var Delphi := Delphi + 1;

SmartPointers + TVirtualInterface

Nulla è come sembra...
...il trucco c'è ma non si vede

wintech italia

speaker: Maurizio Del Magno

embarcadero
reseller partner

Maurizio Del Magno

Developer

Levante software

i-ORM github.com/mauriziodm/iORM

DJSON github.com/mauriziodm/DJSON

mauriziodm@levantesw.it

mauriziodelmagno@gmail.com

facebook.com/maurizio.delmagno

iORM + DJSON (group)

Membro fondatore

eInvoice4D

https://github.com/delphiforce/eInvoice4D

DelphiDay italian conference

PIACENZA EDITION

2019 - XVIII Edizione

var Delphi := Delphi + 1;

SmartPointers + TVirtualInterface

Nulla è come sembra…
…il trucco c'è ma non si vede

wintech italia

speaker: Maurizio Del Magno

embarcadero reseller partner

USE CASE
(LAZY LOAD)

USE CASE - LAZY LOAD

- Usando un O.R.M. c'è il rischio di caricare un intero DB
- Caricamento/creazione di un oggetto "on-demand"
- Differimento nel tempo della creazione/caricamento di una istanza

LAZY LOAD - SPECIFICHE I-ORM

- Interfacce
- Nessuna modifica alla dichiarazione delle classi
- Nessuna modifica al codice delle classi
- Nessuna modifica al codice che utilizza le istanze delle classi

…in pratica come se il Lazy-Load non ci fosse!

COME DIRE CHE…

…vogliamo un puntatore valido per un oggetto che ancora non esiste…

…che poi si crei da solo, “al volo”, nel momento stesso in cui accediamo per la prima volta ad una sua proprietà o metodo…

…e che questo “meccanismo” sia completamente invisibile!!!

DEMO TIME

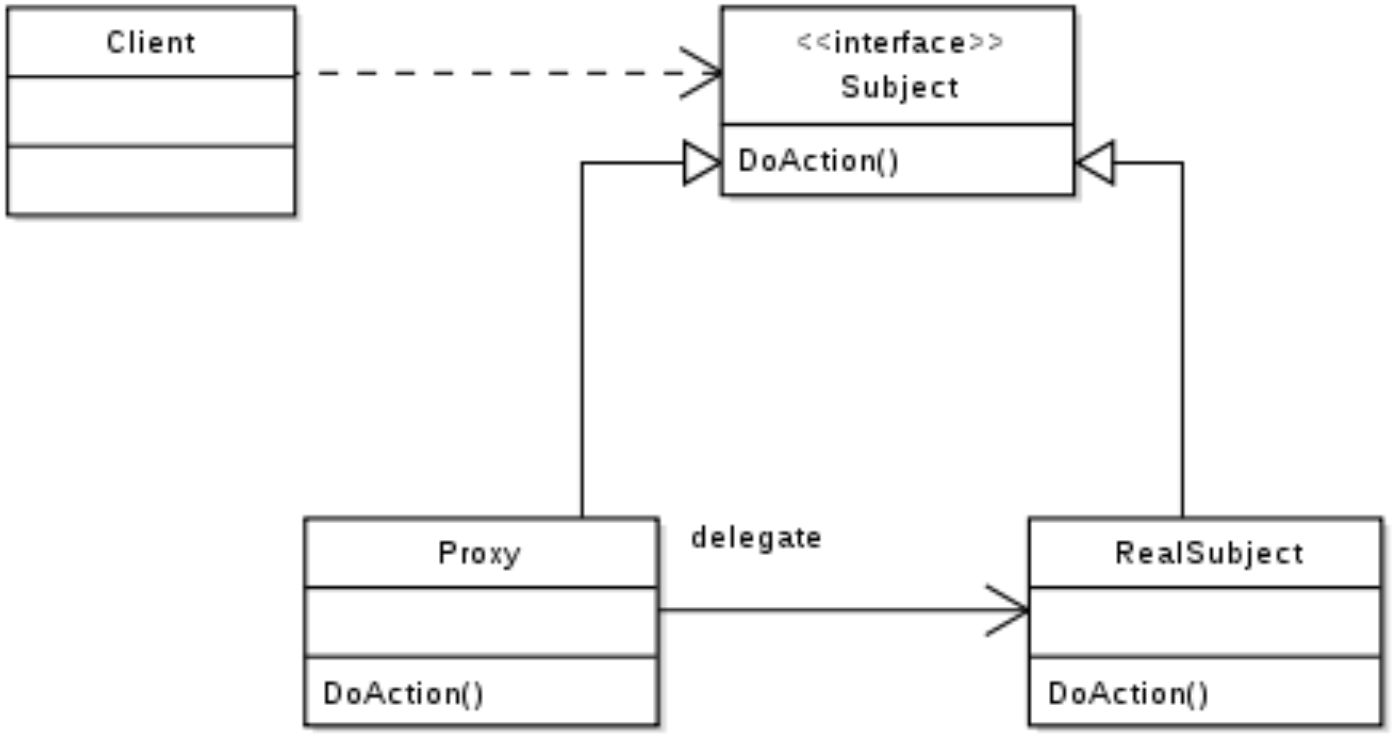
SIAMO TUTTI CONTRIBUTOR DI I-ORM

Proxy Pattern

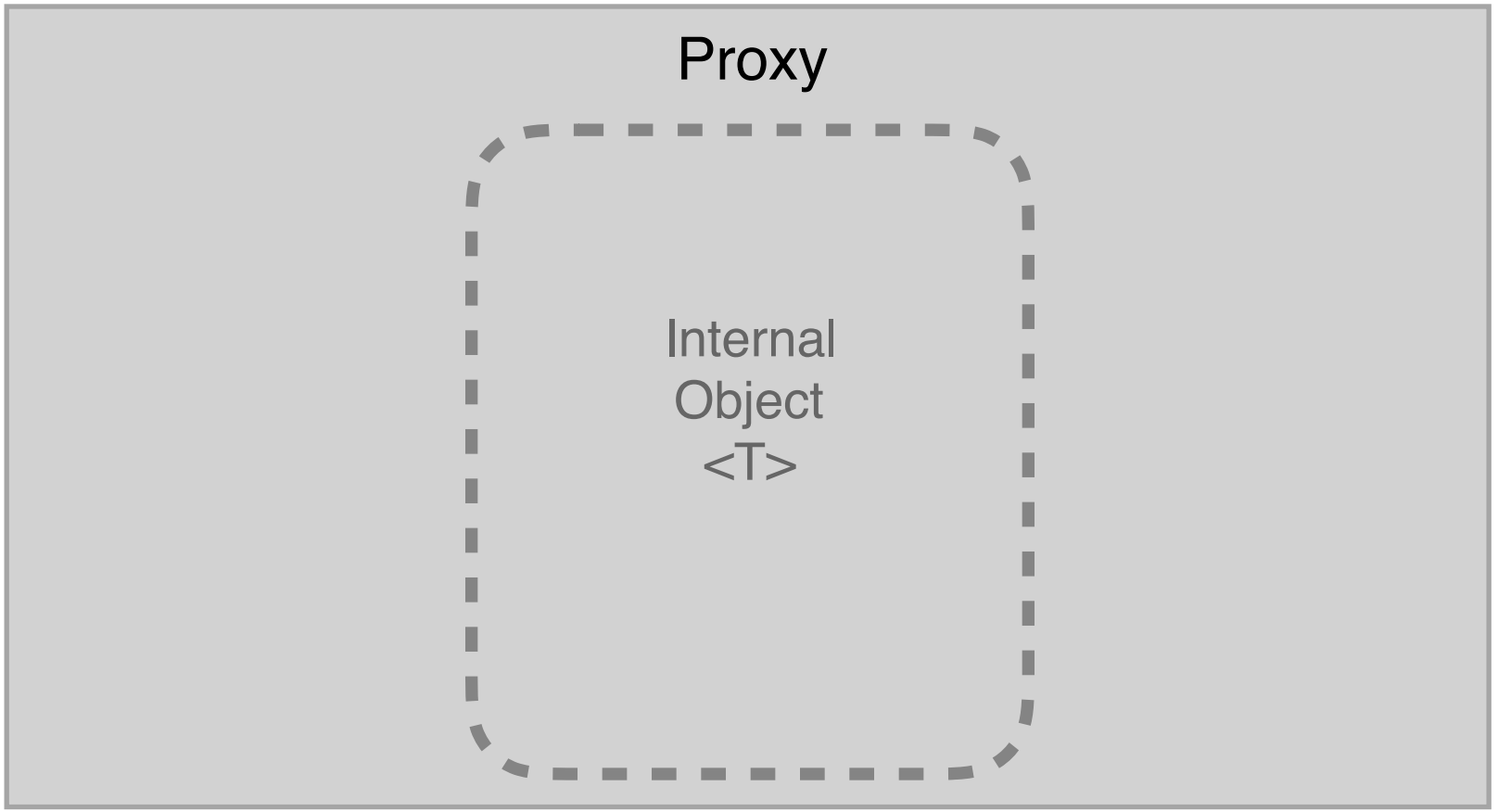
PROXY PATTERN

- E' un design pattern
- Controllare, limitare, regolamentare l'accesso ad un'altra classe
- Tipi di proxy:
  - Protection proxy
  - Remote proxy
  - Virtual proxy
    - regolamentare/ottimizzare l'accesso ad una classe/risorsa dispendiosa da creare (cache)
    - differire nel tempo la creazione di una classe/risorsa dispendiosa da creare

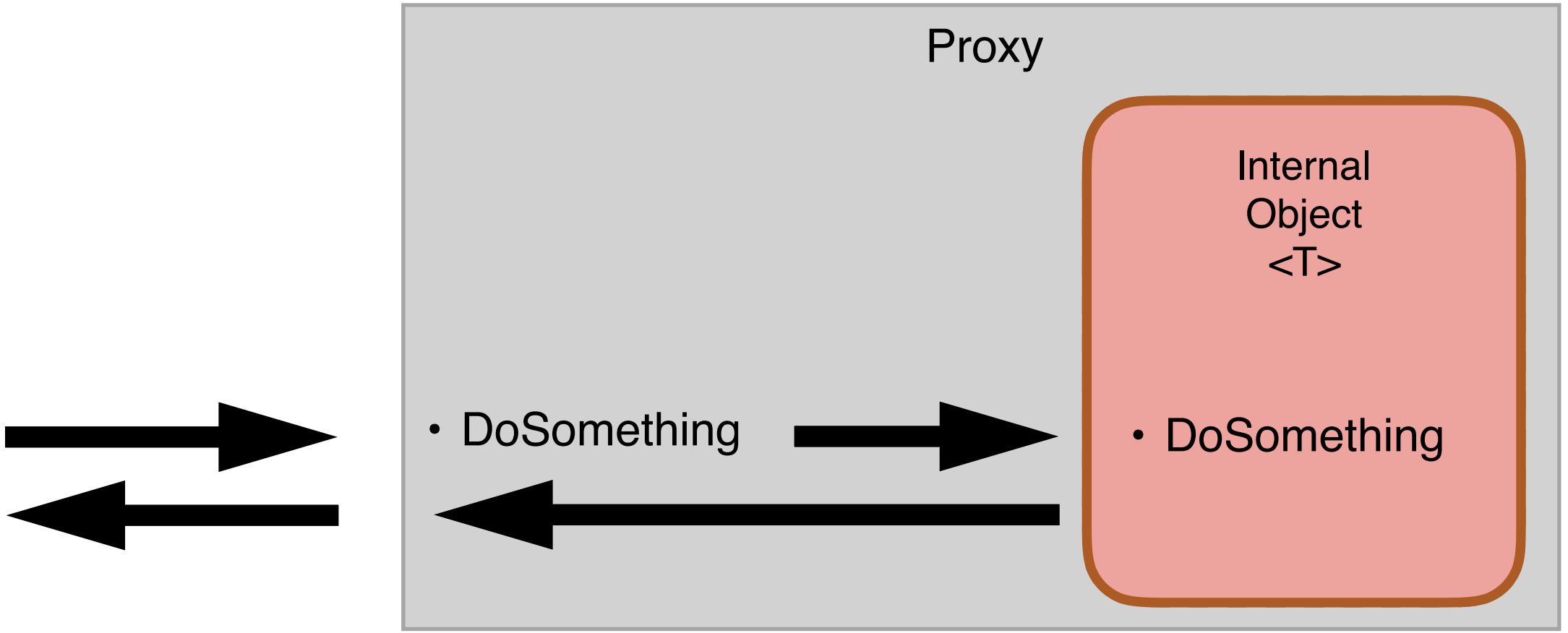
PROXY PATTERN



PROXY PATTERN

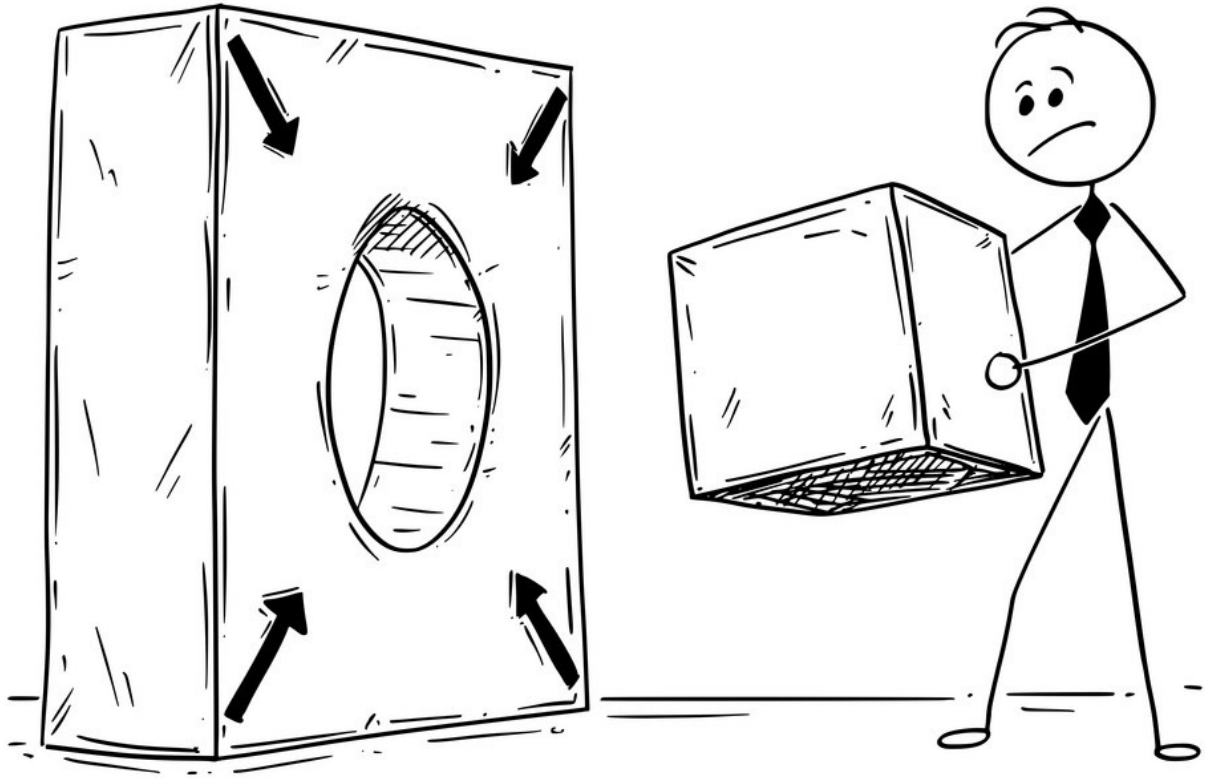


PROXY PATTERN

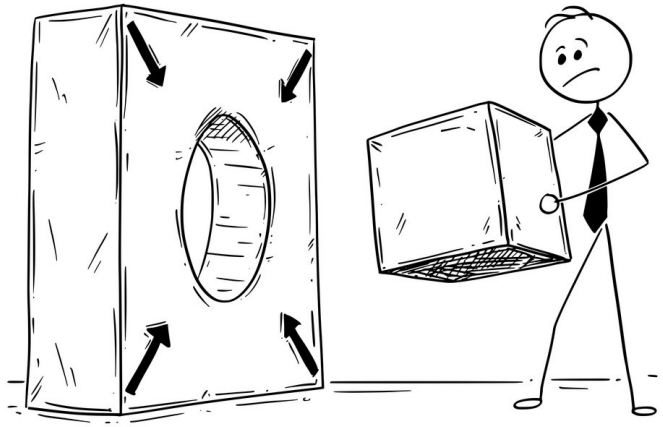


DEMO TIME…

PROBLEMI DA RISOLVERE



Problemi da risolvere



1) TypeCast

```
(Order.Customer.GetValue as ICustomer).Name;
```

2) GetValue

```
(Order.Customer.GetValue as ICustomer).Name;
```

3) Variable/Property type (IProxy…)

```
FCustomer: ILazyLoader;
…
property Customer: ILazyLoader…;
```

GENERICS
(GENERICS BASED PROXY)

GENERICS
(GENERICS BASED PROXY)

- "Generics" or "Generic Types" or "Parameterized Types"
- Un parametro che invece di passare un valore rappresenta un tipo
- Un modo per scrivere codice senza specificare il tipo sul quale opera…
- … quindi è "generico" rispetto al tipo che tratterà
- Il tipo sarà poi "passato" al codice solo al momento del suo utilizzo
- Disaccoppia una classe/interfaccia/metodo dal tipo sul quale opererà

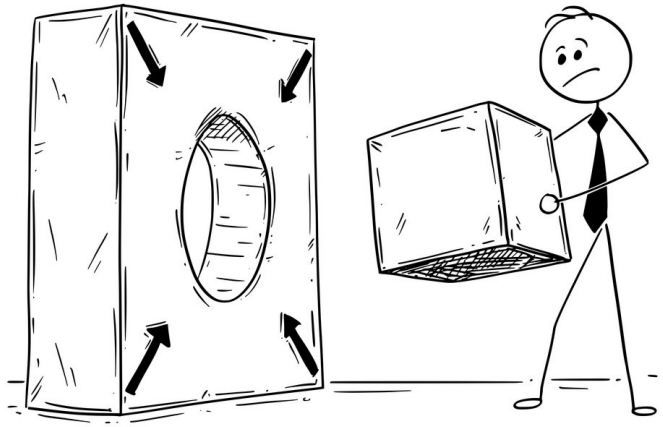
GENERICS
(GENERICS BASED PROXY)

- Benefici:
  - Riutilizzo del codice
  - Eliminazione del typecasting
  - Codice più leggibile
  - Type Safety (compile time & runtime)

GENERICS
(GENERICS BASED PROXY)

DEMO TIME…

Problemi da risolvere



1) TypeCast

(Order.Customer.GetValue as ICustomer).Name;
Order.Customer.GetValue.Name;

2) GetValue

(Order.Customer.GetValue as ICustomer).Name;

3) Variable/Property type (IProxy…)

FCustomer: ILazyLoader;
…
property Customer: ILazyLoader…;

GENERICS
(CONSIDERAZIONI…)

- Il tipo generico deve essere conosciuto in fase di compilazione
- Il codice della classe generica non usa caratteristiche specifiche del tipo generico
- In alcuni casi la classe generica potrebbe usare un insieme ristretto di caratteristiche comuni a tutti i sottotipi del generico (constraint)
  - TMyClass<T:class, constructor> = class…
  - TMyClass<T:TDocumento> = class…
  - TMyClass<T:IInterface> = class…
  - TMyClass<T:IDocumento> = class…

SMART POINTERS

(ANONIMOUS METHODS AS INTERFACE, THE DELPHI WAY…)

ANONIMOUS METHODS

- Sono dei metodi (procedure o funzioni) che non hanno un nome
- Possono essere assegnati ad una variabile
- Possono essere passati come parametro
- Possono essere ritornati come risultato di una funzione
- Utili nella programmazione multi-thread e con la PPL
- Il codice che riceve e invoca l'esecuzione di un anonimous method non conosce nulla di lui, ne cosa effettivamente faccia e nemmeno il contesto nel quale è stato dichiarato e nel quale opera

```
LCalcTotals: TProc;
…
LCalcTotals := procedure
begin
  // Some code
end;
…
Order.Rows.Add(TOrderRow.Create(LCalcTotals));
…
ActualRow.CalcTotals;
```

ANONIMOUS METHODS

```
unit FormOrder;

interface

uses

. . .

type

  TOrderForm = class(Form)
    QryOrder: TFDQuery;
  private
    procedure ButtonCustomerSelClick(Sender: TObject);
  public
  end;

implementation

uses

. . .

procedure TOrderForm.ButtonCustomerSelClick(Sender: TObject);
begin
  Application.CreateForm(TCustSelectionForm, CustSelectionForm);

  CustSelectionForm.CustSelProc := procedure (CustomerID: Integer)
  begin
    QryOrder.FieldByName('ID').Value := CustomerID;
    . . .
  end;

  CustSelectionForm.Show; // ShowModal non disponibile su Android
end;
```

```
unit FormCustSelection;

interface

uses

. . .

type
 TCustSelectionForm = class(Form)
  private
    FCustSelProc: TProc<integer>;
    procedure GridCustomerBblClick(Sender: TObject);
  public
    property CustSelProc: TProc<Integer> read FCustSelProc write FCustSelProc;
  end;

implementation

uses

. . .

procedure TCustSelectionForm.GridCustomerBblClick(Sender: TObject);
begin
  FCustSelProc(  QryCustomers.FieldByName('ID').AsInteger  );
  Self.Close;
end;

. . .
```

DEMO TIME…

SMART POINTERS
(ANONIMOUS METHODS AS INTERFACE, THE DELPHI WAY…)

- Simula un puntatore aggiungendogli funzionalità
- In pratica è un puntatore che “wrappa” un secondo puntatore
- Gestione automatica della memoria (deallocazione, reference counting, weak pointer…)

- Ma non ci sono le interfacce ???
- E cosa centrano gli anonimous methods ???

SMART POINTERS
(ANONIMOUS METHODS AS INTERFACE, THE DELPHI WAY…)

- Ma non ci sono le interfacce ???
- E cosa centrano gli anonimous methods ???

centrano perché un anonimous method in realtà è…

… un'interfaccia!!!

e hanno anche un'altra caratteristica che fa proprio al caso nostro…

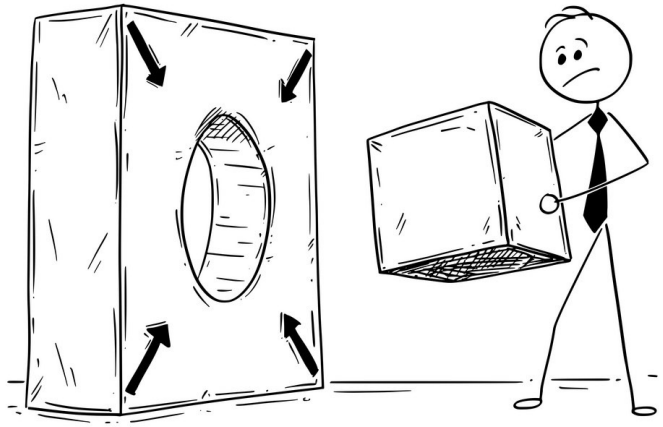
… un metodo "Invoke" che viene invocato automaticamente.

SMART POINTERS

(ANONIMOUS METHODS AS INTERFACE, THE DELPHI WAY…)

DEMO TIME…

Problemi da risolvere



✓ 1) TypeCast

(Order.Customer.GetValue as ICustomer).Name;
Order.Customer.GetValue.Name;

✓ 2) GetValue

(Order.Customer.GetValue as ICustomer).Name;
Order.Customer.Name;

3) Variable/Property type (IProxy…)

FCustomer: ILazyLoader;
…
property Customer: ILazyLoader…;

TVIRTUALINTERFACE

(A MAGIC CLASS…)

Generalmente per usare un'interfaccia scriviamo una o più classi che la implementano ognuna in un suo modo specifico.

Questa implementazione è statica a runtime.

```
ISomething = interface
  procedure DoSomething;
end;


TSomething = class(TInterfacedObject, ISomething)
  procedure DoSomething;
end;


TSomethingElse = class(TInterfacedObject, ISomething)
  procedure DoSomething;
end;


LMyVar: ISomething;

LMyVar.DoSomething;
```

… e se si potesse implementare un'interfaccia senza una classe specifica?

… e implementare qualsiasi interfaccia con un unico "pezzo" di codice?

… e decidere quale interfaccia fargli implementare a runtime?

… dinamicamente!!!

TVIRTUALINTERFACE

(A MAGIC CLASS…)

TVirtualInterface

(a magic class…)

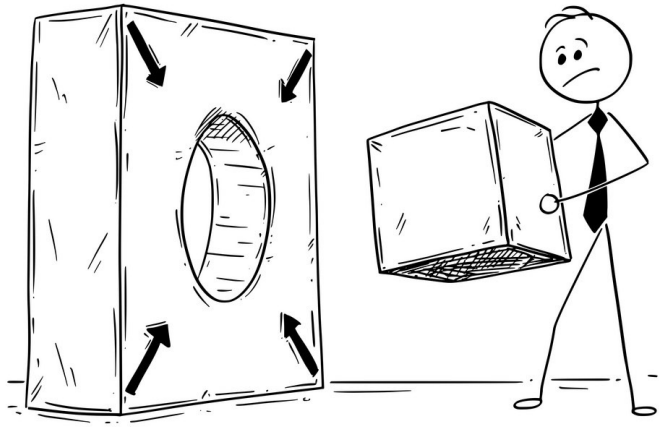
- Può assumere le sembianze di qualsiasi interfaccia dinamicamente…
- … a runtime
- Una chiamata a un metodo causa l'esecuzione dell'evento OnInvoke
- Parametri:
  - Method: TRttiMethod;
  - Args: TArray<TValue>;
  - out Result: TValue;

TVirtualInterface

(a magic class…)

DEMO TIME…

Problemi da risolvere



1) TypeCast

(Order.Customer.GetValue as ICustomer).Name;
Order.Customer.GetValue.Name;

2) GetValue

(Order.Customer.GetValue as ICustomer).Name;
Order.Customer.Name;

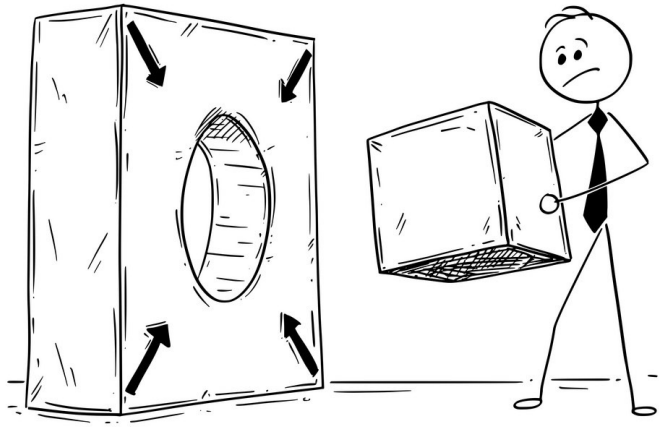
3) Variable/Property type (IProxy…)

FCustomer: ILazyLoader;
…
property Customer: ILazyLoader…;

FCustomer: ICustomer;
…
property Customer: ICustomer…;

PROBLEMI DA RISOLVERE

- [x] 1) TypeCast
- [x] 2) GetValue
- [x] 3) Variable/Property type (IProxy…)



RIFLESSIONE
(PARAMETRI)

- Generics: passare un tipo come parametro
- AnonimousMethods: passare un blocco di codice come parametro
- TVirtualInterface: passare una interfaccia da implementare come parametro
- TRttiMethod: passare la rappresentazione di un metodo come parametro

DOMANDE?

Maurizio Del Magno

Developer

Levante software

i-ORM github.com/mauriziodm/iORM

DJSON github.com/mauriziodm/DJSON

mauriziodm@levantesw.it

mauriziodelmagno@gmail.com

facebook.com/maurizio.delmagno

iORM + DJSON (group)

Membro fondatore

eInvoice4D

https://github.com/delphiforce/eInvoice4D

DelphiDay italian conference

PIACENZA EDITION

2019 - XVIII Edizione

Grazie!!!

wintech italia

speaker: Maurizio Del Magno

embarcadero reseller partner