VZORCI NAČRTOVANJA (DESIGN PATTERNS)

Primož Gabrijelčič

O meni

- Primož Gabrijelčič
- http://primoz.gabrijelcic.org
- programmer, MVP, writer, blogger, consultant, speaker
- Blog *http://thedelphigeek.com*
- Twitter *@thedelphigeek*
- Skype *gabr42*
- LinkedIn *gabr42*
- GitHub *gabr42*
- SO *gabr*

The Delphi Geek

random ramblings on Delphi, programming, Delphi programming, and all the rest

Sunday, October 16, 2022

"Design Patterns" workshop in Ljubljana

For the last Delphi meeting in Slovenia this year we have organized a small workshop about Design Patterns. As usual, it is intended for Slovenian programmers and will be given in the Slovenian language.

Read more »

Posted by gabr42 at 17:15 No comments:
Labels: presentations

Tuesday, May 17, 2022

We'll meet again (finally)!

After two long-distance years we are finally moving back to normality, starting with a Slovenian RAD Studio meeting next Wednesday in Ljubljana.

Po dveh letih virtualnih konferenc vas končno spet vabimo na srečanje v živo! Za izgovor za druženje si bomo ogledali novosti v RAD Studiih iz zadnjih dveh let (10.4, 10.4.1, 10.4.2, 11, 11.1), predvsem pa bomo dogodek izkoristili za klepet ob hrani in pijači in ponovno spoznavanje.

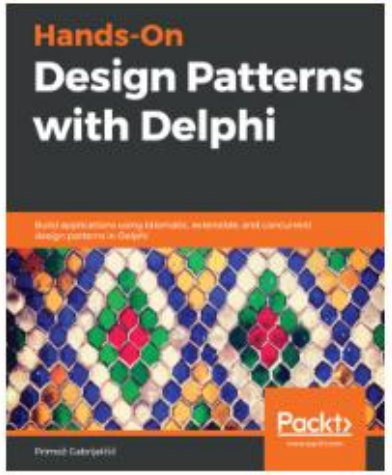
Pridružite se nam 25. maja ob 9h! (klikni za več podatkov in prijavo)

Posted by gabr42 at 12:19 1 comment:
Labels: C++Builder, Delphi, presentations, RAD Studio

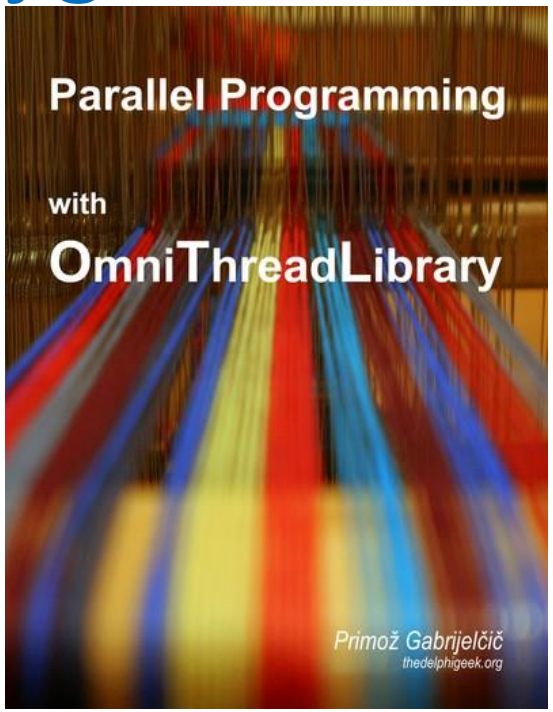
embarcadero MVP

Pages

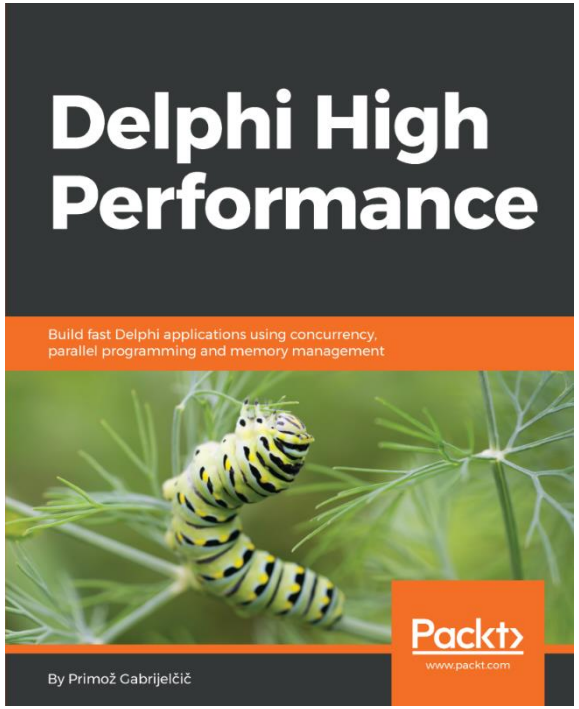
Presentations



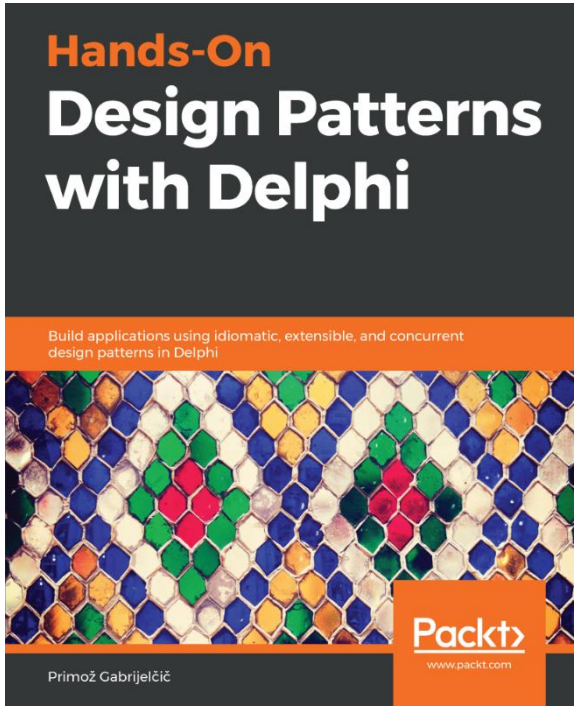
Knjige



http://tiny.cc/
pg-ppotl



http://tiny.cc/
pg-dhp



http://tiny.cc/
pg-dpd

Vzorci načrtovanja

Vzorci načrtovanja

- Vzorec = predloga postopka
- Vzorec = splošno sprejet žargon
- Vzorec ≠ recept

- Arhitekturni vzorci > vzorci načrtovanja > idiomi
- Vzorci načrtovanja ≠ principi načrtovanja (design principles; SOLID, DRY…)

Kritika

- “Klasični” vzorci =
  “Design Patterns: Elements of Reusable Object-Oriented Software”
  - 1994!
  - Specifični za objetno-orientirano programiranje
  - Dokaj vezani na C++
  - Enonitni
  - Dandanes ne predstavljajo vedno najboljših rešitev
- Vzorci načrtovanja niso namenjeni načrtovanju programske arhitekture!
  - Uporabni so za reševanje konkretnih problemov
- Vzorci načrtovanja so **orodje**, ne **cilj**!

Idiomi v Delphiju

- Kreiranje/uničevanje objektov
- Assign in AssignTo
- [Atributi]
- Iteriranje z for..in
- Helperji
- Akcije
- …

```
obj := TMyObject.Create;
try
  …
finally
  FreeAndNil(obj);
end
```

Arhitekturni vzorci

- Model-View-Controller, …
- Oblikovanje, ki temelji na domeni (Domain driven design)
- Multinivojska arhitektura
- …

Principi načrtovanja

- SOLID — Single responsibility, Open-closed, Liskov substitution, Interface segregation, Dependency inversion
- DRY — Don’t repeat yourself
- KISS — Keep it simple stupid
- YAGNI — You ain’t gonna need it
- SoC — Separation of concerns
- NIH/PFE — Not invented here / Proudly found elsewhere

Kategorije vzorcev načrtovanja

- Vzorci kreiranja: *delegiranje*
  - Izdelava objektov in skupin objektov
- Strukturni vzorci: *združevanje*
  - Načini sestavljanja objektov
- Vzorci obnašanja: *komunikacija*
  - Interakcija med objekti
- Vzorci sočasnosti: *sodelovanje*
  - Hkratno delovanje brez stopanja po prstih

Vzorci kreiranja (Creational patterns)

Vzorci kreiranja

- Abstract factory
- Builder
- Dependency injection
- Factory method
- Lazy initialization
- Multiton
- Object pool
- Prototype pattern
- Resource acquisition is initialization (RAII)
- Singleton

Niso v knjigi

Obdelani na delavnici

Singleton

- Obstajati sme samo en primerek razreda
- Ne uporabljaj (pravih) singletonov!
  - Problemi s testiranjem kode (unit testing)
  - Problemi s konfiguriranjem
- Boljši pristopi
  - Globalna tovarna (factory)
  - Globalna spremenljivka
  - Dependency Injection

Dependency injection

- Odvisne komponente dobi razred “od zunaj”
- IOC (inversion of control), vsebniki (containers)
  - “Dependency injection in Delphi, Nick Hodges”
  - http://codingindelphi.com/
- Constructor injection
- Property injection
- Parameter injection

Lazy initialization

- Podatke inicializiramo šele, ko jih potrebujemo
- Enostavno v enonitnem programu

```
if not assigned(lazyObject) then
  lazyObject := TLazyObject.Create;

Use(lazyObject);
```

- Malce bolj zapleteno v večnitnem programu
  - Zaklepanje
  - Mikro-zaklepanje
- Spring.Lazy<T>

Object pool

- Zbirka objektov, ki čakajo na uporabo
- Kadar je kreiranje/inicializacija objekta „drag“ (počasen) proces
- Database connection pool
- HTTP connection pool
- Thread pool

Factory method

- Metoda, ki izdela objekt (vmesnik, drugo metodo)

- Factory method = TFunc, …

Abstract factory

- Tovarna tovarn
  - Primer: Tovarna tovarn, ki generirajo elemente GUI
  - Generira tovarno VCL elementov ali tovarno FMX elemetov
- Dandanes podobne probleme raje rešimo z vmesniki in vbrizgavanjem odvisnosti (DI)

Prototype

- Kloniranje objektov/recordov
- Plitvo kloniranje / globoko kloniranje
  - Prvo je enostavno (lahko kar enostavno prirejanje recordov)
  - Drugo je zapleteno in običajno potrebuje prilagojeno rešitev
- Assign in AssignTo

Builder

- Kreiranje objektov skozi kodo
- XML Builder, SQL Query builder …

Strukturni vzorci (Structural patterns)

Strukturni vzorci

- Adapter
- Bridge pattern
- Composite
- Decorator
- Extension object
- Facade
- Flyweight
- Front controller
- Marker
- Module
- Proxy
- Twin

Niso v knjigi

Obdelani na delavnici

Composite

- Skupek razredov za delo s hierarhičnimi podatki
  - Vsi razredi implementirajo iste operacije

- Dandanes za reševanje takšnih problemov uporabljamo vmesnike

Flyweight

- Namesto kopij objektov uporabimo kazalce na deljeni objekt
- Normalizacija baz podatkov
- Stringi

Marker interface

- Vmesnik, ki opisuje, kaj razred zna (meta lastnosti)
- Dandanes raje uporabljamo atribute

Bridge

- Ločimo abstraktno definicijo od implementacije

Adapter

- Prilagajanje podatkov, protokolov …
- Namenjen reševanju problemov, ne načrtovanju
  - Raje uporabi Bridge

Proxy

- Posreduje med objektom in subjektom operacije
- Protection proxy
- Remoting proxy
- Lazy initialization proxy
- Mocking proxy
- Logging proxy
- Locking/serialization proxy

Decorator

- Doda funkcionalnost obstoječemu vmesniku
- Helperji

Facade

- Vmesnik, ki poenostavi interakcijo s kompleksnim sistemom

Bridge/Adapter/Proxy/Decorator/Facade

- *Bridge*, *adapter*, *proxy* in *decorator* ovijejo **en** razred. *Facade* ovije **več** razredov.
- *Bridge* in *adapter* zagotavljata **drugačen** vmesnik. *Proxy* zagotavlja **enak** vmesnik. *Decorator* zagotavlja **izboljšan** vmesnik. *Facade* zagotavlja **poenostavljen** vmesnik.
- Ko uporabimo *bridge*, **definiramo** tako abstraktni vmesnik kakor tudi implementacijo. Pri *adapterju* implementacija obstaja že **vnaprej**.

Vzorci obnašanja (Behavioral patterns)

Vzorci obnašanja

- Blackboard
- Chain of responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Null object
- Observer (Publish/Subscribe)
- Servant
- Specification
- State
- Strategy
- Template method
- Visitor

Niso v knjigi

Obdelani na delavnici

Null object

- Objekti, ki implementirajo vmesnik, a ne naredijo nič
- Zamenjava za ‘if assigned(…)’
- Uporabni pri testiranju
- *Null* objekt ≠ *nullable* objekt

Template method

- Postopek, ki mu manjkajo nekateri (pomembni) deli (virtual; abstract)
- Izdelamo jih v izpeljanih razredih (override)
- Ali pa uporabimo vmesnike in injiciranje (DI)

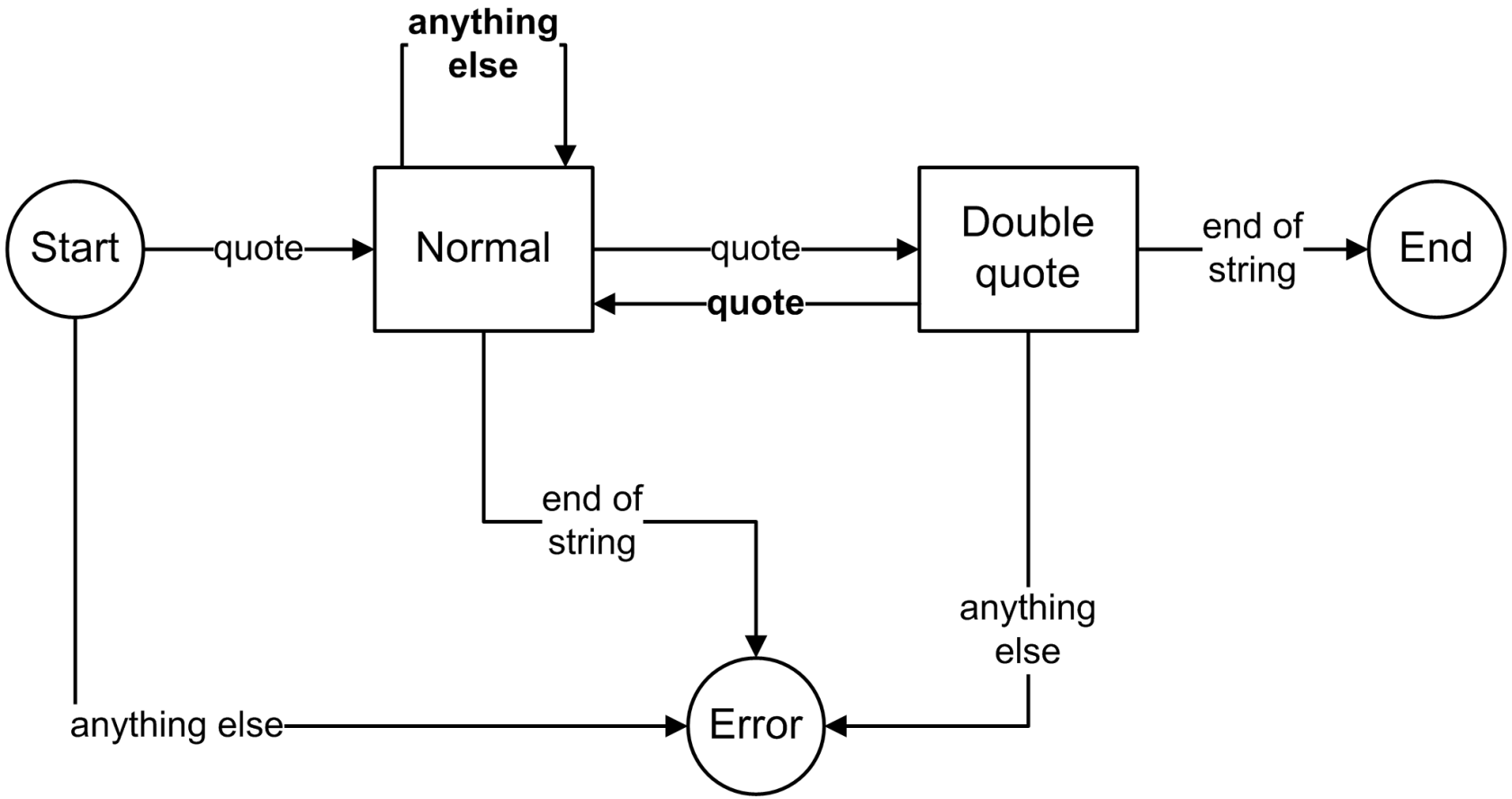
Command

- Zavijanje UI akcij v objekte
- *Client, Command, Receiver, Invoker*
- Undo/Redo
- TAction

State

- Opisuje, kako zamenjati kup zapletenih if..else z objektno strukturo

Razpoznavanje "citiranih" nizov



Iterator

- For..in

- Robustni iteratorji
- Ničelni iteratorji

Visitor

- Sprehodi se po hierarhični strukturi podatkov in izvede (uporabniško) kodo na vsakem objektu

Observer

- Ne kličite nas, bomo mi poklicali
- Publish-Subscribe

- Neposredno izvajanje ali sporočanje
- Opcijska granularnosti
  - Običajno pomeni, da je objekt preveč kompleksen (SRP!)
- Live Bindings
  - TComponent.Observers
- Spring4D Multicast events
  - Event<T>

Memento

- Objekt, ki shrani stanje drugega objekta
- Ločitev stanja od objekta
- TBookmark

Vzorci sočasnosti (Concurrency patterns)

Vzorci sočasnosti

- Active object
- Binding properties
- Blockchain pattern
- Compute kernel
- Double-checked locking
- Event-based asynchronous
- Future
- Guarded suspension
- Join
- Lock
- Lock striping
- Messaging
- Monitor object
- Optimistic locking
- Pipeline (Staged processing)
- Reactor
- Read-write lock
- Scheduler
- Thread pool
- Thread-specific storage

Niso v knjigi

Omenjeni na delavnici

Locking

- Usklajuje dostop do deljenih sredstev
  - Nepotrebno, če **vsi** samo **berejo** iz deljenih sredstev
- Lahko upočasni program
- Lahko prinese probleme (deadlocking)
- Kritične sekcije
- TMonitor

Lock striping

- Zaklepanje na bolj podrobne nivoju
- Enobitne ključavnice

Double-checked locking

- Hitrejše izvajanje kode, ki ni skoraj nikoli uporabljena
- Izdelava deljenih objektov v večnitnem programu
  - Lazy initialization
- *Test/Lock/Test again*

Optimistic locking

- Naredi svoje, pa šele potem poglej, če se je deljeni objekt medtem spremenil
- Hitra izdelava deljenega objekta
  - Pod pogojem, da ni nič narobe, če objekt naredimo dvakrat

Readers-writer lock

- Sočasno delo bralcev, ekskluzivni dostop za pisanje
- Pogosto branje, občasno pisanje
- TMREWSync = TMultiReadExclusiveWriteSynchronizer
  - Počasen!
  - Reentrant, upgradeable, write-biased
- TLightweightMREW
  - Hiter!
  - Non-reentrant, not upgradeable, read-biased

Thread pool

- *Object pool* za niti
- Hitrejši zagon opravil v ozadju
- TTask.Run (namesto .Start)
- IOmniTaskControl.Schedule (namesto .Run)

Messaging

- Nadomestek za deljenje podatkov – prenos sporočil
- Windows messaging
- Queue in Synchronize
- Prilagojene rešitve

Future

- Vzporedno izvajanje funkcionalnosti s poenostavljeno sinhronizacijo
- Odpravi potrebo po zaklepanju
- TTask.Future<T>
  - + TThread.Queue

Pipeline

- „Tekoči trak“ za sočasno izvajanje operacij, ki jih lahko razdelimo na stopnje
- Odpravi potrebo bo zaklepanju

Q&A