



Wstrzykiwanie Zależności w Praktyce

Bogdan Polak - OEC Connection



github.com/bogdanpolak



linkedin.com/in/bogdanpolak

O mnie

OEConnection, Team Leader, **Kraków/Warszawa**, C# & React & Azure & SCRUM

Craneware, Senior Software Engineer, **Edinburgh**, Delphi & C#

BSC Polska, Trainer/Consultant/Community Activist, **Warszawa**, Delphi, C++,
TestComplete, StarTeam,

Microgeo, Entrepreneur / Engineer, **Warszawa**, Delphi, Pascal, 2D Graphics

Przygotowanie

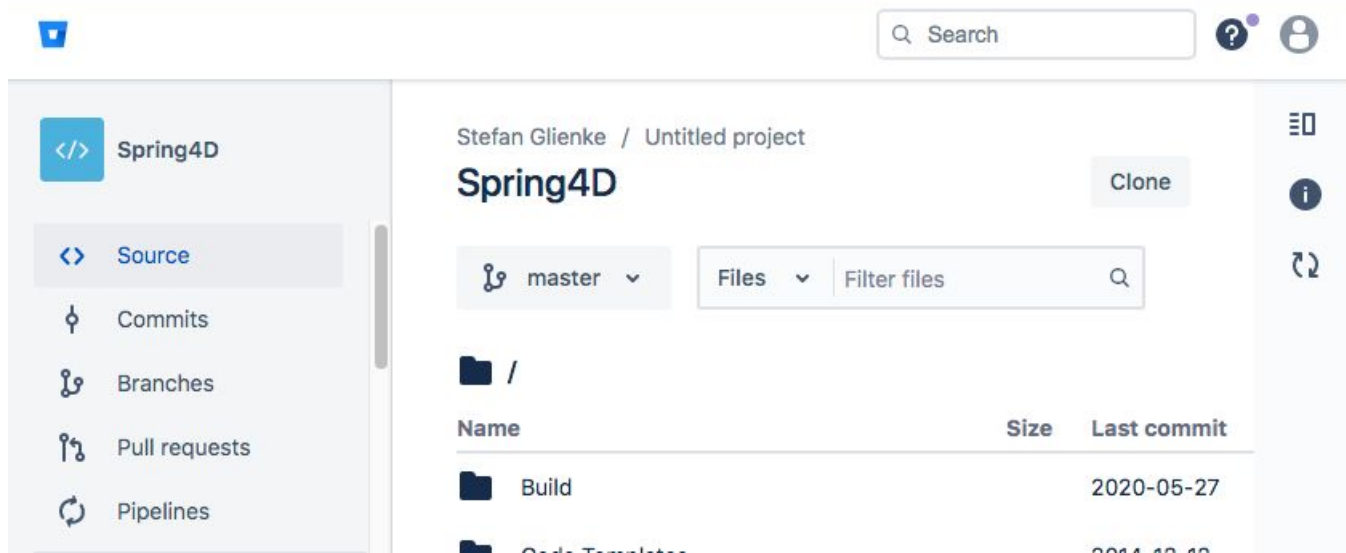


Wymagane pakiety w ramach szkolenia

1. Delphi 10+
2. FireDAC - wystarczy wersja Pro
3. Spring4D
 - a. aktualny branch master <https://bitbucket.org/sglienke/spring4d>

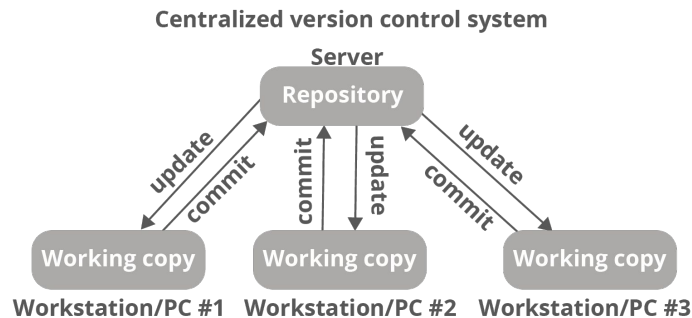
Instalacja Spring4D

<https://bitbucket.org/sglienke/spring4d>

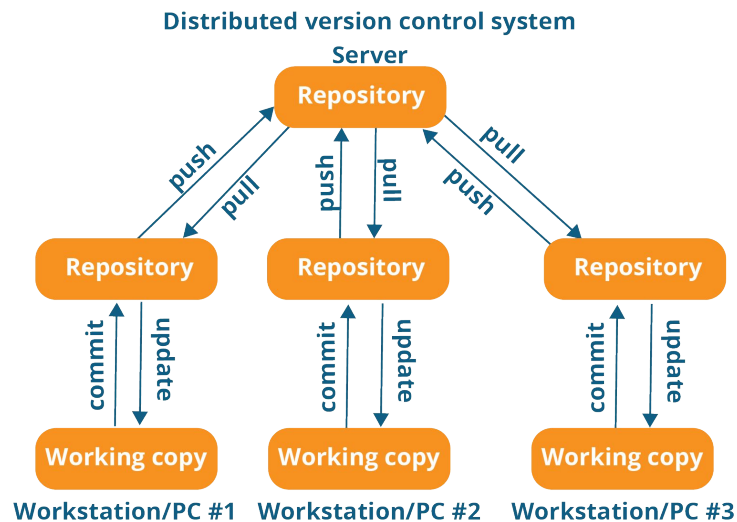




Dlaczego git?



Rozproszony





Autor

Linus

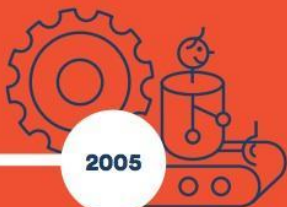


Torvalds

2022
17

YEARS OF GIT

2005



2011



2015



2007



2013



Jak się pracuje z Git-em?

```
mathias at mathBook in ~/dotfiles on master [+]
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

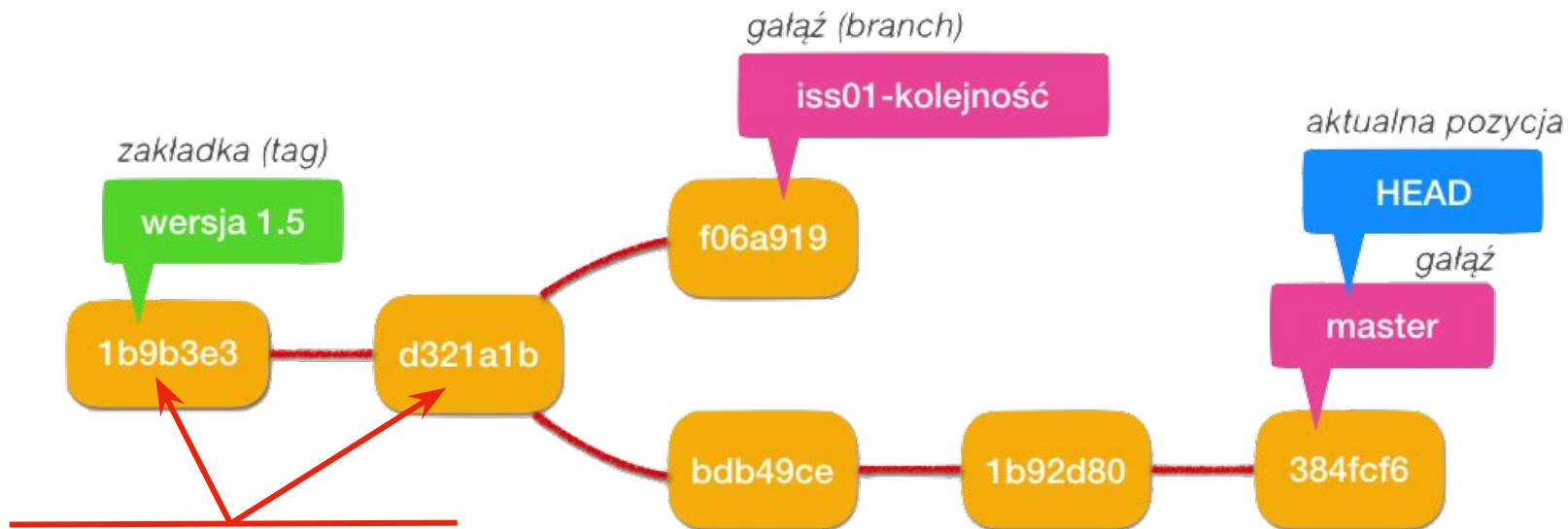
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   .bash_prompt
        modified:   .osx
        deleted:    init/Mathias.terminal
        new file:   init/Solarized Dark xterm-256color.terminal
        renamed:    init/Mathias.itermcolors -> init/Solarized Dark.itermcolors

mathias at mathBook in ~/dotfiles on master [+]
$
```

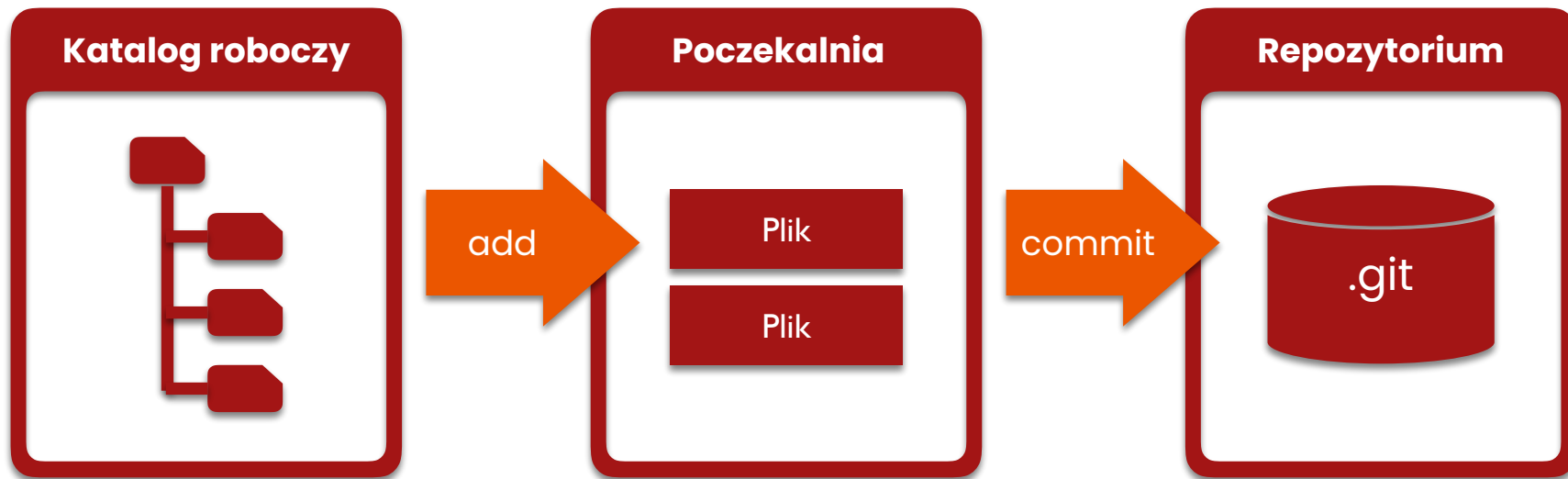
```
react-redux-blog — -bash -
rupa ~ ➤ cd react-redux-blog/
rupa ~ ➤ react-redux-blog ➤ dev ➤ git stash
No local changes to save
rupa ~ ➤ react-redux-blog ➤ dev ➤ touch bla.txt
rupa ~ ➤ react-redux-blog ➤ dev ... 1 ➤ git stash
No local changes to save
rupa ~ ➤ react-redux-blog ➤ dev ... 1 ➤ git add .
rupa ~ ➤ react-redux-blog ➤ dev ● 1 ➤ git stash
Saved working directory and index state WIP on dev: 96054a4 test
rupa ~ ➤ react-redux-blog ➤ dev ➤ 1 ➤ git checkout —detach dev
HEAD is now at 96054a4... test
rupa ~ ➤ react-redux-blog ➤ 96054a4 ➤ 1 ➤
```

Słownik Git-a



rewizja = commit = "komit"

Tworzenie rewizji





Tworzenie rewizji - commitów

```
git add src/OrdersEdit*  
git commit
```

*Dodanie do poczekalni **nowych i zmienionych** plików pasujących do wzorca. Stworzenie nowej rewizji*

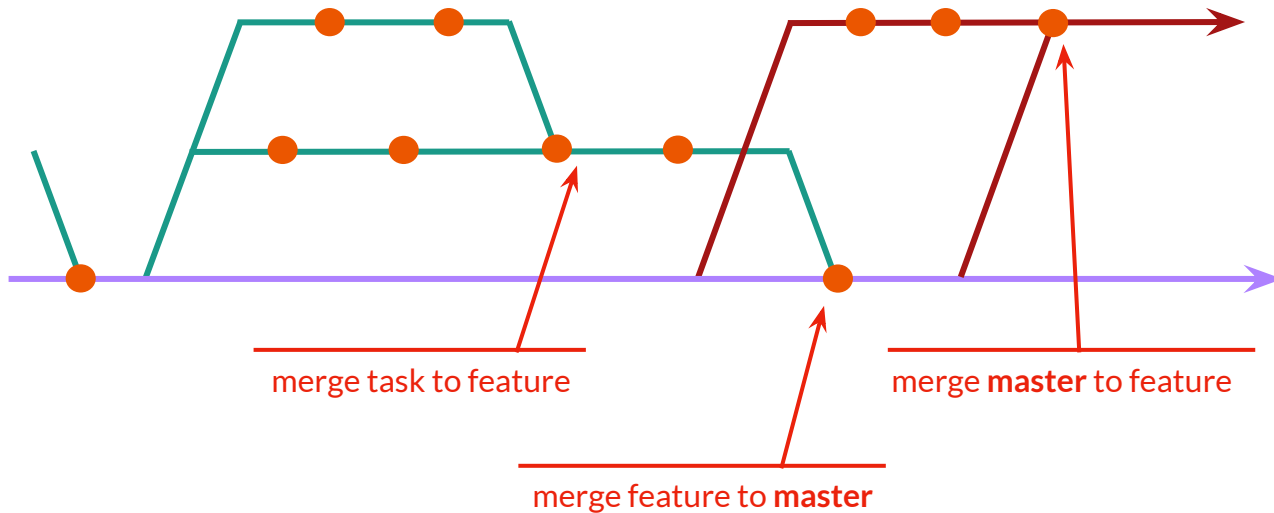
```
git add .  
git commit
```

*Dodanie do poczekalni wszystkich **nowych i zmienionych** plików. Stworzenie nowej rewizji*

```
git commit -am "initial"
```

*Dodanie do poczekalni wszystkich **zmienionych** plików oraz stworzenie nowej rewizji*

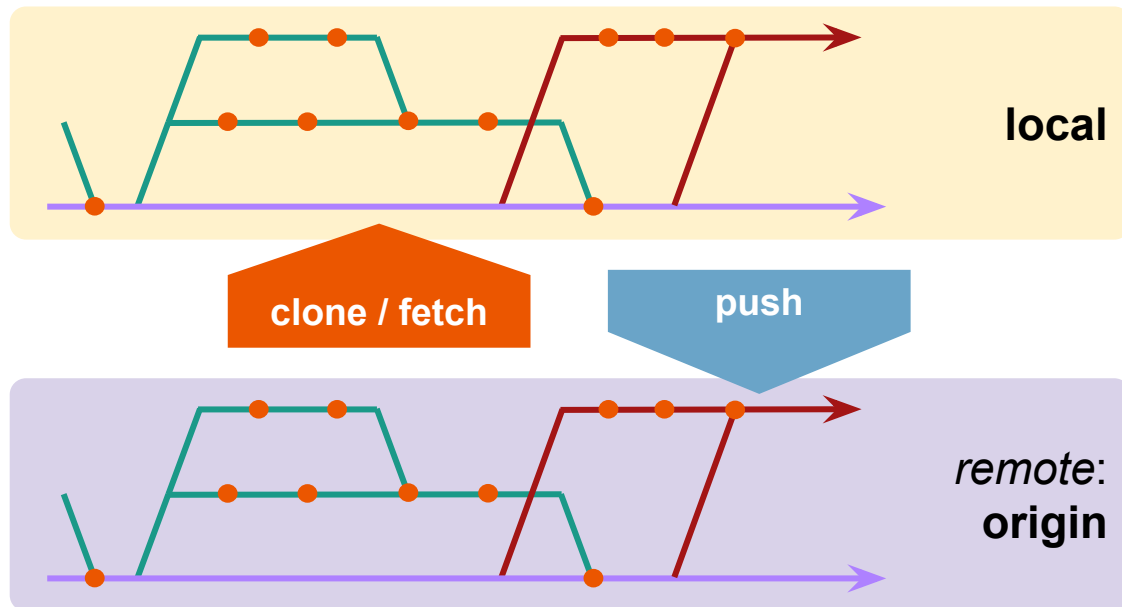
Zawsze rób zmiany w gałęziach



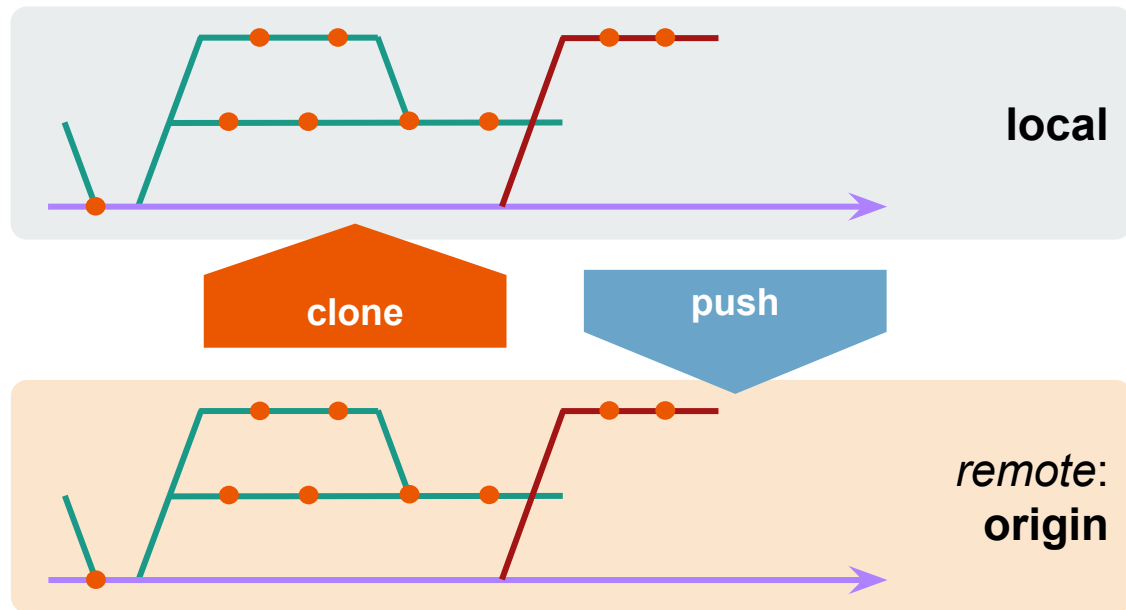
Repozytorium lokalne i zdalne

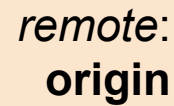
Zmiany są wykonywane
w lokalnym
repozytorium,

Następnie są one
synchronizowane ze
zdalnym repo, które jest
współdzielone z innymi
członkami zespołu

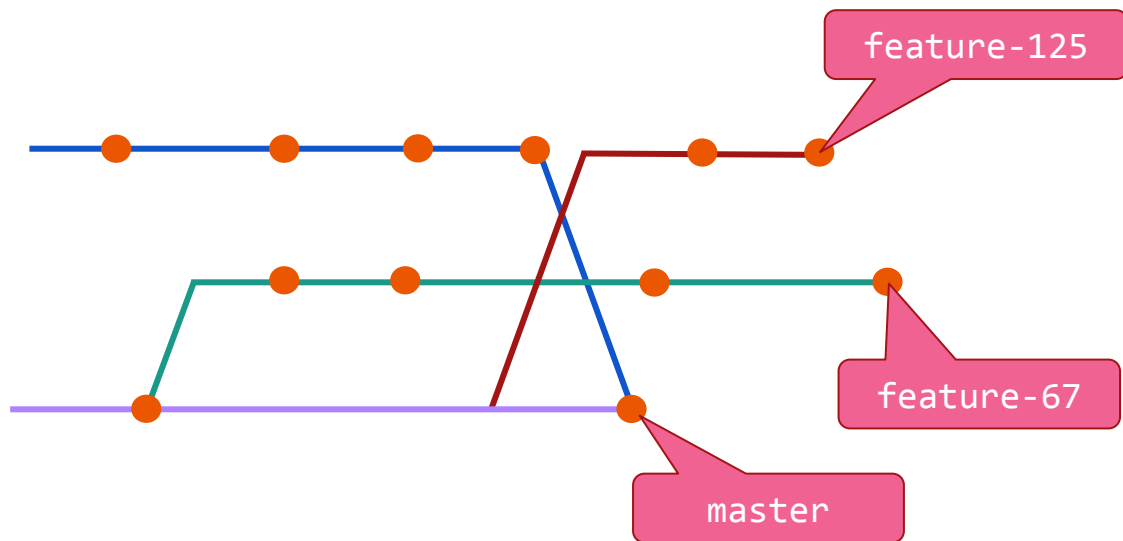


Workflow: local <-> origin





Gałęzie / "brancze" / branches



```
git branch feature-67
```

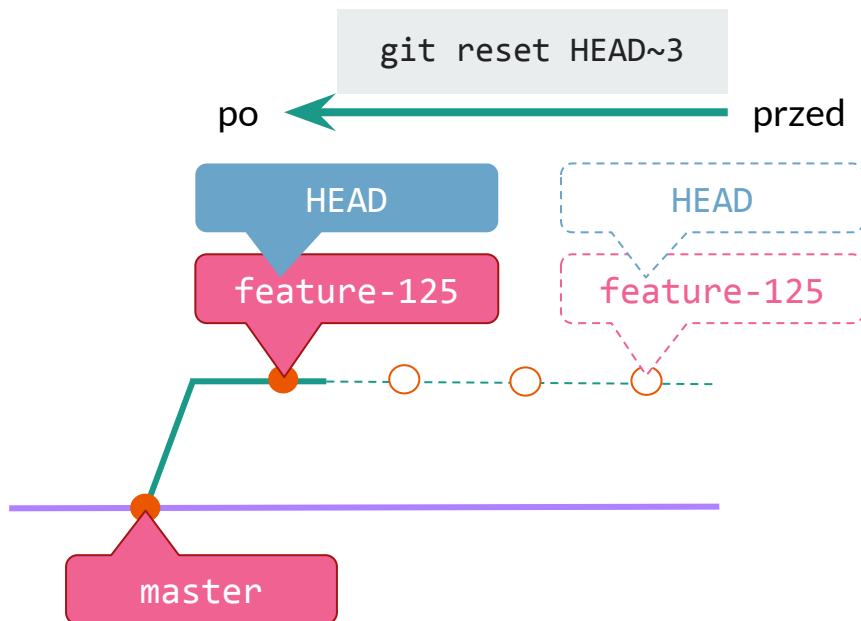
```
git checkout feature-125
```

```
git checkout -b bug-102
```

```
git branch -d feature-67
```

```
git branch -D feature-67
```

Komenda git reset - poruszanie się po gałęzi



Inne przykłady

```
git reset master
```

```
git reset --hard
```

```
git reset --hard origin/feature-125
```

```
git checkout -f
```



.gitignore

Lista plików ignorowanych

Pliki i foldery pasujące do wzorca nie są brane pod uwagę przy wyborze zmian

np. wpis: `__history/` spowoduje zignorowanie wszystkich plików z folderu “__history”

The logo for gitignore.io, featuring a small blue diamond icon followed by the text "gitignore.io" in a bold, blue, sans-serif font.

Create useful .gitignore files for your project

Delphi x

Create

<https://www.toptal.com/developers/gitignore/>



Ćwiczenia z git-a

Command-line History Graph

Sprawdź historię repozytorium

```
> git log --graph --oneline --all
```

Zadanie 1

Zrób poprawki w nowej gałęzi i dołącz ją do master

```
master git checkout -b bugfix/25
bugfix/25 ...
+ bugfix/25 git add . & git commit -m "Fixed calculation"
bugfix/25 git checkout master
master git merge bugfix/25 --no-ff
```

Zadanie 2

Zapisz zmiany z wybranych plików w repozytorium

```
git add src/Form.OrdersEditor.*  
git commit -m "Added 🐛 Orders Editor form"  
  
git log
```


Zadanie 3

Zmień opis rewizji / commita

```
git commit --amend -m "Added Orders Editor form"
```

```
git log
```

Zadanie 4

Cofnij mój ostatni commit i podziel go na dwa

git reset in mixed mode

```
git reset HEAD~1  
git add "src/Unit2.*" "src/*.dpr"  
git commit -am "Added new form"  
git add .  
git commit -am "Added Dataset and logic to DataModule1"  
  
git push --force
```

Zadanie 5

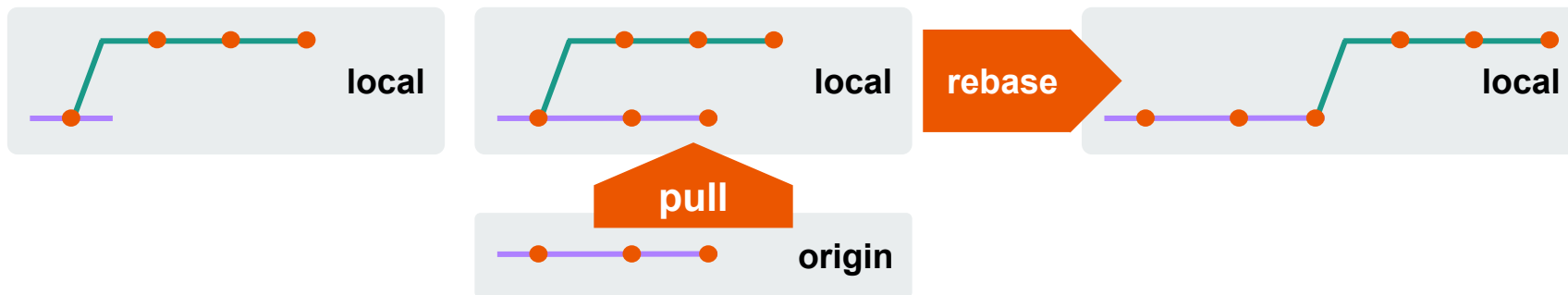
Tymczasowo zapisz prace w toku, aby sprawdzić inną gałąź

		<code>git stash</code>
		<code>git checkout feature-41</code>
		
		<code>git checkout feature-67</code>
		<code>git stash apply</code>
		<code>\$</code>

Zadanie 6

Zsynchronizuj master z origin i uwzględnij najnowszego master-a

```
git checkout master  
git pull master origin  
git checkout bugfix-71  
git rebase master  
git push --force
```



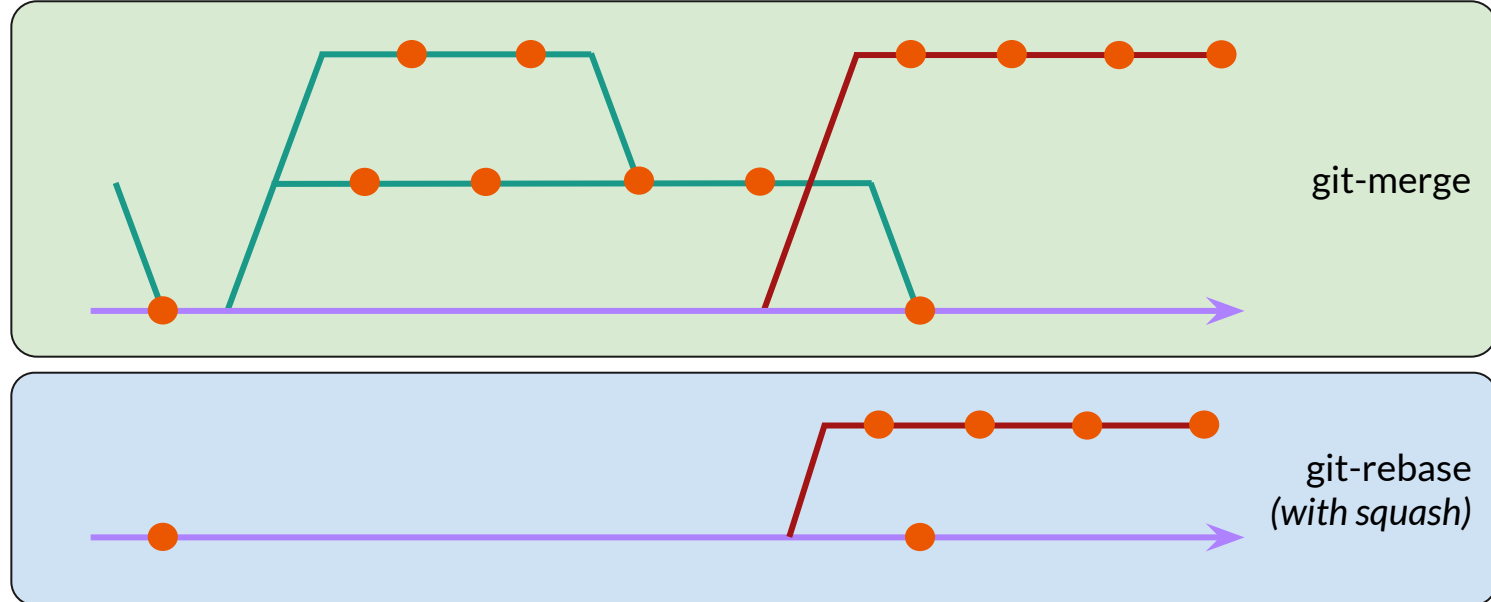
Zadanie 7

Cofnij zmiany w jednym z plików do wersji początkowej

```
> git commit -a  
> git commit -a  
  
> git checkout master -- src/DataModule1.*  
> git commit -am "Reverted changes in DataModule1"
```

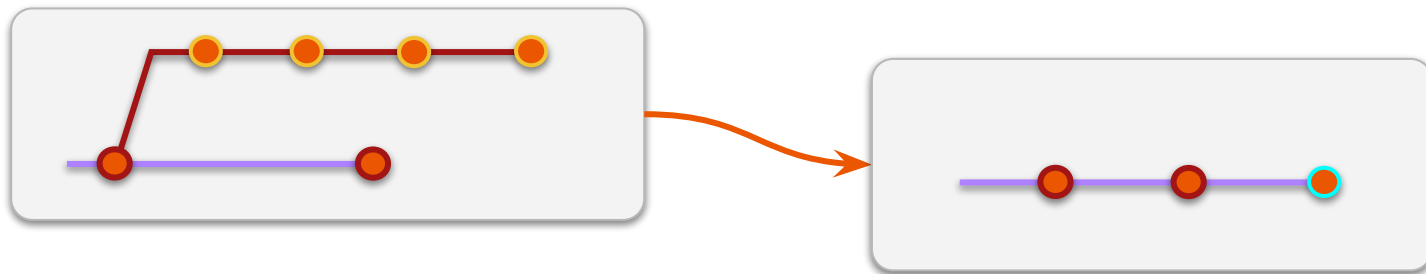
<https://mfcallahan.blog/2022/06/10/different-git-merge-types-azure-dev-ops/>

Rozjazdy czy liniowa historia ze zgniataaniem



Graficznie: git rebase & squash

<https://tinyurl.com/git-rebase-gif-animation>



Historia liniowa - git squash

Trochę więcej pracy aby dostał ładną i przejrzystą historię

aktualizacja mastera z origin

*zgniecenie (ang. squash) zmian z
gałęzi roboczej do jednej rewizji*

```
> git checkout master  
> git pull  
> git merge --squash feature/151-customer-filter
```

Rozstrzygnięcie konfliktów


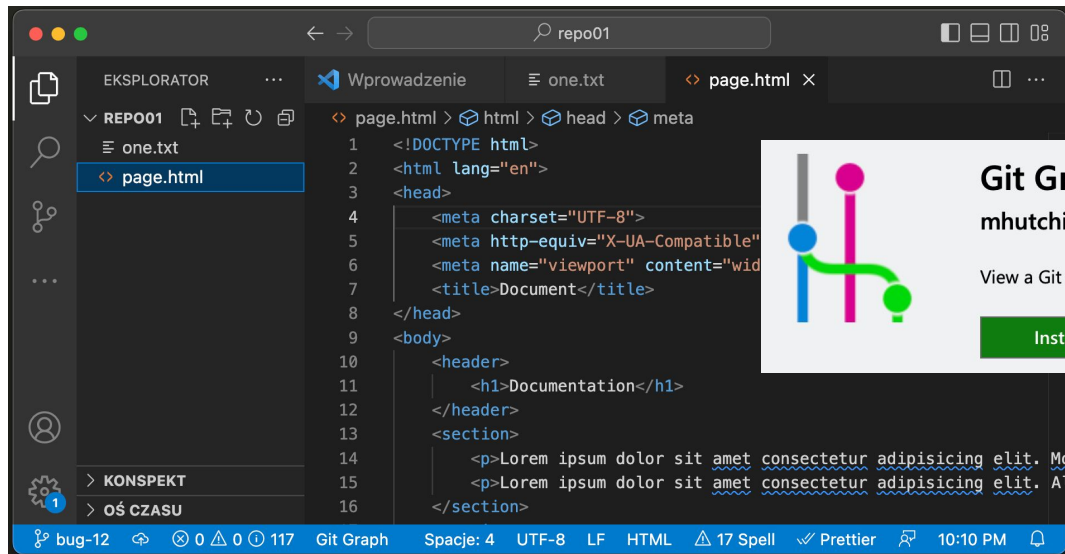


Praca domowa



1. **Usuń** środkową rewizję z mojej lokalnej gałęzi.
2. Przenieś rewizje, które zapisałem przypadkowo do master, **do nowej gałęzi**.

Polecane narzędzia do ćwiczeń z git



Git Graph

mhutchie | 4,176,525 installs | ★★★★★ (445) | Free

View a Git Graph of your repository, and perform Git actions from the graph.

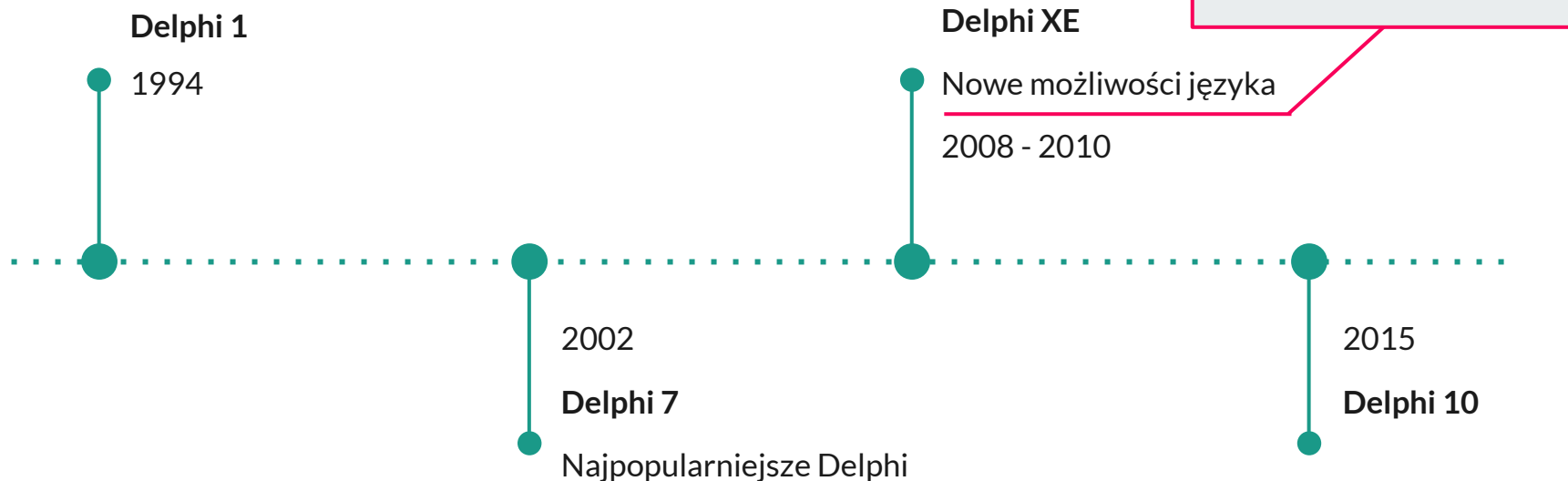
[Install](#) [Trouble Installing?](#)

Nowoczesny kod Delphi





Historia Delphi





Typy Generyczne

Wykorzystywanie gotowych klas

Tworzenie własnych typów



Delphi Generics

How Generics Works?

The terms **generics** or **generic types** describe the set of things in a platform that can be parameterized by type. The term **generics** can refer to either generic types or generic methods, i.e., generic procedures and generic functions

Generics are a set of abstraction tools that permit the decoupling of an algorithm (such as a procedure or function) or a data structure (such as a class, interface, or record) from one or more particular types that the algorithm or data structure uses.

Typy Generyczne

```
type
  TReaderThread = class(TThread)
  private
    fQueue: TThreadedQueue<byte>;
  protected
    procedure Execute; override;
  public
    constructor Create(aQueue: TThreadedQueue<byte>);
  end;

procedure TReaderThread.Execute;
begin
  while not Terminated do
  begin
    TThread.Sleep(700);
    fQueue.PushItem(Random(256));
  end;
end;
```

<https://docwiki.embarcadero.com/Libraries/Alexandria/en/System.Generics.Collections.TThreadedQueue>

Typy Generyczne

```
type
    TData = record
        str: string;
        int: Integer;
        bool: Boolean;
        flt: Double;
    end;

var
    ctx: TSuperRttiContext;
    data: TData;
    obj: ISuperObject;
begin
    ctx := TSuperRttiContext.Create;
    try
        data := ctx.AsType<TData>(
            SO('{str: "foo", int: 123, bool: true, flt: 1.23}'))
    );
    obj := ctx.AsJson<TData>(data);
finally
    ctx.Free;
end;
end;
```

<https://github.com/pult/SuperObject.Delphi>

Deklarowanie typu generycznego

```
type
  TBase<T:Class> = class
    protected
      fObj: T;
    public
      constructor Create;
      destructor Destroy; override;
    end;

  constructor TBase<T>.Create;
begin
  fObj.Create;
end;

  destructor TBase<T>.Destroy;
begin
  fObj.Free;
  inherited;
end;
```



Typy generyczne - Dokumentacja

[https://docwiki.embarcadero.com/RA/DStudio/Alexandria/en/Overview of Generics](https://docwiki.embarcadero.com/RA/DStudio/Alexandria/en/Overview_of_Generics)

Generics Index

[Go Up to Delphi Language Guide Index](#)

Presents an overview of generics, a terminology list, a summary of grammar changes for generics, and details about declaring and using parameterized types, specifying constraints on generics, and using overloads.

Topics

- [Overview of Generics](#)
- [Terminology for Generics](#)
- [Declaring Generics](#)
- [Overloads and Type Compatibility in Generics](#)
- [Constraints in Generics](#)
- [Class Variable in Generics](#)

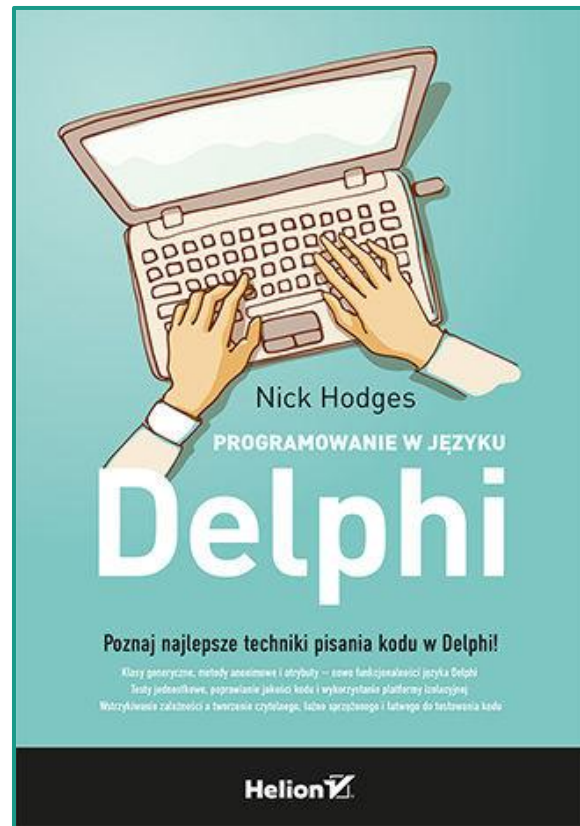
Typy generyczne - Książka

<https://helion.pl/ksiazki/programowanie-w-jezyku-delphi-nick-hodges.prodel.htm>

Nick Hodges

“Programowanie w języku Delphi”

Przetłumaczone na jęz. polski przez Helion





Metody anonimowe

Metody anonimowe

```
TThread.CreateAnonymousThread(  
    procedure  
    var  
        result: string;  
    begin  
        DoSomeWork;  
        result := ReadResult;  
        TThread.Synchronize(nil,  
            procedure  
            begin  
                Button1.Text := result;  
            end);  
    end).Start;
```

<https://docwiki.embarcadero.com/Libraries/Alexandria/en/System.Classes.TThread.CreateAnonymousThread>

Metody anonimowe

```
begin
  dataset := CreateDataSet(fOwner, [
    [1, 'Edinburgh', 5.1, EncodeDate(2018, 05, 28)],
    [2, 'Glassgow', 3.4, EncodeDate(2015, 09, 13)],
    [3, 'Cracow', 5.8, EncodeDate(2019, 01, 01)],
    [4, 'Prague', 4.7, EncodeDate(2013, 06, 21)]]);
  trips := '';
  dataset.ForEachRow(
    procedure
    begin
      trips := IfThen(trips = '', '', trips + ', ' +
        FormatDateTime('yyyy-mm',
          dataset.FieldName('visited').AsDateTime);
    end);
  // trips: '2018-05, 2015-09, 2019-01, 2013-06'
end;
```



Spring4D

Kolekcje

Inne: TEnum, ILazy<>, Nullable<>, TLazy<>, Tuple<>

Spring4D Collections

List

uses

```
Spring.Collections;
```

var

```
employees: IList<TEmployee>;
```

begin

```
employees := TCollections.CreateList<TEmployee>(True);
```

```
employees.Add(TEmployee.Create('Jan Kowalski', 9800));
```

```
employees.Add(TEmployee.Create('Tomasz Jankowski', 6300));
```

```
employees.Add(TEmployee.Create('Ewa Tomaszewska', 8500));
```

```
employees.Sort(
```

```
    function(const e1, e2: TEmployee): integer
```

```
    begin
```

```
        Result := e1.Salary - e2.Salary;
```

```
    end);
```

```
for emp in employees do
```

```
    writeln(emp.FirstName, ' ', emp.LastName, ': ', emp.Salary);
```

```
end.
```

<https://bitbucket.org/sglienke/spring4d/src/master/Samples/SpringDemos/Demo.Collections/uSortCustomers.pas>

Spring4D Collections

Dictionary

```
var
  loaded: IDictionary<string, IShared<TWeather>>;

function GetLoacation(const location: string): IShared<TWeather>;
begin
  if not loaded.ContainsKey(location) then
    begin
      loaded.Add(location, GetWeather(location));
    end;
  Result := loaded[location];
end;

begin
  loaded := TCollections.CreateDictionary<string, IShared<TWeather>>;
  loaded.Add('poland/warsaw', GetWeather('poland/warsaw'));
  writeln(loaded['poland/warsaw'].ToString);
  writeln(GetLoacation('germany/berlin').ToString);
end.
```

Spring4D General

TEnum

```
uses
    Spring;
type
    TNumberEnum = (One, Two, Three, Four, Five, Six, Seven,
        Eight, Nine, Ten);
var
    names: TStringDynArray;
    values: TIntegerDynArray;
    number: TNumberEnum;
    s: string;
begin
    names := TEnum.GetNames<TNumberEnum>;
    values := TEnum.GetValues<TNumberEnum>;
    number := TEnum.Parse<TNumberEnum>('Seven');
    s := TEnum.GetName<TNumberEnum>(number);
end.
```

<https://bitbucket.org/sglienke/spring4d/src/master/Samples/SpringDemos/Demo.General/uEnumDemo.pas>

Spring4D General

Nullable<T>

```
procedure ShowDate(const date: Nullable<TDateTime>);
begin
    if date.HasValue then
        Writeln('date = ', DateToStr(date.Value))
    else
        Writeln('date is null')
end;

var
    date: Nullable<TDateTime>;
begin
    ShowDate(date);
    date := EncodeDate(2023, 01, 15);
    ShowDate(date);
    date := nil;
    // date.Value - should throw EInvalidOperationException
end.
```

Spring4D
General

TLazy<T>

```
uses
    Spring;

function TDataModule1.BalanceFor(id: integer): ILazy<Currency>;
var
    dm: TDataModule1;
begin
    dm := Self;
    balance := TLazy<Currency>.Create(
        function: Currency
        begin
            Result := dm.GetBalanceOfCurrentAccountId();
        end);
end;
```

Spring4D
General

Tuple<>

```
var
    pair1: Tuple<Integer, string>;
    pair2: Tuple<Integer, string>;
    employee: Tuple<Integer, string, Currency, TDateTime>;
begin
    pair1 := Tuple<Integer, string>.Create(43, 'Jan Kowalski');
    pair2 := [41, 'Adam'];
    writeln(pair1.Value1, ' ', pair1.Value2);
    writeln(Format('pair1 %s pair2',
        [IfThen(pair1.Equals(pair2), '=', '≠')]));
    employee := [21, 'Paweł Branko', 8500, EncodeDate(2017,06,01)];
    employee.Unpack(empId, empName, empSalary, empHireDate);
end.
```



Class Helpers



Class helpers

Rozszerzenie funkcjonalności
klas bibliotek podstawowych

- RTL
- VCL
- FireDAC, AnyDAC, ...

```
type
  FormHelper = class helper for TForm
  private
    procedure SetOnFormReady(aOnReady: TProc);
  public
    property OnFormReady: TProc write SetOnReady;
  end;

procedure FormHelper.SetOnFormReady(aOnReady: TProc);
begin
  // ...
end;
```

Record Helper for TBytes

Użycie

```
var
  bytes: TBytes;
  idx: integer;
  memoryStream: TMemoryStream;
  command := TUploadImageCommand;
begin
  bytes.Size := 1000;
  for idx := 0 to bytes.Size-1 do
    bytes[idx] := idx div 10;
  memoryStream := TMemoryStream.Create();
  bytes.CompressToStream(memoryStream);
  bytes.LoadFromStream(memoryStream);
  memoryStream.Free;
  command.Image := bytes.GenerateBase64Code();
  command.ControlSum := bytes.GetSectorCRC32(0, bytes.Size);
  SendCommandToRestServer(command);
end.
```


GitHub Repository



<https://github.com/bogdanpolak/class-helpers>

Delphi Class Helpers - VCL RTL and FireDAC

Delphi Support XE4 .. 11.2 version 1.8

Delphi Class Helpers

Make Delphi code more readable using class helpers:

Classic call:

```
StoreDataset (1, mysqlDataSet, true);
```

More readable call:

```
mysqlDataSet.StoreSelected_StopAfterFirstError(  
  function():boolean  
  begin  
    Result := mysqlDataSet.FieldByName('Status').Value = '1'  
  end), fDataStorer);
```

Praktyki kodu solidnego w aplikacjach klient-serwer



S.O.L.I.D.



Pięć zasad S.O.L.I.D.

- Single Responsibility Pojedynczej odpowiedzialności
- Open/Closed Otwarte-zamknięte
- Liskov's Substitution Podstawienia Liskova
- Interface Segregation Segregacji interfejsów
- Dependency Inversion Odwrócenia zależności



SOLID - kto? kiedy? dlaczego?

- Robert C. Martin
 - Rok 2002
 - Artykuł: Principles of Object Oriented Design
- Tworzenie łatwego w utrzymaniu kodu obiektowego
 - Otwarte na rozbudowę
 - Odseparowane moduły
 - Ograniczona odpowiedzialność
 - Odporne na błędy



1

Single Responsibility

Pojedyncza odpowiedzialność



Jedna odpowiedzialność / zadanie

- Co powinno być odpowiedzialnością klasy TBook

- Autor
- Tytuł
- Treść
- Kontekst
- ~~Wyświetlanie / Drukowanie~~
- ~~Zapisywanie~~

```
type
  TBook = class
  private
    FCurrentPage: integer;
    FTitle: string;
    FAuthor: string;
    procedure SetTitle(const Value: string);
    procedure SetAuthor(const Value: string);
  public
    procedure DisplayPage; // Book Stuff
    function TurnPage: integer; // Book stuff
    procedure PrintCurrentPage; // Prints itself
    procedure SaveCurrentPage; // Saves itself
    property Title: string read FTitle write SetTitle;
    property Author: string read FAuthor write SetAuthor;
  end;
```



2

Open / Closed

Otwarte na rozszerzenie – zamknięte na zmianę



Otwarte i zamknięte

- Klasy tak projektujemy aby były
 - Otwarte na rozszerzenie
 - Zamknięte na zmianę

```
procedure TForm2.btnValidateClick(Sender: TObject);
var
    iAge: Integer;
begin
    if edtEmail.Text='' then
        ShowMessage(StrErrEmailEmpty)
    else if not TestEmailFormat(edtEmail.Text) then
        ShowMessage(StrErrEmailWrongFormat);
    if edtAge.Text='' then
        ShowMessage(StrErrAgeEmpty)
    else if not TryStrToInt(edtAge.Text,iAge) then
        ShowMessage(StrErrAgeNotNumber)
    else if (iAge<0) and (iAge>199) then
        ShowMessage(StrErrAgeNotNumber)
end;
```



3

Liskov's Substitution

Podstawienia Liskova



Dziedziczenie i hierarchia

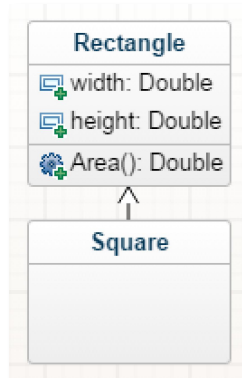
- Wprowadzona w książce
 - Data Abstraction and Hierarchy
 - Barbara Liskov
- Bardzo skomplikowana zasada
 - Podtyp powinien być zastępowalny przez typ bazowy bez konieczności zmiany zachowania
- Opisuje:
 - Jak dobrze zaprojektować hierarchię klas
 - Kiedy wprowadzać klasy abstrakcyjne

Proste ćwiczenie

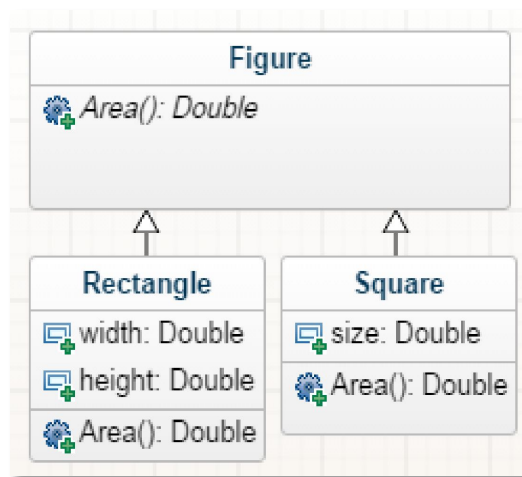
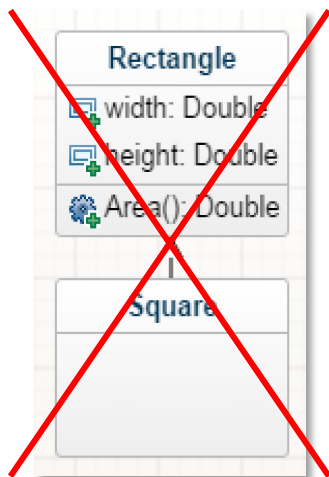
```
var
  r,s: TRectangle;
begin
  r:=TRectangle.Create;
  r.Width:= 10;
  r.Height:= 15;
  ShowMessage(r.Area.ToString);
  s:=TSquare.Create;
  s.Width:= 10;
  ShowMessage(s.Area.ToString);
end;
```

```
type
  TRectangle = class
  private
    FWidth: double;
    FHeight: double;
  protected
    procedure SetWidth (x: double);
    procedure SetHeight (x: double);
  public
    property Width: double read FWidth
      write SetWidth;
    property Height: double read FHeight
      write SetHeight;
    function Area: Double;
  end;

  type
    TSquare = class(TRectangle)
  protected
    procedure SetWidth (x: double);
  public
    property Width: double read FWidth
      write SetWidth;
  end;
```



Poprawne dziedziczenie





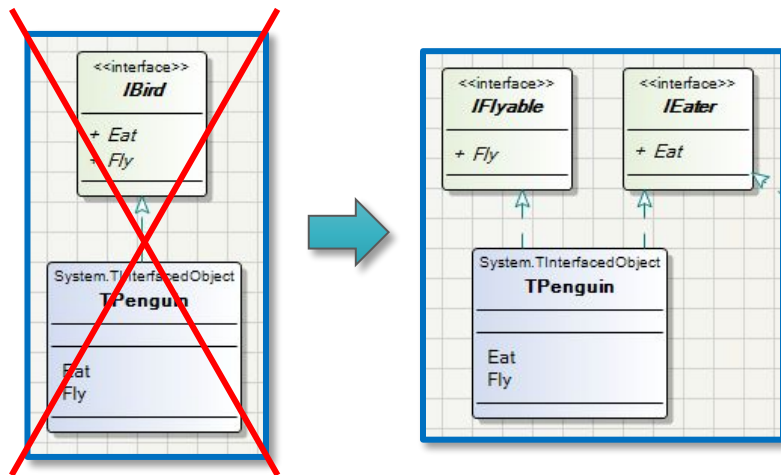
4

Interface Segregation

Segregacji interfejsów

Jak projektować interfejsy

- Zasada
 - Żaden klient nie powinien być zmuszany do zależności od metod, których nie używa
- Inaczej
 - Interfejs powinien być dopasowany do zachowania (nie oszczędzaj na interfejsach)





5

Dependency Inversion

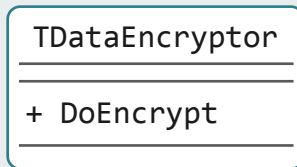
Odwrócenia zależności

Jak projektować interfejsy

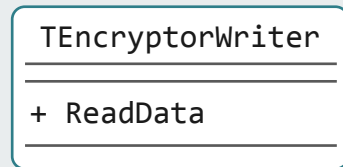
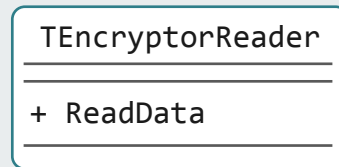
- Zasada

- Klasy wyższego poziomu nie powinny zależeć (być związane) z klasami niższego poziomu.
- Obie klasy powinny być zależne od abstrakcji (interfejsu)
- Abstrakcja nie powinna zależeć od detali.
- Detale powinny zależeć od abstrakcji

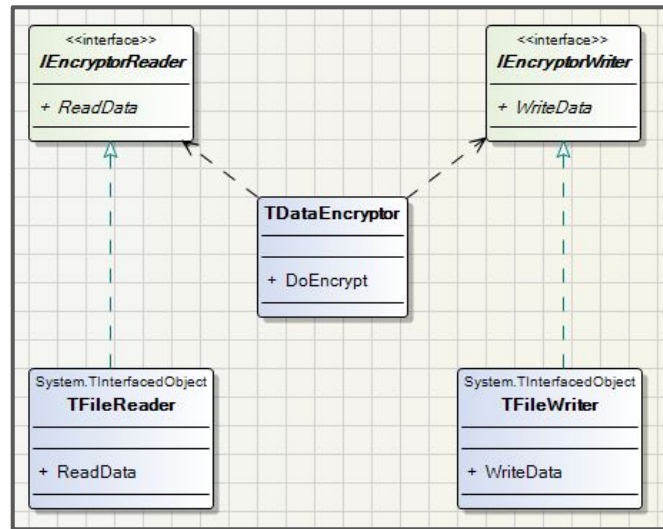
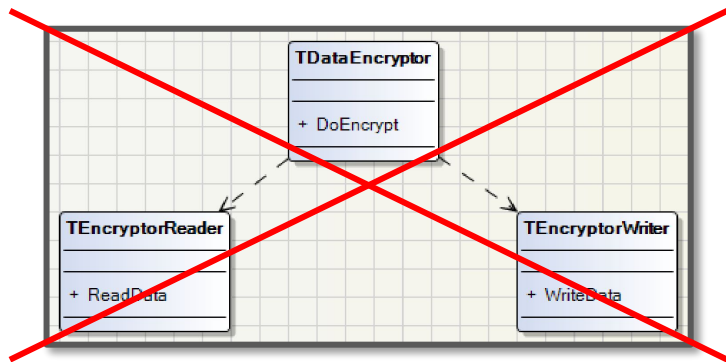
Klasa wyższego poziomu (algorytm szyfrowania)



Klasy niższego poziomu



Ćwiczenie – klasa szyfrująca





Wstrzykiwanie zależności

Dependency Injection Container



Spring Container

uses

```
Spring.Container;
```

begin

```
container := TContainer.Create();
```

try

```
// Register
```

```
// Build
```

finally

```
container.Free;
```

end;

end;

uses

```
Spring.Container;
```

begin

```
GlobalContainer.RegisterType<>();
```

end;

Przykład

deklaracja

```
TRankCalculator = class(TInterfacedObject, IRankCalculator)
private
    fLogger: ILogger;
    fCustomerRepo: ICustomerRepo;
public
    constructor Create(
        const aLogger: ILogger;
        const aCustomerRepo: ICustomerRepo);
    procedure Calculate(aCustomerId: integer): TCustomerRank;
end;
```

rejestracja

```
aContainer.RegisterType<TRankCalculator>();
aContainer.RegisterType<TCustomerRepo>();
aContainer.RegisterType<TCustomerRepo>();
```

stworzenie

```
rankCalculator := aContainer.Resolve<IRankCalculator>();
rank := rankCalculator.Calculate(customerId);
```



Dobre praktyki dla aplikacji Delphi

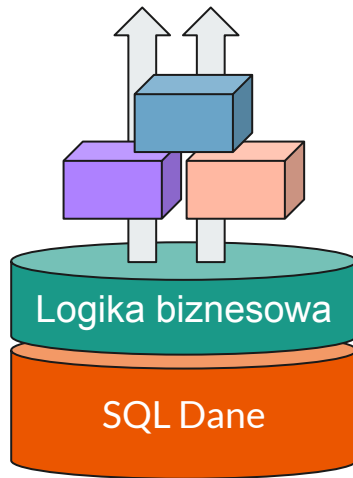
Dług techniczny



Wyciąganie logiki z bazy danych

Dawniej umieszczanie logiki w bazie SQL było zalecanym podejściem, ale tak już nie jest. Stosowane głównie w projektach trwających od dawna, czyli legacy.

Wady: Logika jest trudna w utrzymaniu. Coraz mniej młodych programistów posiada solidne podstawy. Trudno tworzyć kod wariantowy ponieważ możliwości języka są mocno ograniczone, itd.



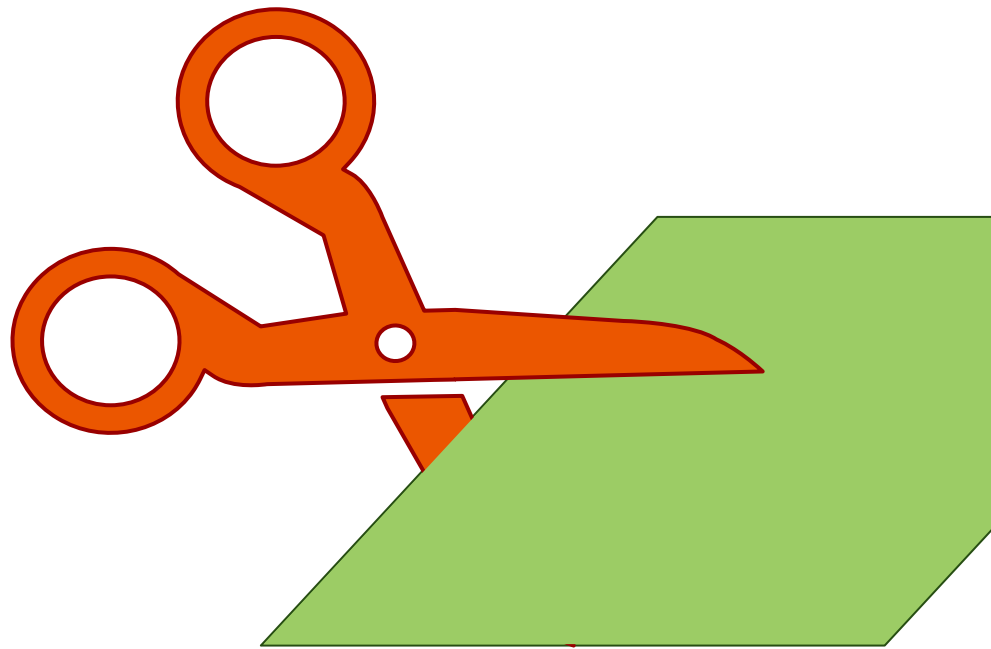


Usuwanie zbędnej logiki z bazy danych

- Vladimir Khorikov - Blog:
<https://enterprisecraftsmanship.com/posts/is-sql-good-place-for-business-logic/>
- Dyskusja **StackOverflow** - rok 2009 - początek transformacji:
<https://stackoverflow.com/questions/1473624/business-logic-in-database-versus-code>
- Dyskusja **SoftwareEngineering** - rok 2016:
<https://softwareengineering.stackexchange.com/questions/314490/business-logic-database-vs-code>
- Artykuł **InfoWorld** (15 września 2021):
<https://www.infoworld.com/article/3633005/put-business-logic-in-the-application-not-the-database.html>



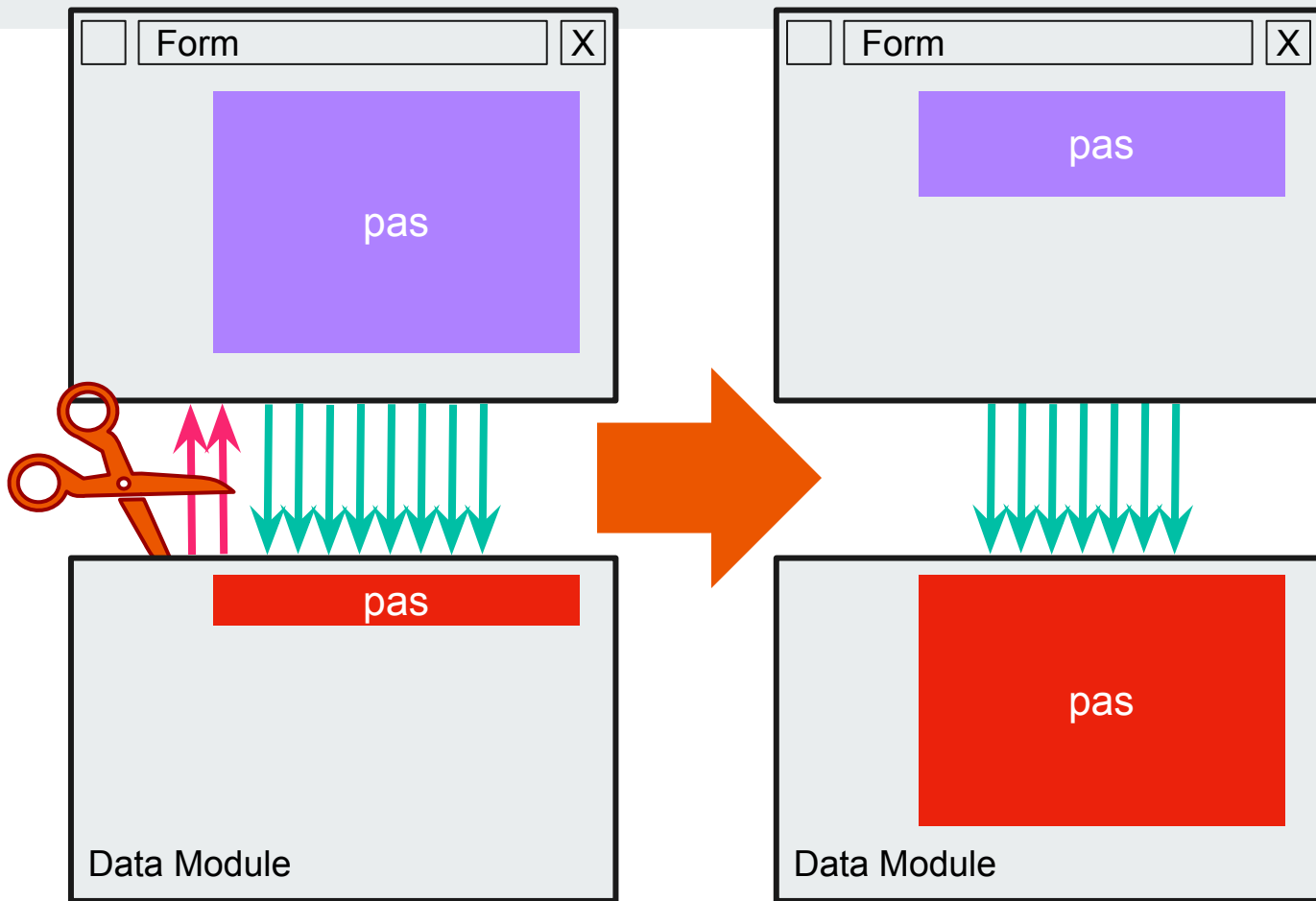
Odcinanie logiki od GUI



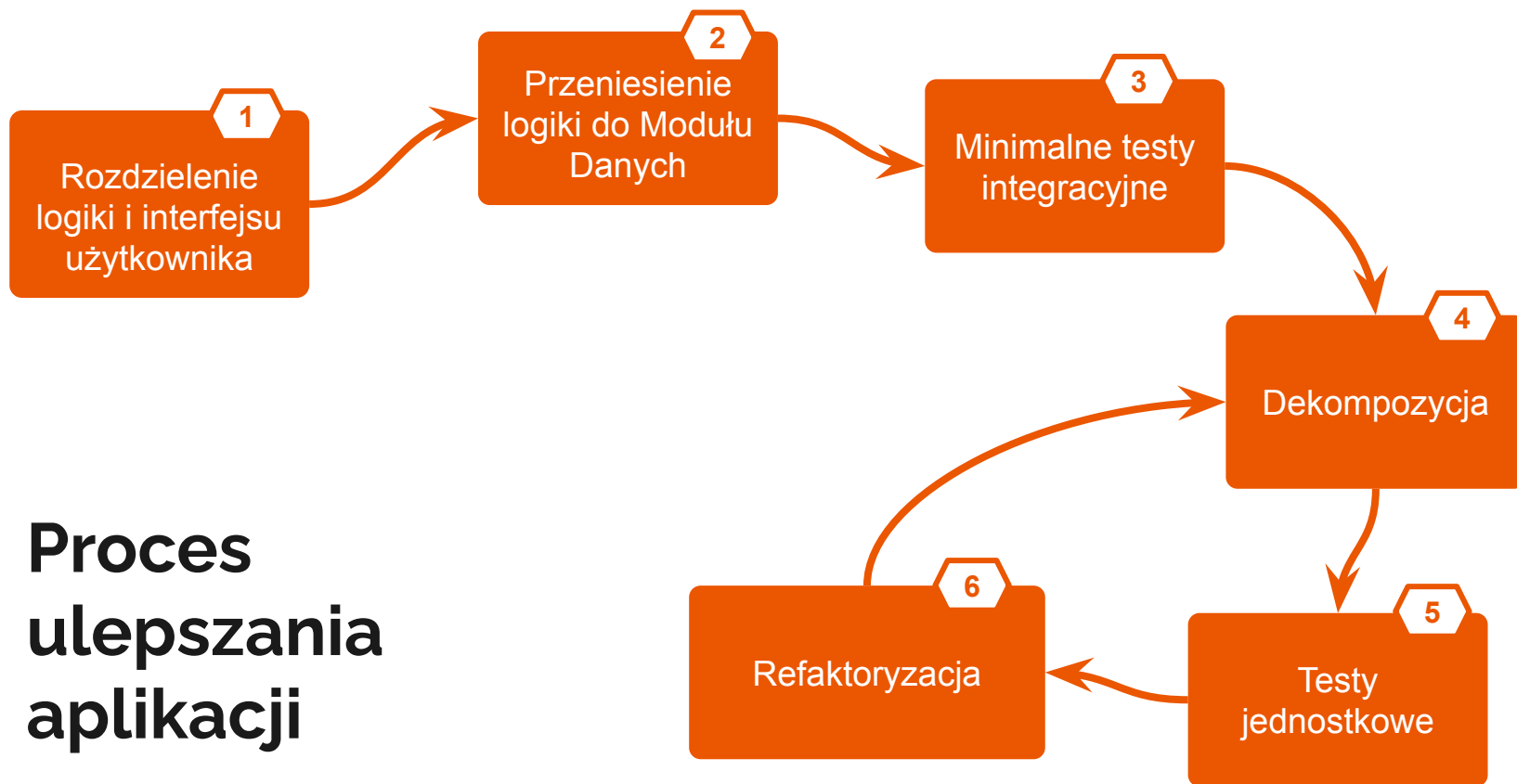
<https://en.delphipraxis.net/topic/4254-cross-platform-messaging-system/>



Pierwszy krok



Proces ulepszania aplikacji





Komunikacja z GUI

Zasady:

1. Formatka powinna znać i korzystać z serwisu, czyli z logiki
2. Serwis **nie może** znać ani odwołać się do żadnej formatki

Rozwiązania:

- Systematyczne odpytywanie serwisu o jego stan
- Interfejs na klasie formatki
- Event handler w formatce zaimplementowany w serwisie
- Wzorzec Obserwator
- Message Event Bus

Jak pisać zrozumiały i prosty kod Delphi?



Nazywanie klas, metod i zmiennych



Kod bez “efektów ubocznych”



Uczciwy kod → Metody szczerą ze sygnaturą



Singleton i fabryki produkujące obiekty

Tworzenie klas w bezpośrednio kodzie biznesowym jest złą praktyką

- Powoduje że kod jest silnie związany
- Zazwyczaj prowadzi do łamania zasad SRP (SOLID #1) i OCP (SOLID #2)
- Trudno zrefaktoryzować wszystkie miejsca w których takie klasy są tworzone
- Zazwyczaj programiści łączą w jednym obiekcie tworzenie klas danych oraz tworzenie procesorów

Algorytmy + Struktury danych = Programy

Singleton

ILogger

```
type
  ILogger = interface
    ['{851E0102-828F-3CA9-85EC-4A34D74155F8}']
    procedure Log(const msg: string; const level: integer = 3);
  end;

  TLoggerFactory = class
  private
    fLogger: ILogger;
  public
    class function CreateLogger: ILogger;
  end;

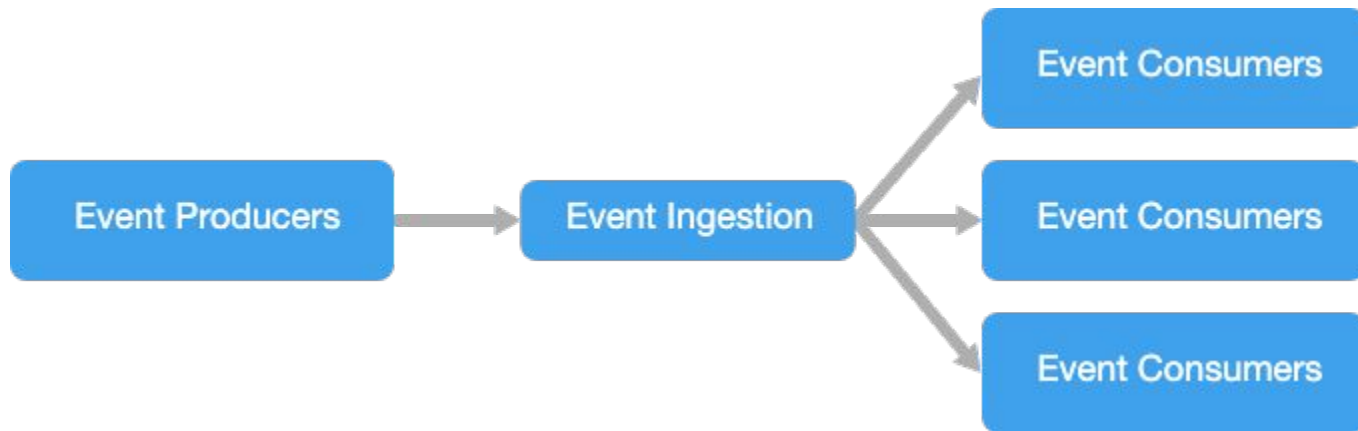
class function TLoggerFactory.CreateLogger: ILogger;
begin
  if not assigned(fLogger) then
    fLogger := TElasticsearchLogger.Create;
  Result := fLogger;
end;
```




Szyna powiadamiania w Delphi

System.Messaging

Szyna powiadamiania - Event Bus



<https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/event-driven>

Delphi Messaging

Producer
/
Publisher

```
uses
    System.Messaging;

procedure TForm2.SendButtonClick(Sender: TObject);
begin
    TMessageManager.DefaultManager.SendMessage(
        Self,
        TMessage<string>.Create('Test message'));
end;
```

Delphi Messaging

Subscribe / Consume

```
procedure TForm1.SubscribeButtonClick(Sender: TObject);
begin
    fSubscriptionId := TMessageManager.DefaultManager
        .SubscribeToMessage(
            TMessage<string>,
            procedure(const Sender: TObject; const M: TMessage)
            begin
                Memo1.Lines.Add(TMessage<string>(M).Value);
            end);
end;

procedure TForm1.UnsubscribeButtonClick(Sender: TObject);
begin
    TMessageManager.DefaultManager.Unsubscribe(
        TMessage<string>,
        fSubscriptionId);
end;
```

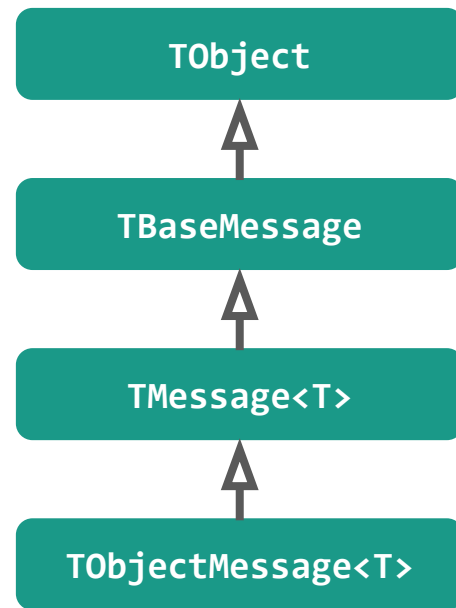
Rodzaje wiadomości

Wiadomość wartościowa

- Wiadomość: String, Integer, TDateTime, record ...
- Klasa: **TMessage<T>**

Wiadomość referencyjna

- Wiadomość: obiekty
- Klasa: **TObjectMessage<T>**



Message Object

```
type
  TBusCommand = class
    MessageId: TGuid;
  end;

  TOverdueRecallingCommand = class(TBusCommand)
    GeneratedBy: string;
    Orders: IList<integer>;
    ValidBy: TDateTime;
    Recommendation: TNotificationChannel;
  end;

TMessageManager.DefaultManager.SendMessage(
  Self,
  TObjectMessage<TOverdueRecallingCommand>.Create(
    'bogdan.polak',
    [1201, 5022, 8001],
    Now()+Minutes(6), ncEmail)
);
```

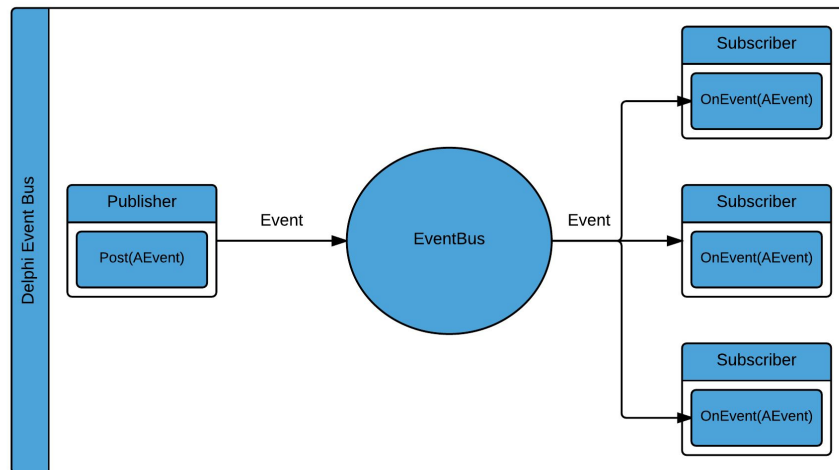
Odbieranie
komunikatu z
wątku
roboczego

```
procedure TForm1.SubscribeButtonClick(Sender: TObject);
begin
    TMessageManager.DefaultManager
        .SubscribeToMessage(
            TMessage<string>,
            procedure(const Sender: TObject; const M: TMessage)
            begin
                msg := TMessage<string>(M).Value;
                TThread.Queue(nil, procedure
                begin
                    Memo1.Lines.Add(msg);
                end);
            end);
end;
```

Alternatywa dla System.Messaging



<https://github.com/spinettaro/delphi-event-bus>




Komunikaty w DEB

```
GlobalEventBus.Post(command);
```

```
type
  ISomethingChangedCommand = interface
    ['{DCFE64D2-9BA8-4949-9BB1-F5CD672E51A2}']
    procedure SetState(const aState: TSomethingState);
    function GetState: TSomethingState;
  end;
```

```
uses
  EventBus;
type
  TForm1 = class(TForm)
    // VCL controls and event handles
  public
    [Subscribe]
    procedure OnSomethingUpdate(
      aCommand: ISomethingChangedCommand);
  end;
```





Testowanie modułów danych

Nowoczesne aplikacje SQL

4. [Nowość] Nowoczesne programowanie w Delphi w aplikacji SQL
 - a. Ograniczanie dostępu osób trzecich do bazy danych oraz weryfikacja bezpieczeństwa
 - b. Zabezpieczanie informacji wrażliwych w bazie danych
 - c. wykrywanie i optymalizacją problemów wydajnościowych aplikacji klient-serwer
 - d. Testowanie funkcji serwerowych, autoryzacji dostępu oraz wydajności serwera baz danych



[Nowość] Nowoczesne aplikacje SQL

1. Ograniczanie dostępu osób trzecich do bazy danych oraz weryfikacja bezpieczeństwa
2. Zabezpieczanie informacji wrażliwych w bazie danych
3. wykrywanie i optymalizacją problemów wydajnościowych aplikacji klient-serwer
4. Testowanie funkcji serwerowych, autoryzacji dostępu oraz wydajności serwera baz danych

Koniec