European Conference
Design patterns with Spring4D
About me

Primož Gabrijelčič
http://primoz.gabrijelcic.org
• programmer, MVP, writer, blogger, consultant, speaker
• Blog http://thededphigeek.com
• Twitter @thededphigeek
• Skype gabr42
• LinkedIn gabr42
• GitHub gabr42
Hands-on Design Patterns with Delphi

- Packt Publishing
- www.packtpub.com
- http://tiny.cc/pg-dpd
My other books

http://tiny.cc/pg-ppotl

Parallel Programming with OmniThreadLibrary

http://tiny.cc/pg-dhp

Delphi High Performance

Build fast Delphi applications using concurrency, parallel programming and memory management
Design patterns
Design patterns

- Pattern = template for a solution
- Pattern = common vocabulary
- Pattern ≠ recipe

- architectural patterns > design patterns > idioms
- design patterns ≠ design principles (SOLID, DRY ...)

- https://en.wikipedia.org/wiki/Software_design_pattern
Categories

- Creational patterns: *delegation*
  - Creating objects and groups of objects
- Structural patterns: *aggregation*
  - Define ways to compose objects
- Behavioral patterns: *communication*
  - Define responsibilities between objects
- Concurrency patterns: *cooperation*
  - Make multiple components work together
Patterns and Spring4D
Creational patterns

Pattern
- Singleton
- Dependency injection
- Lazy initialization
- Factory method

Spring4D
- TSingleton
- Spring.Container ...
- Lazy
- TFactoryMethod, TFactory
Structural patterns

Pattern
- Proxy
- (Marker interface)

Spring4D
- Shared, Spring.Interception, Mock
- Spring.Reflection
Behavioral patterns

Pattern
- Iterator
- Observer
- Specification

Spring4D
- IEnumerable, Spring.Collections
- TObservable, Event
- TSpecification
Concurrency patterns

Pattern
- Optimistic initialization

Spring4D
- TLazyInitializer
Creational patterns

https://en.wikipedia.org/wiki/Creatational_pattern
Singleton

“Ensures that a class has only one instance.”

TSingleton

Spring.DesignPatterns

Lazy initialization

https://en.wikipedia.org/wiki/Singleton_pattern
Dependency injection

“Put appropriate instances in; don’t let the object create them.”

⇒ Dependency injection

Lazy initialization

“Delays the creation of an object until it is actually needed.”

ILazy, ILazy<T>, TLazy, TLazy<T>, Lazy<T>

Spring

https://en.wikipedia.org/wiki/Lazy_initialization
Factory method

- “An interface for creating a single object.”

- TFactoryMethod<T>, TFactory<T>
  - Spring.DesignPatterns

  ➞ Dependency injection

- "https://en.wikipedia.org/wiki/Factory_method_pattern"
Structural patterns

https://en.wikipedia.org/wiki/Structural_pattern
Proxy

“Provides a replacement for another object so it can control access to the object.”

- `IShared<T>, Shared<T>, Shared`
  - `Spring`
- `Spring.Interception, Mock<T>`
  - `⇒ Interception and dynamic proxy`

- `https://en.wikipedia.org/wiki/Proxy_pattern`
Marker interface

- "Allows us to associate metadata with a class."

- HasCustomAttribute<T>, GetCustomAttribute<T>
  - Spring.Reflection

- https://en.wikipedia.org/wiki/Marker_interface_pattern
Behavioral patterns

https://en.wikipedia.org/wiki/Behavioral_pattern
Iterator

“Provides a way to access elements of an aggregate object without exposing the underlying implementation.”

- IEnumerable
- Spring.Collections

⇒ Collections

Observer
[Publish-Subscribe]

“A system where a change of objects results in all of its dependents being notified about the change.”

- `IObservable<T>, TObservable<T>`
- `IEvent, IEvent<T>, Event<T>, INotifyEvent, INotifyEvent<T>`

- Spring.

Specification

“Allows business rules to be recombined by chaining them together using boolean logic.”

ISpecification<T>, TSpecification<T>
  Spring.DesignPatterns

Concurrency patterns

https://en.wikipedia.org/wiki/Concurrency_pattern
Optimistic initialization

- Lazy initialization
  - TLazyInitializer
  - Spring

- https://en.wikipedia.org/wiki/Lock_(computer_science)
Design pattern pitfalls
Pattern problems

- Using patterns to architect the software
- Blindly applying patterns
- Blindly copying C++/Java implementations

- Design patterns are a **tool**, not a **goal**!