Diese Wiki Seite sieht etwas verhagelt aus, da sie in erster Linie im Präsentationsmodus funktionieren muß.

Browser im Kiosk Modus: * Safari: keine Ahnung * Firefox recht gut per Plugin: https://addons.mozilla.org/de/firefox/addon/1568 * Opera von Haus aus

= Unit Testing Cocoa

The main thing that distinguishes legacy code from non-legacy code is tests, or rather a lack of tests. Michael Feathers

Vortrag am 17. März 2010

ungeordnete Stichpunkte * Michael Feathers http://www.objectmentor.com/omTeam/feathers_m.html * Martin Fowler

- CI http://www.martinfowler.com/articles/continuousIntegration.html
- OT: VCS http://martinfowler.com/bliki/VersionControlTools.html

* Michael T. Nygard http://www.pragprog.com/titles/mnee/release-it * SQLite Testing http://www.sqlite.org/testing.html

== Was ist ein Unit Test?

* Testprogramm für einzelne Programmbausteine (Klassen, einzelne Methoden), * zustandslos, bereitet die Vorbedingungen selbst vor, * automatisierbar, prüft alle Nachbedingungen, Schritt im Build Prozeß, * normalerweise in der gleichen Sprache geschrieben wie der zu testende Code, * nicht Teil des ausgelieferten Programms. == Quelltext Impression

```
#import <SenTestingKit/SenTestingKit.h>
@interface MyClassTC : SenTestCase {}
@end

#import "MyClass.h"
@implementation MyClassTC
-(void)setUp {
}

-(void)tearDown {
```

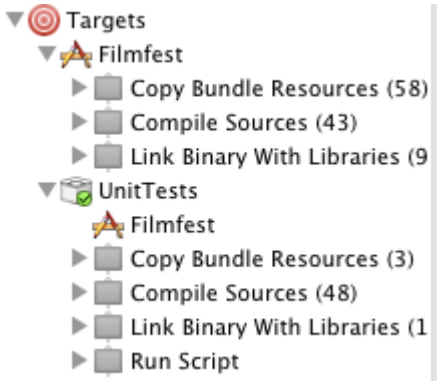
```
}

-(void)testMethodXY {
  ...
  STAssertEqualObjects(@"expected", ..., @"fail");
}
@end
```

== Was nützen mir Unit Tests?

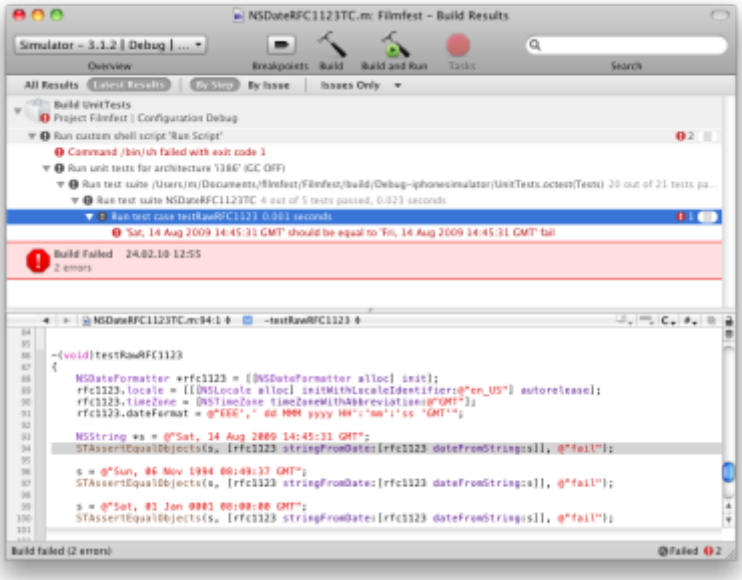
* Klarheit der Abhängigkeiten in meinem Code (Stichwort Dependency Injection & Inversion of Control), * Refactoring ohne Nervenkitzel, * eindeutig definiertes Verhalten (Design by Contract), Förderung der Teamarbeit, * keine „vergessenen“ Tests, da jeder Build alle prüft, Bedingung für Continuous Integration. == Wie geht's prinzipiell? * Test Framework zur bequemen Entwicklung, Ausführung und für Berichte, * normalerweise pro Logikbaustein (Klasse) eine Testklasse, * pro Bug / gewünschtem Verhalten eine Testmethode:

```
-(void)testRFC1123 {
  NSString *s = @"Fri, 14 Aug 2009 14:45:31 GMT";
  STAssertEqualObjects(s, [[NSDate dateFromRFC1123:s]
    rfc1123String], @"fail");
}
```

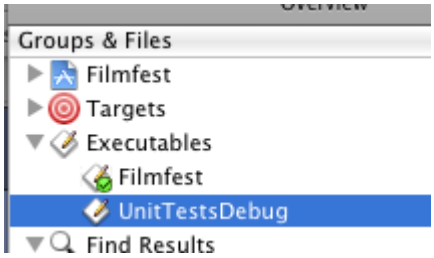
== Wie geht's mit Cocoa



* SenTestingKit / OCUnit kommt von Apple mit - gibt aber auch andere Implementierungen und Zusätze, * neues XCode Target: „Cocoa > Unit Test Bundle“ * direkte Abhängigkeit vom bisherigen Target, fertig.

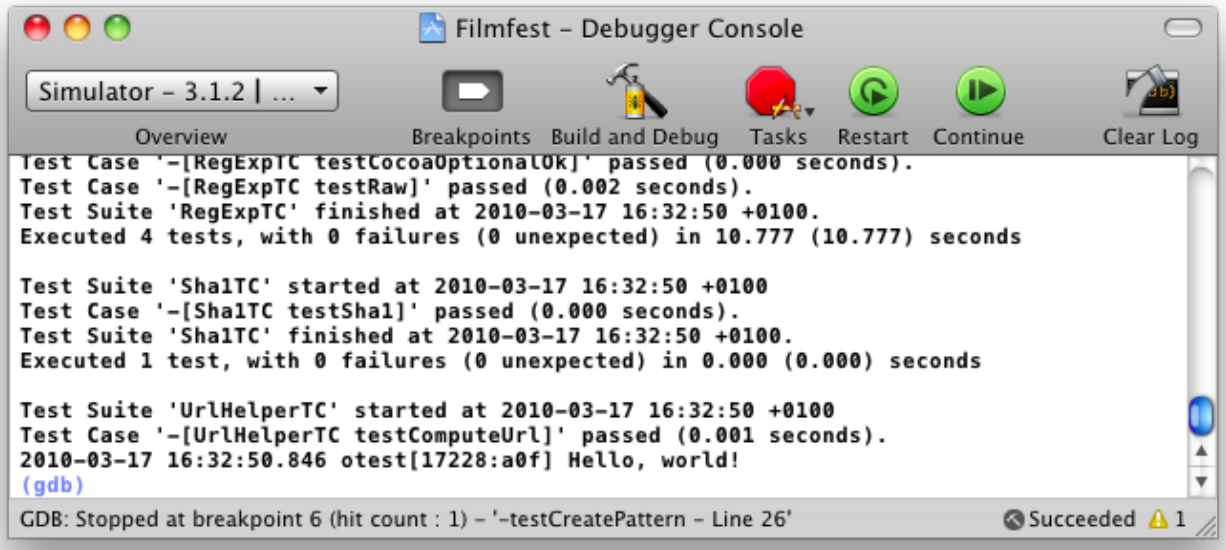


* ...und jetzt der schwierige Teil: == Step & Trace



dazu brauchen wir ein „Custom Executable“: * Pfad `Developer/usr/bin/otest` relativ zu „Current SDK“ * einige Parameter und etliche Umgebungsvariablen das können wir dann als `Active Executable` starten und debuggen. **Vorsicht:** * Parameter und Umgebungsvariablen gehen in die `*.pbxuser` Datei ein und gehen drum leicht verloren. * die Debugsitzung startet nur, wenn alle Tests erfolgreich sind (Build Abhängigkeit) - ich kommentiere die kritischen `STAsserts` zeitweise aus (Holzhammer)

== NSLog * Terminal: `xcodebuild` auf das Test Target loslassen und die NSLogs kommen auf die Konsole * XCode: `Custom Executable` als `Active Executable` und schon sehen wir die NSLogs auch.



== Exkurs: Entwicklungsprozesse * Extreme Programming (XP), * Test Driven Development (TDD), * Documentation Driven Development (DDD), * Design by Contract (DbC).

---

* DDD mit Link zu "presentation on Documentation-Driven Development" (Slideshare)

== Was ich vermisse * einfachere Projekteinrichtung, * benutzerbezogene Einstellungen abschaffen, * Testabdeckung messen und anzeigen. (CoverStory?)

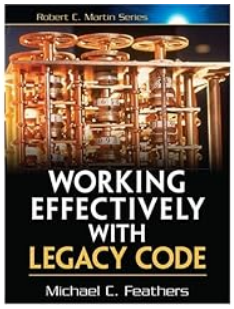
```
120        public void setDrawToTeeTime(final do
121   46       if (Double.isInfinite(drawToT
122    0           mu = 0;
123             else
124   46           mu = 2.0 * (IceSize.F
125                                / (g
126   46        PropModelHelper.setDrawToTeeT
127   46    }
128   }
```

Report generated by Cobertura 1.9.2 on 23.07.09 17:23.

== Locker bleiben *Don't stress about unit tests. They are intended as a tool for ensuring good test coverage and memory management. Use them in that way to aid your development process. ADC*

* Unit Tests verhindern natürlich nicht jeden Bug, * Testing-Taliban schrecken moderate Seelen oft ab, * ein laufendes Programm ist wichtiger als 100% Testabdeckung, * die ersten Tests für Spaghetti Code können allerdings frustrieren. == Querverweise / Lesetips

* Michael Feathers: Working Effectively with Legacy Code * Erica Sadun bei TUAW



XCode Anleitungen die ich schlußendlich verstanden habe: * Cocoa with Love: A sample iPhone App with unit tests * Cocoa with Love: A sample Mac Application with unit tests * meine Zusammenfassung zur XCode Projekteinrichtung == Vielen Dank

für Eure Aufmerksamkeit.

Feedback willkommmen an Marcus Rohrmoser

Die Folien zum Nachlesen gibt's hier:

http://mro.name/go/cocoaheads_testing

---

aus Sicht eines iPhone Entwicklers.

ein Vortrag bei den CocoaHeads von

CocoaHeads, Unit, Test, Cocoa, XCode