



Spring4D + DSharp

Como estas bibliotecas podem ajudar a elevar a qualidade do software.

Eurides Baptistella

Palestra
#33

Da sala de reunião à vida real

Embarcadero Conference 2019

Principal Objetivo

- Apresentar **funcionalidades** existentes nestas bibliotecas que podem **elevar a qualidade** dos **softwares** que desenvolvemos.

Objetivos Específicos

- Spring4D:

- Sobre
- Recursos:
 - Collections;
 - Nullable;
 - Managed;
 - Interceptor;
 - Dependency Injection Container;
 - Design Patterns;
 - ORM;

Objetivos Específicos

- DSharp:

- Sobre;
- MVVM;

Spring4D

- Biblioteca de código aberto (Open Source) para Delphi XE e versões superiores.
- Consiste em 3 principais módulos:
 - ➔ **Base** (RTL Extension, Logging...);
 - ➔ **Core** estrutura de injeção de dependência e interceptador;
 - ➔ **Persistência** ORM.

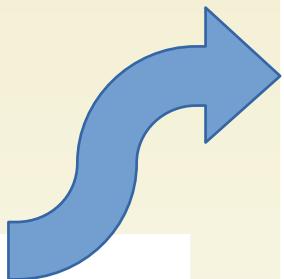
- Sua utilização é gratuita para uso comercial (esta sob licença Apache 2.0)
- Biblioteca mantida principalmente por Stefan Glienke com a colaboração de outros desenvolvedores por meio de Pull Requests e Issues.

Spring4D - Sobre

- Versão corrente: 1.2.2

Screenshot of a GitHub branches page for the Spring4D repository. The page shows the following branches:

Branch	Behind	Ahead	Updated
master	MAIN	DEVELOPMENT	2018-11-26
develop	2 154		2019-10-08
feature/reactive	2 181		2019-09-10
feature/container-refactoring	2 170		2019-08-28
feature/priorityqueue	2 141		2019-08-07
feature/coroutines	2 140		2019-08-05
hotfix/1.2.3	0 2		2019-04-23

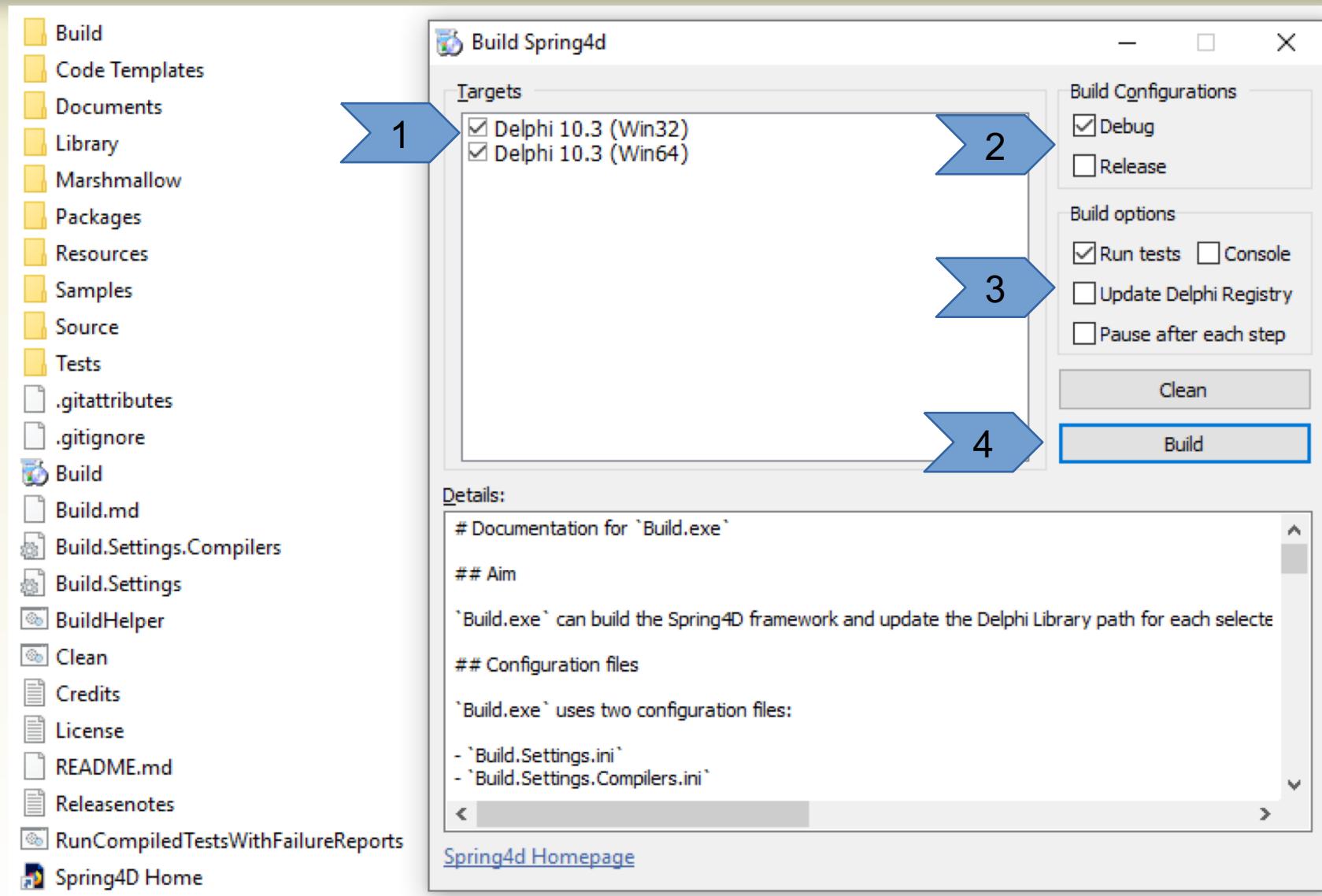


Screenshot of a GitHub commits page for the Spring4D repository, showing the commit history for Stefan Glienke. The commits are listed in reverse chronological order.

Author	Commit	Message	Date
Stefan Glienke	532cd25	Removed virtual from TRefCountedObject methods for IInterface	2019-10-08
Stefan Glienke	32094b0	Added routines to mark object reference as weak which causes them to ...	2019-10-08
Stefan Glienke	c04995a	Implemented CreateList folding for string via CreateStringList	2019-10-08
Stefan Glienke	4abcf01	implemented support for resolving lazy for string and integer types	2019-10-02
Stefan Glienke	5195b87	optimization - replaced zero length check with nil check on dynamic arrays	2019-09-26
Stefan Glienke	b57fe46	fixed missing ownsObjects parameter	2019-09-12
Stefan Glienke	e228609	added 10.3 package	2019-09-10
Stefan Glienke	d42d253	fixed AV in unit tests resulting from RTL bug	2019-09-02
Stefan Glienke	63c1dfc	updated jedi.inc	2019-09-02
Stefan Glienke	6a34402	fixup! [experimental] support for record equalitycomparer	2019-08-30
Stefan Glienke	202f568	added IEnumerable<T> to implemented interfaces	2019-08-30
Stefan Glienke	9d64784	fixed #324	2019-08-30
Stefan Glienke	4a2b9c8	implemented TObservable.Timer	2019-08-30
Stefan Glienke	b30ea27	implemented SelectMany and SequenceEqual	2019-08-30

- Disponível para download no repositório git do Bitbucket:
 - <https://bitbucket.org/sglienke/spring4d.git>
- Use sua ferramenta favorita, faça pull e a configuração dos fontes no Delphi.
- Ou, utilize o executável Build.exe disponível no repositório do git para configurar o ambiente.

Spring4D - Sobre



- Baseado em Interfaces, que por si só facilita o gerenciamento de memória;
- Facilmente utilizado para retornar ou enviar como parâmetro em procedimentos/funções;

Spring4D - Collections

- Adiciona uma série de recursos em coleções, semelhantes ao fornecido por outras linguagens (como Stream do Java):
 - Sort
 - Last/First
 - OrDefault
 - Min/Max/Sum/Group
 - Where

- Forma segura de adicionar o estado nulo ao tipo de valor existente.
- Permite validar (HasValue) se a variável recebeu valor.

- Facilita a instanciação e destruição de objetos sem necessidade de se preocupar com a liberação da memória. Também facilita a inicialização (valor default) de atributos de classe.
- Custom Attributes [Managed], [Default];
- Herança de das classes TManagedObject e TManagedInterfacedObject.

Spring4D - Interceptor

- São filtros que permitem executar operações antes ou depois das chamadas a métodos e que podem, inclusive, substituir a chamada ao método original.
- Muito comum utilizar em logs/traces;
- Verificar permissão de acesso.

- Injeção de dependência é uma das formas de se aplicar **Inversion of Control** (IoC).
- A referência que IoC faz é de desacoplamento, invertendo o controle da instanciação dos objetos.
- Você faz DI quando retira as dependências existentes em uma parte do programa, possivelmente uma classe, desacoplamento assim uma da outra.

- A melhor definição de DI é permitir que estados e comportamentos sejam determinados através de passagem de parâmetros.
- Ou seja, é permitir que você refencie uma classe ainda não conhecida durante o desenvolvimento da classe atual.
- Isto pode ser feito através de parâmetros nos construtores, métodos comuns ou em propriedades da classe atual.

- Containers: Um container de injeção de dependência gerencia e automatiza as instanciações.
- Dizemos pra ele como um objeto deve ser criado e então sempre que o precisarmos, basta que usemos o container para obtê-lo.

- Singleton
- Observer
- Factory
- Decorator

- Técnica utilizada para fazer mapeamento entre sistemas e bancos de dados, onde as tabelas do banco são representadas por classes e os registros das tabelas seriam instâncias destas classes.
- Spring4D utiliza o ORM Marshmellow.

- **Session:** Classe que faz a ligação entre a aplicação e o ORM. Faz a abstração do serviço de persistência.
- Principal objetivo desta classe é oferecer operações de criação, leitura, atualização e exclusão de instâncias de classes de entidade.

- **ConnectionFactory**: Classe responsável por retornar uma instância de objeto que implementa IDBConnection, que é utilizado para atribuir para a Session uma conexão com o banco de dados.

```
dtSQLite  = 'DRIVER_TYPE_SQLITE';
dtADO      = 'DRIVER_TYPE_ADO';
dtMSSQL    = 'DRIVER_TYPE_MSSQL';
dtASA      = 'DRIVER_TYPE ASA';
dtOracle   = 'DRIVER_TYPE_ORACLE';
dtDBX      = 'DRIVER_TYPE_DBX';
dtUIB      = 'DRIVER_TYPE_UIB';
dtZeos     = 'DRIVER_TYPE_ZEOS';
dtMongo    = 'DRIVER_TYPE_MONGO';
dtFireDAC = 'DRIVER_TYPE_FIREDAC';
```

Spring4D - ORM

```
if not(FDBXConnection.Connected) then
  FDBXConnection.Open;

if not(FDBXConnection.Connected) then
  raise TConnectionFactoryException.Create('ADBXConnection can not get a connection!'); ↴

FDBConnection           := TConnectionFactory.GetInstance(dtDBX, FDBXConnection);
FDBConnection.AutoFreeConnection := False;

FDBConnection.QueryLanguage := qlPostgreSQL;

FSession := TSession.Create(FDBConnection);
.
```

Spring4D - ORM

```
type
[Entity]
[Table('Products')]
TProduct = class
private
  [Column('PRODID', [cpRequired, cpPrimaryKey, cpNotNull, cpDontInsert], 0, 0, 0, 'Primary Key')]
  [AutoGenerated]
  FId: Integer;
private
  FName: string;
  FPrice: Currency;
  FQuantity: Integer;
  fCreationDate: TDate;
  fCreationDateTime: TDateTime;
  fCreationTime: TTime;
public
  property ID: Integer read FId;
  [Column('PRODNAME', [], 50, 0, 0, 'Product name')]
  property Name: string read FName write FName;
  [Column('PRODPRIce', [], 0, 0, 0, 'Product price')]
  property Price: Currency read FPrice write FPrice;
  [Column('PRODQUANTITY')]
  property Quantity: Integer read FQuantity write FQuantity;
  [Column('PRODCREATIONDATE')]
  property CreationDate: TDate read fCreationDate write fCreationDate;
  [Column('PRODCREATIONDATETIME')]
  property CreationDateTime: TDateTime read fCreationDateTime write fCreationDateTime;
  [Column('PRODCREATIONTIME')]
  property CreationTime: TTime read fCreationTime write fCreationTime;
end;
```

- **TsimpleRepository<TCustomer,string>.FindOne('ABC')**
- **TsimpleRepository<TCustomer,string>.FindAll(): IList<TCustomer>**
- **TsimpleRepository<TCustomer,string>.FindWhere(ICriterion):
IList<TCustomer>**
- **TsimpleRepository<TCustomer,string>.Save(TCustomer): TCustomer**
- **TsimpleRepository<TCustomer,string>.Delete(TCustomer)**
- **TsimpleRepository<TCustomer,string>.DeleteAll()**
- **TsimpleRepository<TCustomer,string>.Exists()**

→ Exemplos:

- Disponíveis no próprio repositório, diretório Samples, ou então observe os testes implementados.

DSharp

Uma pequena biblioteca para fornecer ligação de dados no Delphi.

Não requer componentes para vincular às propriedades. Ele também fornece programação orientada a aspectos.

- Disponível para download no repositório git do Bitbucket:
<https://bitbucket.org/sglienke/dsharp.git>
- Use sua ferramenta favorita, faça pull e a configuração dos fontes no Delphi.

- Model – View – ViewModel
- Pattern criado originalmente para Windows Presentation Fundation (WPF). Utiliza geralmente bindings para fazer a ligação da View com o ViewModel.

→ Exemplos

- Disponíveis no próprio repositório, diretório Samples, ou então observe os testes implementados.

- Projeto conceitual
 - git clone <https://github.com/ebaptistella/delphi-mvc-concept.git>



Obrigado



eurides.baptistella@gmail.com



github.com/ebaptistella



linkedin.com/in/ebaptistella



twitter.com/ebaptistella

Da sala de reunião à vida real

Embarcadero Conference 2019

ID #33