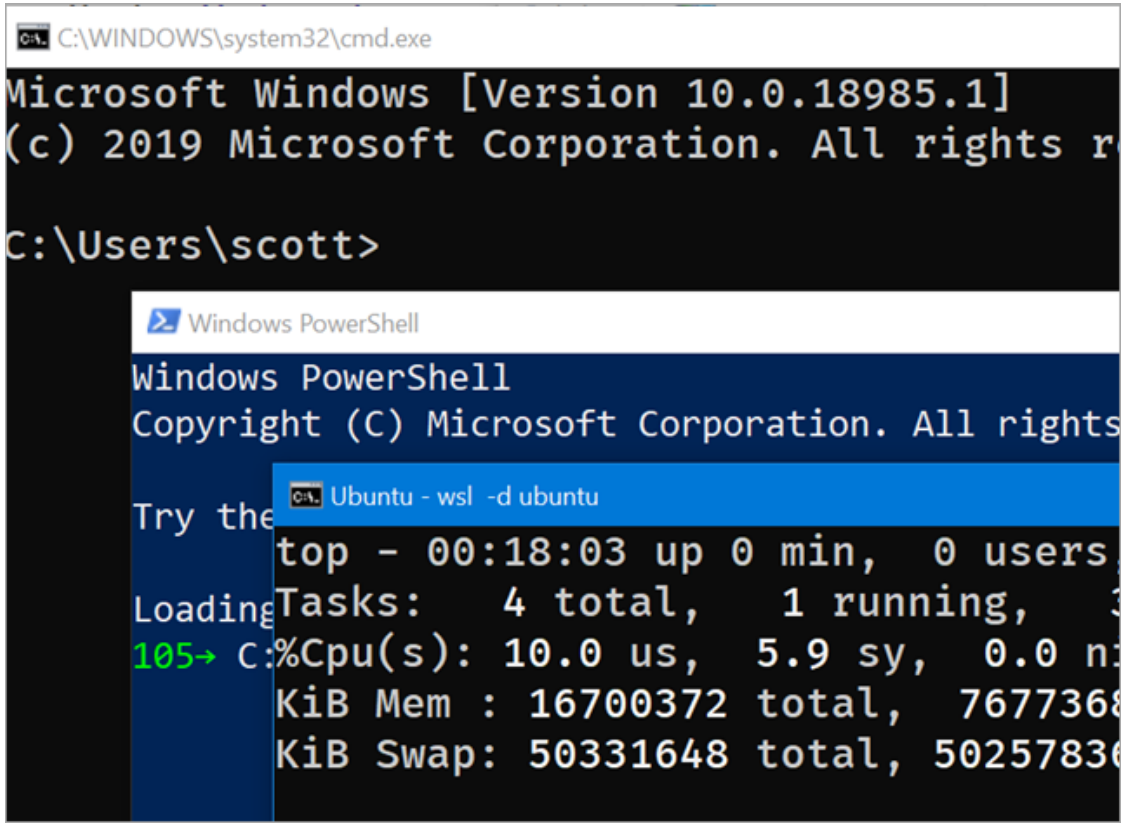
- **Introduzione**: le migliori «CLI» moderne, nuova vita al terminale!
- **Benefici (e svantaggi)**: come e quando creare una CLI
- **Progettazione e design**: i comandi, le opzioni, la guida, i codici di errore
- **Creazione app in Delphi**: esempi pratici dalle basi ai concetti avanzati
- **Usciamo dal desktop**: rendere invocabile la CLI da pagine Web
- **Il tocco finale**: dal setup agli abbellimenti irrinunciabili
- **Conclusioni**: recap, Q&A e saluti

C’era una volta…

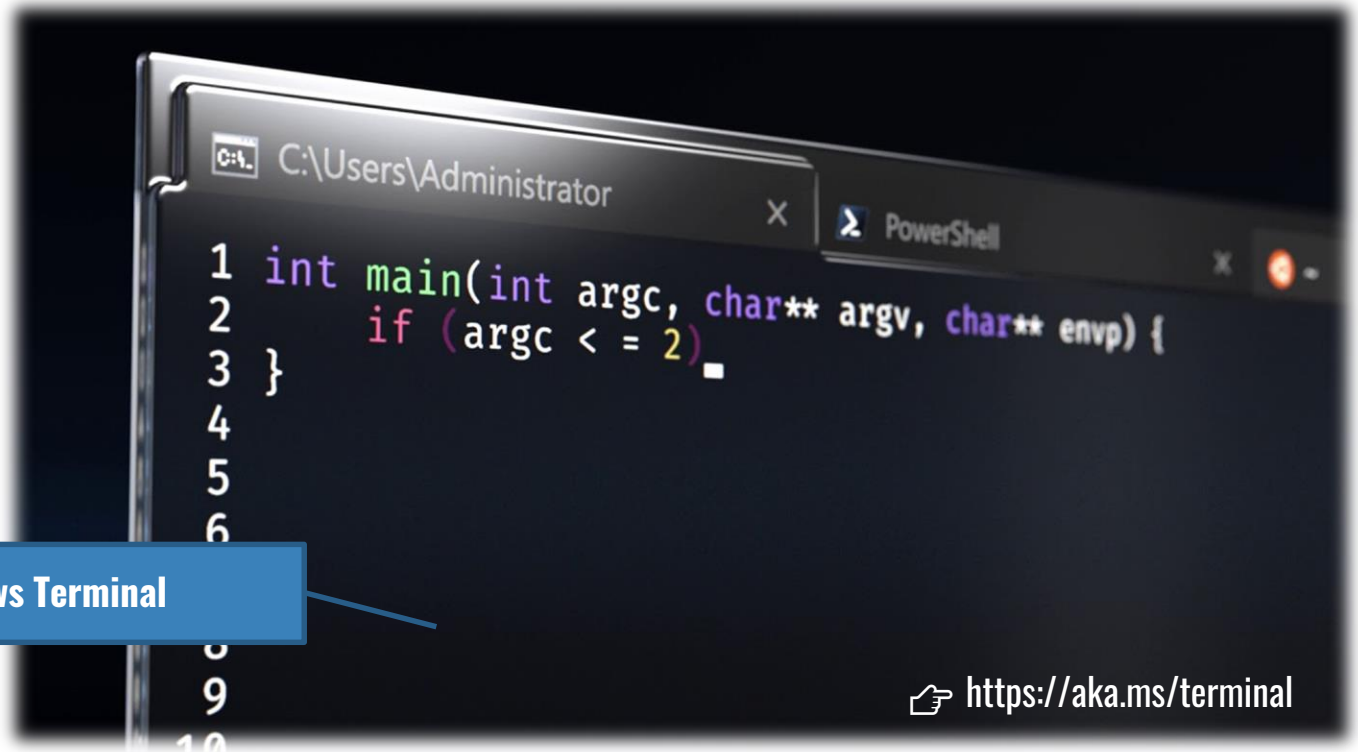
In principio fu…

…poi venne…

…fino a oggi…


C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.18985.1]
(c) 2019 Microsoft Corporation. All rights r
C:\Users\scott>
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights
Try the
Loading
105→ C:
Ubuntu - wsl -d ubuntu
top - 00:18:03 up 0 min, 0 users,
Tasks: 4 total, 1 running,
%Cpu(s): 10.0 us, 5.9 sy, 0.0 ni
KiB Mem : 16700372 total, 7677368
KiB Swap: 50331648 total, 5025783

…e oltre!


C:\Users\Administrator
PowerShell
1 int main(int argc, char** argv, char** envp) {
2 if (argc < = 2)
3 }
4
5
6
9
NUOVO! Windows Terminal
https://aka.ms/terminal

Piccolo glossario

- **Shell**: interfaccia utente che consente l'accesso ai servizi offerti da un sistema operativo
- **CLI**: tipo di interfaccia (in questo caso, a riga di comando) usata dall'utente per interagire con la shell
- **Terminale**: programma che apre una finestra grafica e consente di interagire con la shell

Primi passi con le CLI

Definizione

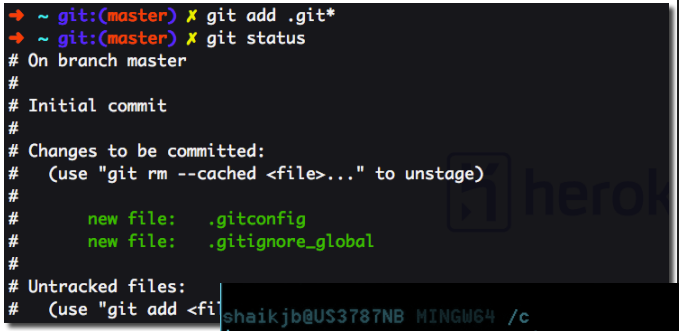
«Una interfaccia a riga di comando (dall'inglese **CLI**, *Command Line Interface*) è un tipo di interfaccia utente caratterizzata da una interazione testuale tra utente ed elaboratore (ad esempio, tramite una shell)».

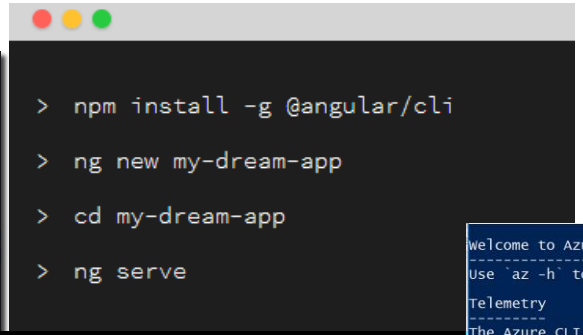
Cosa si può fare?

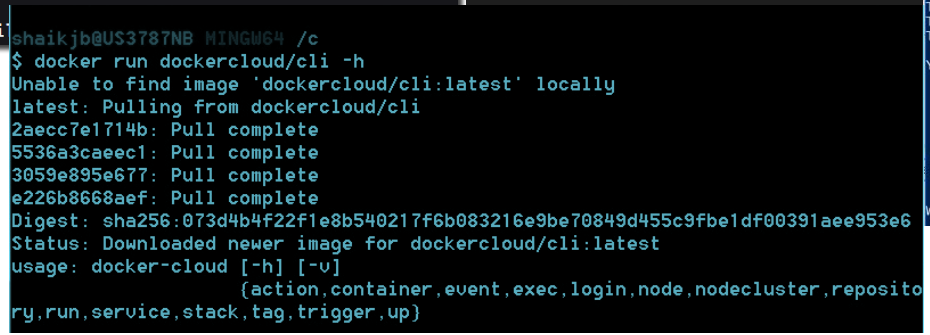
- ➔ Impartire comandi al sistema operativo
- ➔ Creare nuovi documenti (progetti, file, ecc.)
- ➔ Gestire il codice sorgente (es. usando «git-cli»)
- ➔ Avviare il download o la conversione di file dal Web
- ➔ Giocare ai videogiochi (o programmarli)

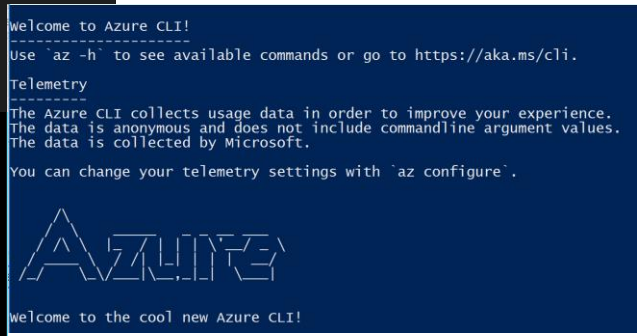
...e molto altro ancora!

Esempi di CLI /1









Esempi di CLI /2

```
PS C:\dev\scratch\dotnet-add> dir
PS C:\dev\scratch\dotnet-add> dotnet new
Created new C# project in C:\dev\scratch\dotnet-add.
PS C:\dev\scratch\dotnet-add> dotnet restore
  Restoring packages for C:\dev\scratch\dotnet-add\dotnet-add.csproj...
  Generating MSBuild file C:\dev\scratch\dotnet-add\obj\dotnet-add.csproj.nuget.g.props.
  Generating MSBuild file C:\dev\scratch\dotnet-add\obj\dotnet-add.csproj.nuget.g.targets.
  Writing lock file to disk. Path: C:\dev\scratch\dotnet-add\obj\project.assets.json
  Restore completed in 1.03 sec for C:\dev\scratch\dotnet-add\dotnet-add.csproj.

  NuGet Config files used:
      C:\Users\steve\AppData\Roaming\NuGet\NuGet.Config
      C:\Program Files (x86)\NuGet\Config\Microsoft.VisualStudio.Offline.config

  Feeds used:
      https://api.nuget.org/v3/index.json
      C:\Program Files (x86)\Microsoft SDKs\NuGetPackages\
PS C:\dev\scratch\dotnet-add> dotnet build
Microsoft (R) Build Engine version 15.1.539.38876
Copyright (C) Microsoft Corporation. All rights reserved.

  dotnet-add -> C:\dev\scratch\dotnet-add\bin\Debug\netcoreapp1.0\dotnet-add.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)
```

```
Course: 1
Distance: 1
Computer reports navigation error; Enterprise stopped at 6,9

Command: srscan

Short range sensor scan
  0 1 2 3 4 5 6 7 8 9
0 * . . . . . . . . 0   stardate      2102.15
1 . . . . . . . . * 1   condition     GREEN
2 . . . . * . . . . 2   position      0,0/6,9
3 . . . . * * . . . 3   warp factor  5.0
4 . . . . . . . . . 4   total energy  4061
5 * . . . . . . . . 5   torpedoes     10
6 . . . * . * . . . E 6 shields       up, 100%
7 . . . . . . . . . 7   Klingons left 5
8 . . . . . . . . * . 8 time left     5.85
9 . . . . . . . . . . 9 life support  active
  0 1 2 3 4 5 6 7 8 9

Command: move
                                    ports navigation error; Enterprise stopped at 6,9
```

```
Documents — [redacted] — ~/Documents — -zsh — 83×27
➜  Documents monaca create CLI/
? Select an option: Sample Apps
? Select a sample app: jQuery TODO App

Downloading template...

Project created successfully.

Type "cd CLI/" and run monaca command again.
  > monaca preview       => Run app in the browser
  > monaca debug         => Run app in the device using Monaca Debugger
  > monaca remote build  => Start remote build for iOS/Android/Windows
  > monaca upload        => Upload this project to Monaca Cloud IDE
➜  Documents
```

Pro e contro

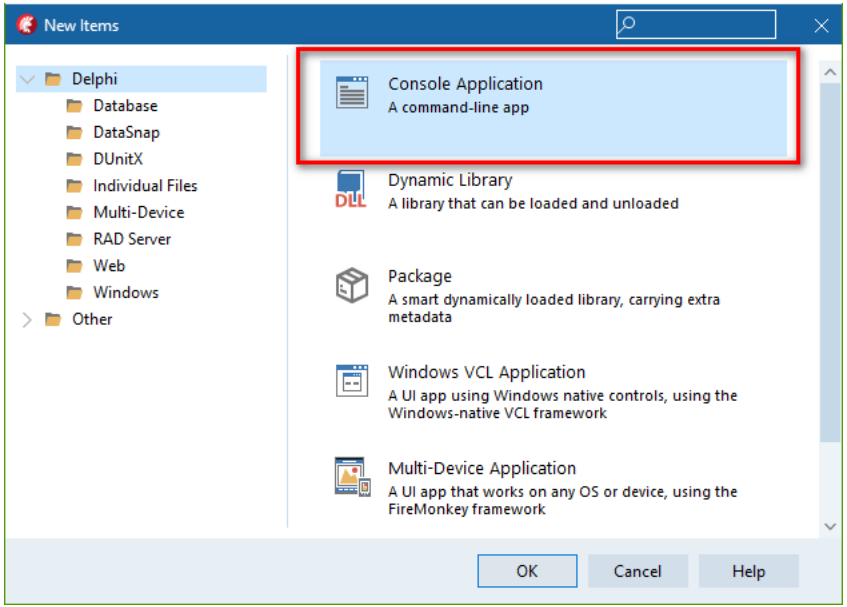
- ➔ Raggiungere una maggiore produttività
- ➔ Automatizzare task ripetitivi
- ➔ Accedere a funzionalità non offerte dalla GUI tradizionale
- ➔ Puoi apparire come un vero «mago» del computer

- ➔ Non è intuitiva per utenti base
- ➔ Curva di apprendimento più ripida (richiede un pochino di esercizio)
- ➔ Interattività limitata (pensiamo a una griglia)
- ➔ Alcuni terminali hanno supporto limitato

Dalla teoria alla pratica

Primi passi

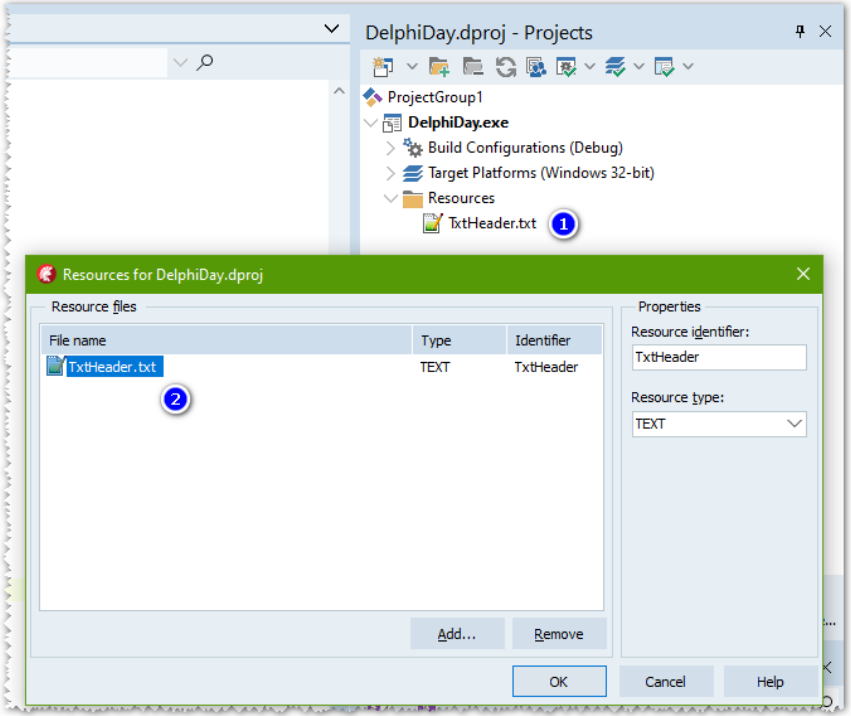
- Crea una nuova «Console Application» in Delphi
- Scrivere il codice principale nella sezione `begin…end`
- Leggere il numero totale di parametri tramite la variabile globale `ParamCount`
- Leggere i valori dei singoli parametri tramite `ParamStr()`
- Scrivere nella console usando `WriteLn()` e leggere con `ReadLn()`



Prologo e guida

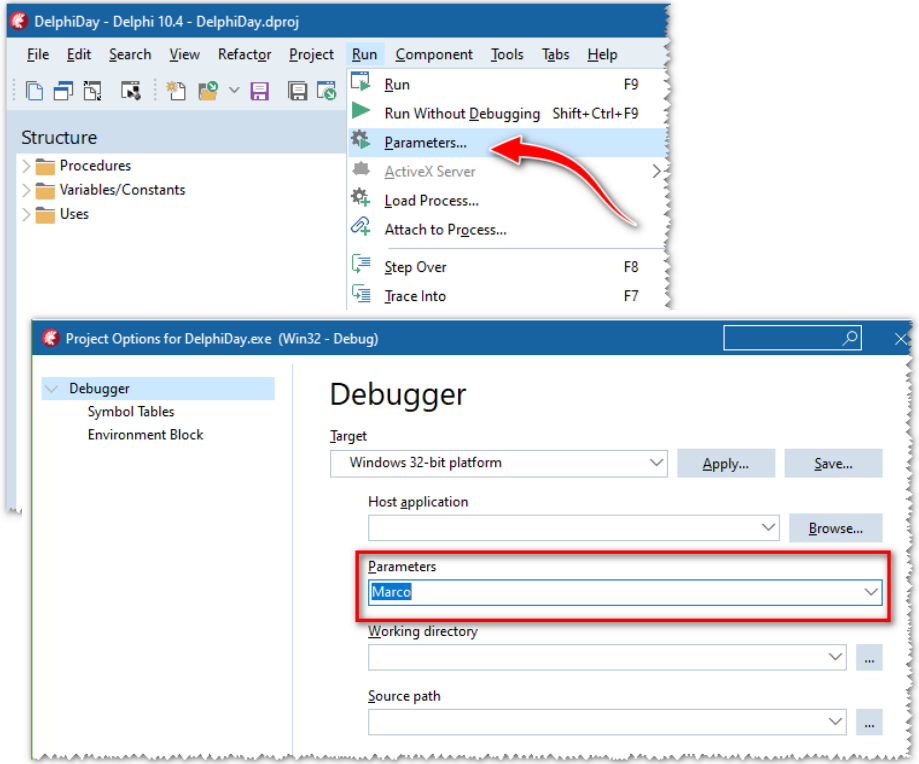
- ➔ Aggiungi al progetto file di testo con header, loghi, guida ai parametri, ecc.
- ➔ Censisci i file come risorse del progetto tramite il menu **[Project|Resources and Images...]**
- ➔ Estrai le informazioni a runtime usando la classe **TResourceStream**

- ➔ Non usare **WriteLn()** per le stampe: meno gestibile e più difficile da formattare!



Eseguire il debug

- ➔ Impostare i parametri da riga di comando tramite **[Run|Parameters...]**
- ➔ Inserire i valori nella casella **Parameters** così come verrebbero indicati direttamente sulla riga di comando
- ➔ Lanciare l'applicazione con **[Run|Run]** (oppure **[Run|Run Without Debugging]**)



JOIN ME
IN DEMO TIME

Inoltre…

- **Gestire sempre le eccezioni!**
  - Se una eccezione «sfugge», si verifica un runtime error e potrebbe apparire la finestra di «Segnalazione errori» di Windows.
- **Verificare la validità dell'input**
  - Essendo che l'utente inserisce prevalentemente testo, questo va sempre verificato (ad esempio, qualora occorra convertire i dati in un formato numerico)
  - Usare i metodi TryParse() oppure Regex per agevolare il compito (e ridurre il codice)
- **Fornire sempre un testo di aiuto come guida alle opzioni e ai comandi**

Progettazione di una CLI

Prima del coding…

- ➔ Analizzare i requisiti, gli obiettivi e le finalità da esporre tramite CLI
- ➔ Scegliere una modalità di interazione (es. «spot», «con sessione», «mista»)
- ➔ Definire i comandi da implementare e le opzioni supportate per ciascuno (attribuendo loro un nome identificativo e – consigliato – una abbreviazione)
- ➔ Definire i possibili errori (di alto livello) da restituire al processo chiamante
- ➔ Disegnare uno schema o un diagramma sintetico del processo *(opzionale)*
- ➔ Acquisire librerie aggiuntive (o scrivere routine di supporto) per gestire il tutto

Analisi dei requisiti

- ➔ Analizzare i requisiti, gli obiettivi e le finalità da esporre tramite CLI
- ➔ Scegliere una modalità di interazione (es. «spot», «con sessione», «mista»)

Obiettivo

Creare una CLI per generare nuovi progetti in Delphi

Modalità

Spot (una tantum) non interattiva

Definizione dei comandi

- ➔ Definire i comandi da implementare
- ➔ Definire le opzioni supportate per ciascun comando
- ➔ Assegnare nomi univoci a comandi e relative opzioni
- ➔ Assegnare abbreviazioni a comandi e relative opzioni (consigliatissimo!)

delphi-cli new

delphi-cli new type:con <name>
delphi-cli new type:vcl <name>
delphi-cli new type:fmx <name>
delphi-cli n t:fmx <name>

delphi-cli add

delphi-cli add unit <name>
delphi-cli add form <name>
delphi-cli a fo <name>
delphi-cli a dm <name>

Livelli di errore

➔ Definire i **possibili errori gestiti** (di alto livello) da restituire al processo chiamante, al batch o al prompt

➔ Usare un **livello di errore progressivo** (per dettaglio incrementale o viceversa)

- `0` → Ok!
- `1` → Errore generico
- `2` → Comando non trovato
- `3` → Opzione non trovata

ecc.

Le sfide da affrontare

«Parsing» dei parametri /1

- Eseguire il «parsing» dei comandi e delle relative opzioni
  - La «tokenization» della riga degli argomenti è solo il primo passo
  - Le combinazioni possono essere molteplici e in ordine sparso
  - I parametri delle opzioni devono essere convertiti nei tipi nativi
  - I valori che contengono spazi sono racchiusi tra delimitatori
  - I parametri possono essere obbligatori o facoltativi
  - La presenza di alcuni comandi può precludere l'uso di altri

«Parsing» dei parametri /2

➔ Interpretare manualmente la riga di comando usando `CmdLine`

- Valido solo nei casi più semplici
- Adatto per formati custom o non-standard

➔ Leggere i singoli parametri usando `ParamStr` e `ParamCount`

- Buon «tokenization» di primo livello
- Limitato al crescere della complessità

➔ Usare la funzione (unit *SysUtils*) **FindCmdLineSwitch**

```
function FindCmdLineSwitch(
  const Switch: string;
  var Value: string;
  IgnoreCase: Boolean = True;
  const SwitchTypes: TCmdLineSwitchTypes =
[clstValueNextParam, clstValueAppended]): Boolean;
```

- **Valida per i formati che usano «switch» e relativi valori assegnati**

➔ **Scrivere la propria logica!**

Istruzioni per l'uso /1

- E' prassi che il programma restituisca informazioni sull'uso
  - Se usato con parametro di help o senza parametri, il tool deve illustrare l'uso
  - E' opportuno che la documentazione segua di pari passo l'evoluzione del tool
  - In caso di errore dell'utente, la segnalazione del tool deve essere chiara
  - In caso di valori enumerativi, il programma deve proporre le opzioni valide

Istruzioni per l'uso /2

- ➔ Utilizzare uno o più file di testo come risorse nel progetto
  - Utili principalmente per intestazioni fisse
  - Facili da manutenere e veloci da modificare
  - Non possono cambiare dinamicamente in base al contesto d'uso dell'applicazione

- ➔ Generazione dinamica di testo (es. classe `TStringBuilder`)
  - ➔ Estremamente flessibile per testi dinamici
  - ➔ Difficile da manutenere al crescere della gamma di comandi e opzioni disponibili
  - ➔ Richiede la scrittura di molto codice

La «Business Logic»?

- ➔ Gli argomenti vanno riconciliati con la Business Logic
  - ➔ Il passaggio dai parametri all'attivazione della logica deve essere agevolato
  - ➔ Occorre evitare il tipico proliferare di costrutti `if`, `case…of` e così via
  - ➔ Deve essere sempre possibile aggiungere facilmente nuovi comandi e opzioni
  - ➔ La stessa logica potrebbe rispondere a diversi comandi/parametri
    (es. in caso di forme abbreviate e alias)
  - ➔ La riga di comando ha una lunghezza limitata

Dal progetto al codice

Usiamo un framework!

➔ VSoft Command Line Parser for Delphi

👉 https://github.com/VSoftTechnologies/VSoft.CommandLineParser

- Si definisce un facile e immediato «Simple Command Line Options Parser»
- Gratuito e open source (disponibile su GitHub)
- Usato nel celebre Test Framework «DUnitX» integrato in Delphi
- Supporta il parsing, la validazione, la documentazione di comandi e opzioni
- Fornisce costrutti (basati su feature moderne del linguaggio Delphi) per implementare i comandi e le opzioni ed estenderle nel tempo

Le opzioni globali

```
option := TOptionsRegistry.RegisterOption<string>(
    'inputfile', 'i', 'The file to be processed',
      procedure(value : string)
      begin
          SampleOptions.InputFile := value;
      end);
option.Required := true;
```

```
option := TOptionsRegistry.RegisterOption<string>(
    'outputfile', 'o', 'The file to be written',
      procedure(value : string)
      begin
          SampleOptions.OutputFile := value;
      end);
option.Required := False;
```

```
option := TOptionsRegistry.RegisterOption<Boolean>(
  'mangle', 'm', 'Mangle the file!',
      procedure(value : Boolean)
      begin
          SampleOptions.MangleFile := value;
      end);
option.HasValue := False;
```

```
option := TOptionsRegistry.RegisterOption<Boolean>(
  'options', '', 'Options file', nil);
option.IsOptionFile := True;
```

Comandi e parsing

```
Cmd := TOptionsRegistry.RegisterCommand(
  'help', 'h', 'get some help', '',
  'commandsample help [command]');

Opt := Cmd.RegisterUnNamedOption<string>(
  'The command you need help for',
  procedure(const value : string)
  begin
    HelpOptions.HelpCommand := value;
  end);
```

```
ParseResult := TOptionsRegistry.Parse;

if ParseResult.HasErrors then
begin
  Writeln(ParseResult.ErrorText);
  ExitCode := 1;
end;
```

```
TOptionsRegistry.PrintUsage('inputfile',
  procedure(const AUsage: string)
  begin
    Writeln(AUsage);
  end);
```

Interagire con la shell

La unit **Winapi.Windows** importa funzioni API anche per la Console

- ➔ Identificare la Console corrente
  - `GetStdHandle(STD_OUTPUT_HANDLE)`
- ➔ Cambiare il titolo della finestra
  - `SetConsoleTitle(PChar(ATitle))`
- ➔ Colorare il testo scritto in output
  - `SetConsoleTextAttribute(BACKGROUND_GREEN or FOREGROUND_BLUE)`
- ➔ … e tantissime altre API!

Documentazione Console API Windows: 👉 https://docs.microsoft.com/en-us/windows/console/

E la «ciccia»?

Aggiungila tu!

JOIN ME
IN DEMO TIME

Dalla CLI… al Web

Web: ambiente sicuro

➔ L'ambiente Web deve essere il più sicuro possibile

- Non deve essere possibile accedere a qualsiasi locazione del disco fisico
- Non deve essere possibile parlare con periferiche esterne come stampanti
- Non è possibile interagire direttamente con il desktop dell'utente
  (a meno di non usare le feature standard che lo consentono, o un plugin)

Non sarebbe bello poter passare parametri dalla pagina Web a un'applicazione Delphi in locale? 🙄

Creiamo un «Web Client»

➔ Si tratta di un’applicazione che riceve parametri da una pagina Web

➔ Possiamo pensarla come una sorta di «App Tethering» per il Web

➔ Estende le feature della pagina lanciando un’applicazione registrata

➔ I parametri a riga di comando vengono passati tramite URL

➔ Per poterla eseguire, è necessaria una «pre-registrazione»

➔ Alla prima esecuzione, occorre dare una conferma al browser

Esempi di URL

mailto:user@domain.com?cc=user2@ext.com

skype:call?username

zoom:joinMeeting?meetingId

yourapp:command?parameters

JOIN ME
IN DEMO TIME

La «ciliegina» sulla torta

Un po’ di «ASCII Art»

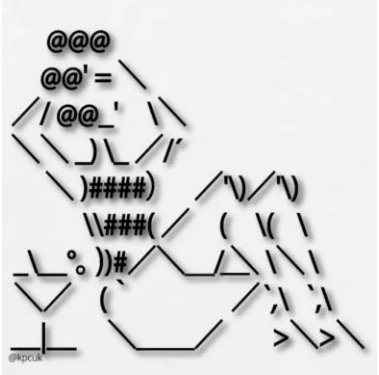
➔ Cerca «ascii art» su Google per trovare numerosi esempi

➔ Diversi siti consentono di generare ASCII da immagini o da stringhe di testo

👉 http://patorjk.com/software/taag/

👉 https://manytools.org/hacker-tools/convert-images-to-ascii-art/

```
DELPHI DAY
```



Creare un setup

➔ Per le esigenze più comuni, **Inno Setup** è perfetto!

👉 http://www.innosetup.org/

➔ Può essere utile per l'installazione delle CLI

- ➔ Copia dei file necessari in una cartella specifica
- ➔ Registrazione del percorso nel PATH! (fondamentale)

➔ Consente la registrazione corretta del Web Client

- ➔ Il setup viene eseguito con privilegi elevati (UAC)

JOIN ME
IN DEMO TIME

Conclusioni

Brainstorming

- ➔ Pensate al software di vostra produzione e come l'aggiunta di una CLI potrebbe migliorarlo
- ➔ CLI non è solo uno strumento per sviluppatori!
- ➔ Può essere utile per rilasciare feature (avanzate) in anteprima

Altre risorse utili

- ➔ VSoft Command Line Parser – Libreria per parsing comandi/opzioni
  - 👉 https://github.com/VSoftTechnologies/VSoft.CommandLineParser
- ➔ Craig Chapman – Calling Linux Commands from Delphi
  - 👉 http://chapmanworld.com/2017/04/06/calling-linux-commands-from-delphi/
- ➔ Rudy Velthuis «Console Units» - Porting di API Console Windows
  - 👉 https://github.com/rvelthuis/Consoles
- ➔ Documentazione Embarcadero Delphi
  - 👉 http://docwiki.embarcadero.com/RADStudio/Rio/en/Console_Applications

Q & A

➔ Se avete dubbi, chiedete pure!

➔ Raccontate le vostre esperienze personali

➔ Proponete le vostre idee!

THANK YOU

wintech italia