ITDevCon

Delphi European Conference

Tecniche avanzate di Unit Testing

Marco Breveglieri
Software and Web Developer, Trainer and Consultant

October 25/26 2012 VERONA

bit Time software

Marco Breveglieri

- *Sviluppo software e Web*
- *Consulenza e tutoring*

Blog: **www.compilaquindiva.net**

Mail: **marco@abls.it**

Introduzione

ITDevCon

Non si vendono metodologie

TDD

BDD

Scrum

DDT?

...

Si parla di Unit Test a 360°

- Unit Test: la definizione
- Creare Unit Test «come si deve»
- Tool per scrivere ed eseguire test in Delphi
- Mock, Stub e altri strumenti interessanti
- Esperienze e conclusioni
- Q & A

Unit Test

A Unit Test is a piece of code (usually a method) that invokes another piece of code and checks the correctness of some assumptions afterward.

If the assumptions turn out to be wrong, the unit test has failed.

A «unit» is a method or function.

(fonte: Wikipedia)

Le caratteristiche di un «buon» Unit Test

- Automatizzato
- Ripetibile
- Consistente
- Facile
- Accessibile
- Riutilizzabile
- Veloce
- Onnipotente (rispetto al sistema sottoposto a test)

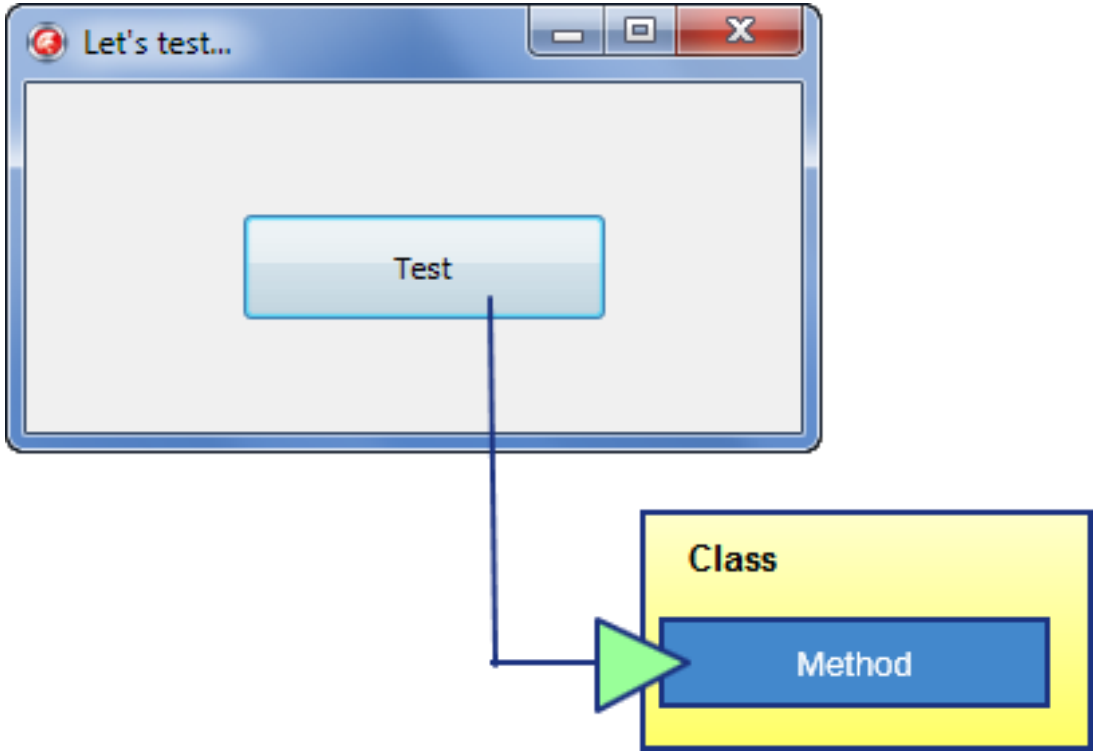
Integration Test

Integration testing *is testing a unit of work with one or more of its real dependencies such as time, network, database or code you cannot control such as threads and random number generators.*

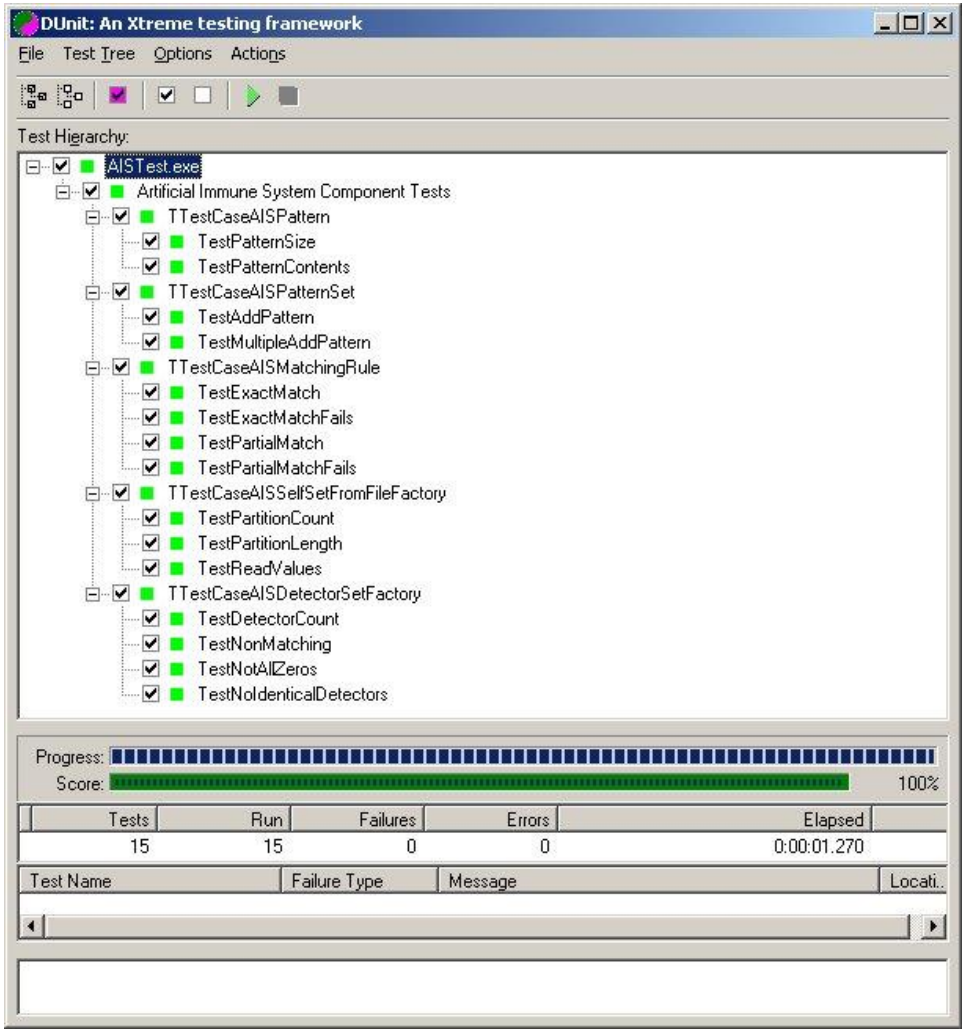
Unit Testing in Delphi

ITDevCon

Il più semplice degli Unit Test



Meglio usare DUnit!



- Consente di dare una struttura ai test
- Rende i test automatizzabili e ripetibili
- Riduce gli errori nel codice del test
- Produce un rendiconto dell’esito dei test
- Aggiungono generalmente una interfaccia utente (GUI)
- Può fornire supporto alla gestione delle dipendenze del sistema sotto test

Un po' di codice...

Gestire le dipendenze

ITDevCon

Una definizione

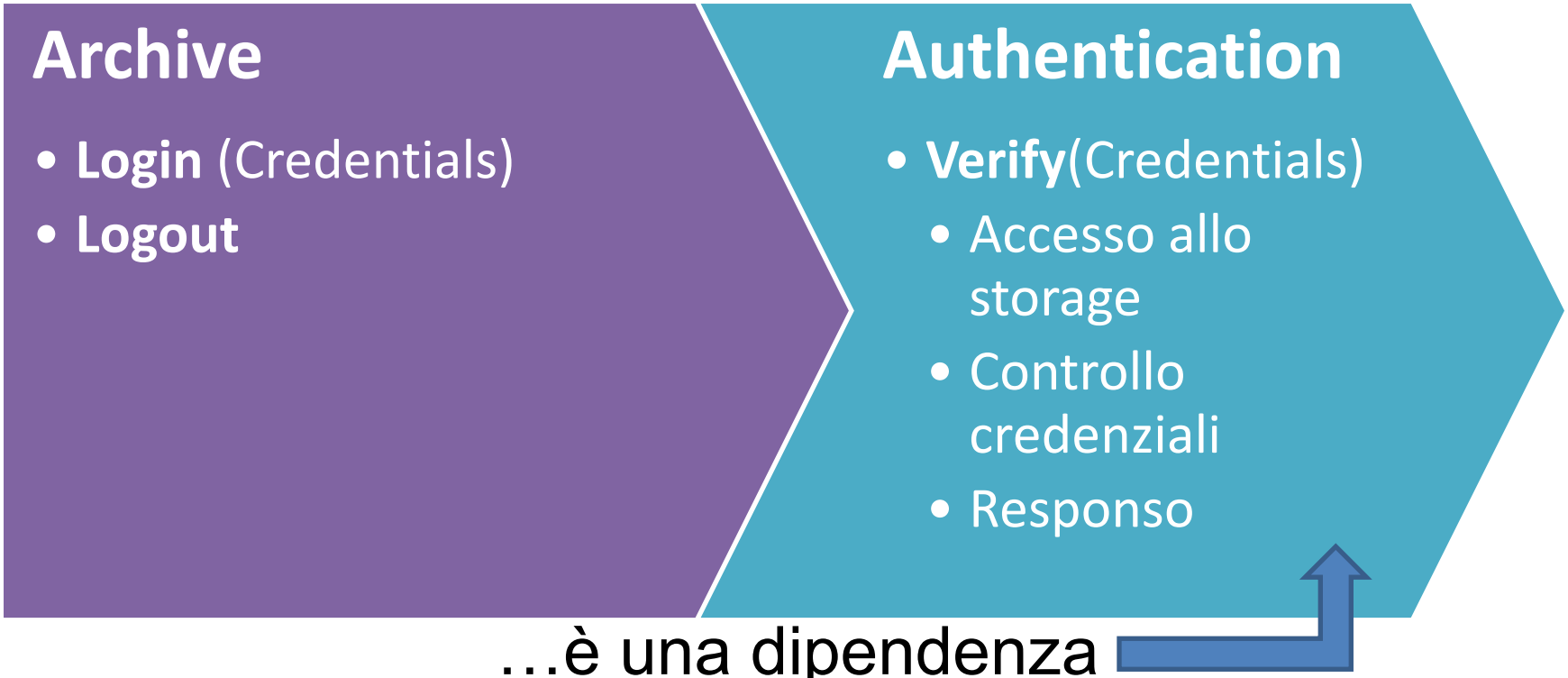
Una dipendenza esterna *è un oggetto del sistema con cui il codice sotto test interagisce e sul quale non ha controllo.*

- File system
- Network
- Database

Lo «stub» è un componente controllabile che rimpiazza una dipendenza esterna.

In questo modo, è possibile eseguire il test senza avere a che fare con tale dipendenza.

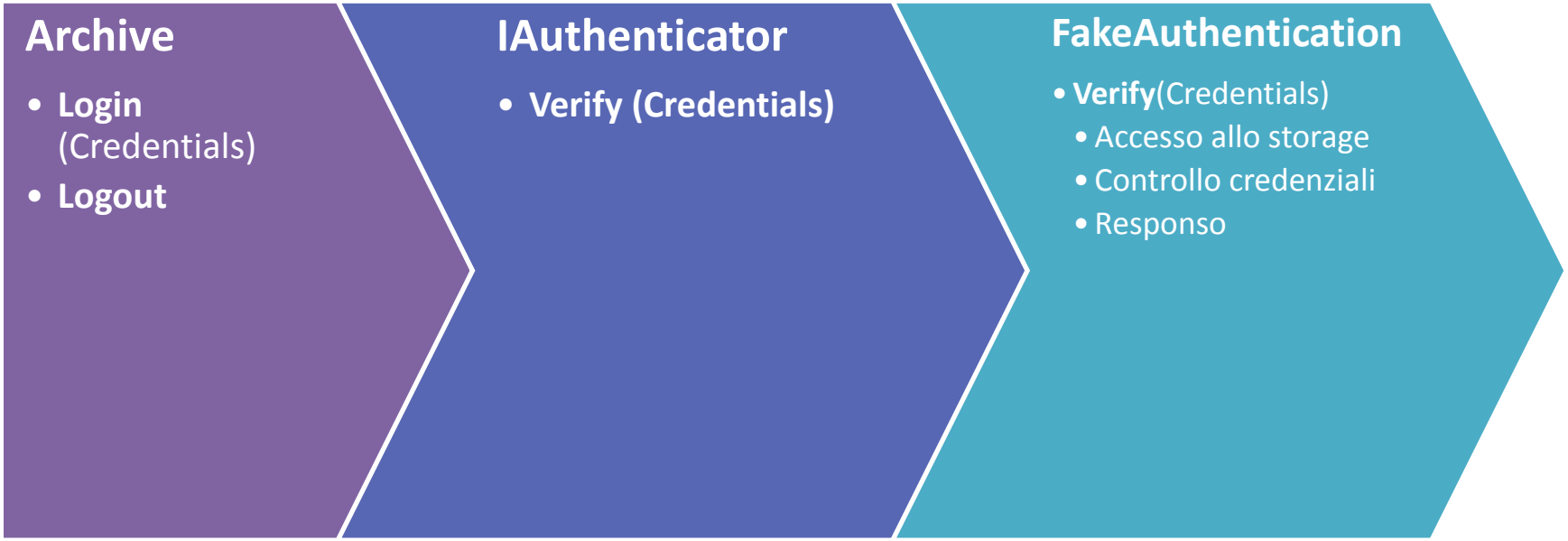
Accesso ad archivio con appoggio a servizio di autenticazione delle credenziali



Risolvere le dipendenze: un esempio

E’ possibile:

- introdurre una interfaccia
- implementare un «finto» servizio di autenticazione



Un altro po' di codice…

Come posso creare – in fase di test e nell'applicazione vera e propria – tutte le dipendenze richieste dagli oggetti?

- Testing: generalmente, il testing framework fornisce il supporto necessario
- Esecuzione: si sfrutta la «Dependency Injection» (DI) *

(*) partecipa al talk di Daniele Teti

Il controllo dell'esito dell'esecuzione del test si basa su

- **State-based testing** (*result-driven*)
  lo stato dell'oggetto

- **Interaction testing** (*action-driven*)
  l'interazione con altri oggetti per ricevere input o fornire output

Esempio dell'irrigatore

- **State-based**
  - l'erba del prato è verde?
  - la terra è abbastanza umida?
- **Interaction-based**
  - quante volte viene irrigato il prato in un lasso di tempo?
  - quanta acqua viene erogata?

Un «mock» è un oggetto che interagisce con il sistema sotto test e determina se il test è passato oppure no.

ATTENZIONE:
ogni test non dovrebbe contenere più di un «mock object»

Ancora codice…

Scrivere «fake» stanca…

- Ci vuole tempo per scrivere il codice
- Quando le classi sono complesse, la scrittura di codice diventa insostenibile
- Per testare più condizioni, è necessario farcire il mock con serie infinite di variabili
- In alcuni casi, diventa impossibile riutilizzare uno stub/mock per altri test

Per fortuna, c’è una soluzione…

Isolation Frameworks

Un isolation framework *è una libreria programmabile che*

- aiuta a creare *fake* (mock e stub) con più facilità
- solleva lo sviluppatore dalla necessità di scrivere codice per implementare i fake
- acquisisce informazioni sulle interazioni tra gli oggetti

Delphi Mocks: è un framework creato da *VSoft Technologies*, i creatori di FinalBuilder, che utilizza

- le recenti caratteristiche del linguaggio Delphi
- le funzionalità della RTTI estesa
- la classe TVirtualInterface

per creare tipi «on the fly» che fungono da *mock*.

Come si usa?

1. Si abilita la generazione di RTTI con {$M+}
2. Si crea il «mock» del tipo che ci interessa
3. Si procede con il setup dell'oggetto
4. Si procede con il controllo delle *expectation*
5. Si verifica lo stato o il risultato dell'interazione

Facile, no? ☺

Ancora codice…

Conclusioni

I vantaggi degli Unit Test

- Identificazione (e risoluzione) di bug
- Semplificazione del mantenimento di codice «legacy»
- Incremento della fiducia dello sviluppatore nel proprio codice…
- …e anche nel codice degli altri membri del team
- Controllo dei difetti di regressione nelle release
- Censimento di nuovi casi di studio segnalati dai clienti
- Garanzia di una buona qualità generale del software
- Fondazione solida di blasonate metodologie di programmazione (es. TDD)
- Inserimento perfetto in un contesto di sviluppo iterativo con Continuous Integration

Approfondimenti

- Delphi Mocks (framework)
  https://github.com/VSoftTechnologies/Delphi-Mocks
- Art of Unit Testing – Roy Osherove (libro)
  http://artofunittesting.com/

Q & A

Grazie!

ITDevCon