チェックリストの12項目に該当したらTDD脳

Testing Will Challange Your Conventions

角谷 信太郎

s-kakutani@esm.co.jp

(株)永和システムマネジメント

KAKUTANI Shintaro; Eiwa System Management, Inc.; a strong Ruby proponent

XP祭り2007;江戸川区文化センター; 2007-09-01(土)

角谷 信太郎

- ✓ (株)永和システムマネジメント
- ✓ テスト駆動開発者
- ✓ 日本Rubyの会理事
- ✓ http://kakutani.com

IPA未踏ソフトウェア創造事業2006年度下期
千葉PM採択プロジェクト最終成果報告会

- ✓ **Ruby**1.9～これからの**Ruby**～
- ✓ “エンタープライズ”の現場としての**Ruby**
- ✓ 2007.09.07(Fri)
- ✓ http://www.mitou-chiba.org/

『JavaからRubyへ』

マネージャのための実践移行ガイド

Bruce A. Tate 著

角谷信太郎 訳

オライリー・ジャパン 発行

第3刷(fixed 78 bugs)

http://www.amazon.co.jp/o/ASIN/4873113202/kakutani-22

『WEB+DB PRESS』 Vol.40

連載 第2回載ってます:

“アジャイル開発者の習慣

～ acts_as_agile”

角谷信太郎

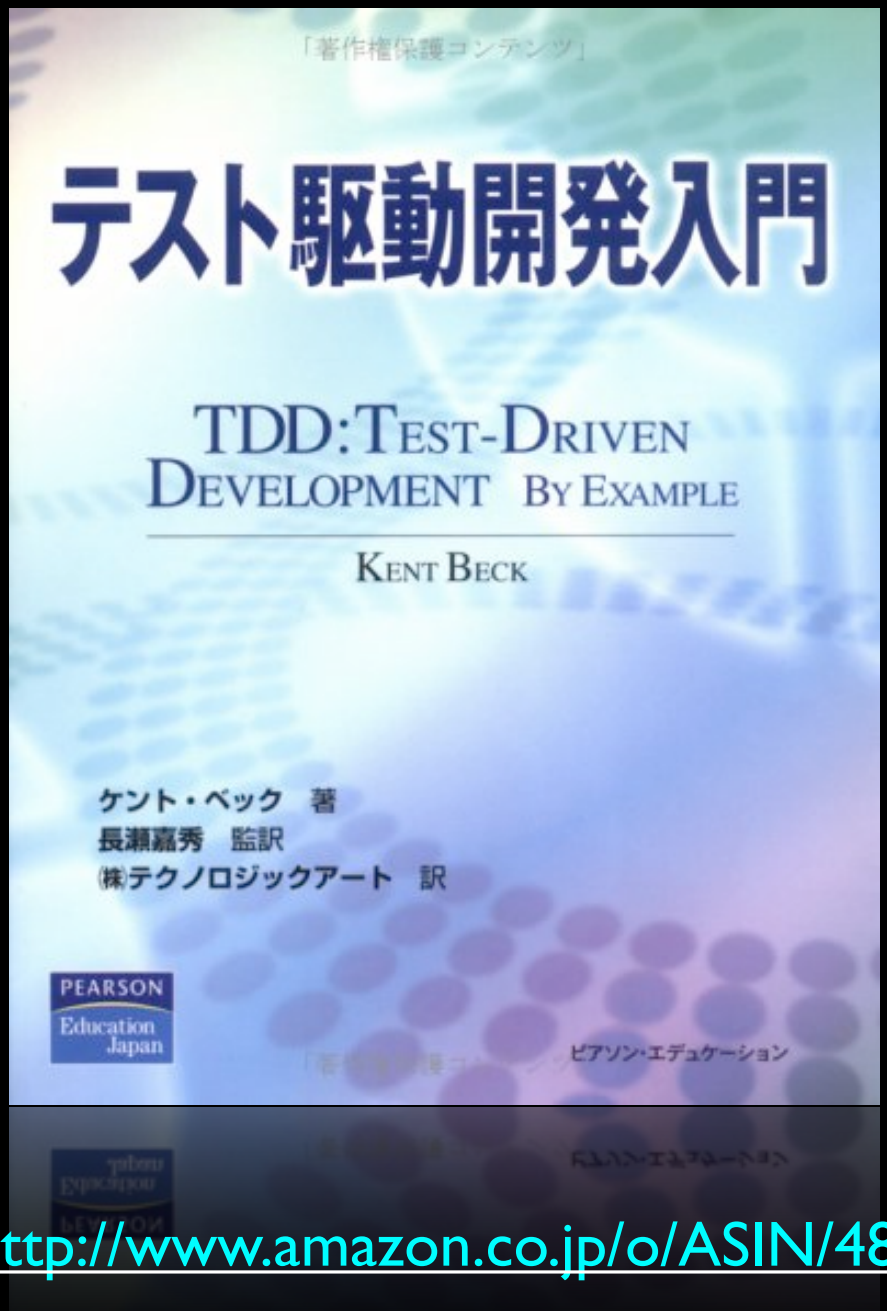
http://www.amazon.co.jp/o/ASIN/477413192X/kakutani-22

よろしく
お願いします

Test Driven Development

テスト駆動開発

テスト駆動開発入門



http://www.amazon.co.jp/o/ASIN/4894717115/kakutani-22

TDDは、

- ✓ テスト技法ではない
- ✓ 分析手法/設計技法であり、
- ✓ 開発のあらゆるアクティビティを構造化するための技法

開発のあらゆるアクティビティを構造化する

TDDは仕事と思考のスタイル

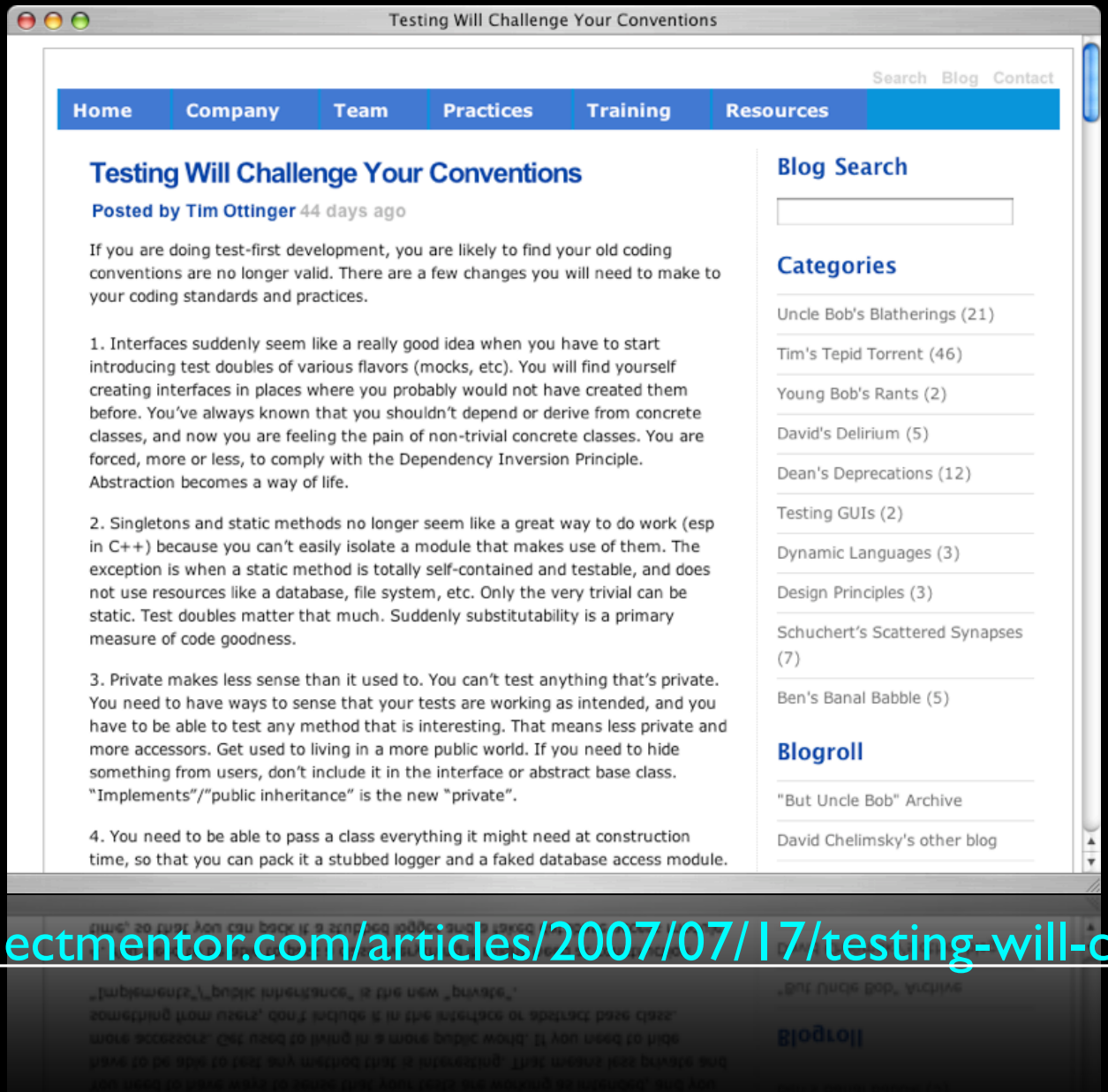
まずは

コードから

今日の

元ネタ

Testing Will Challenge Your Conventions by Tim Ottinger



http://blog.objectmentor.com/articles/2007/07/17/testing-will-challenge-your-conventions

チェックリストの12項目に該当したらTDD脳

1.インターフェイスかわいいよインターフェイス
2.Singletonやstaticメソッドは好みじゃない
3.privateへの思い入れがなくなった
4.コンストラクタ引数にはコラボレータを渡したい
5.小さいメソッドが好き
6.引き継ぎ？ テストコードはどこ？
7.自分の書いたコードの最初のユーザは自分だ
8.最適化を言い出すのはテストを揃えてから
9.テスト全体の実行速度に気を配る
10.依存関係を少なくすることに気を配る
11.「わかりやすい」ことがカッコイイ
12.テスト実行を支援するIDEが良いIDE

ひたすら
紹介してくよ


(1)

1.インターフェイスかわいいよ インターフェイス

- ✓ “インターフェイスに対してプログラミングする”
  - ✓ テストもプログラミングだよ
- ✓ テスト容易性ですよねー
  - ✓ Mock, Stub, DIコンテナ

Rubyに無くね?

- ✓ すべてがインターフェイス
- ✓ Duck Typing
- ✓ インターフェイスは空気
- ✓ カッコいいMock/Stubライブラリたち


(2)

2.Singletonやstaticメソッドは好みじゃない

- ✓ だってテストしづらいもん
- ✓ ほんとに必要かな?
- ✓ テストできるなら使うけど
- ✓ bliki:“staticの置き換え”
  - ✓ http://capsctrl.que.jp/kdmsnr/wiki/bliki/?StaticSubstitution


(3)

3. privateへの思い入れがなくなった

- ✓ だってテストしづらいもん
- ✓ インターフェイスが”public”
  - ✓ “published”
- ✓ 実装クラスが”private”
- ✓ テストの資産価値との関連

(4)

4.コンストラクタ引数にはコラボレータを渡したい

- ✓ 協調するオブジェクトを自分で生成しない
- ✓ だってテストしづらいもん
- ✓ 単一責務の原則(SRP)
- ✓ DIコンテナ


(5)

5.小さいメソッドが好き

- ✓ だってテストしづらいもん
- ✓ 10行を越えると大きいよね
- ✓ メソッドが小さいと、それぞれのテストも小さくなる
  - ✓ 数は増えちゃうかもしれないけど
  - ✓ 極端に短いとかはまた別の話


(6)

6.引き継ぎ?

テストコードはどこ?

- ✓ コメントよりもテストコード
- ✓ テスト容易性にはテストコードの読みやすさも含まれるよ
- ✓ ドキュメントのようなコードを書きたい


(7)

7.自分の書いたコードの最初のユーザは自分

- ✓ 自分の書いたコードのテストを書くのは自分だもん
- ✓ 自分がテストしづらいコードは他の人にも使いづらいはず
- ✓ 使いやすいAPIを設計したい


(8)

8.最適化を言い出すのは テストを揃えてから

- ✓ 速さとテスト容易性とでは、まずテスト容易性を選ぶ
- ✓ 最適化の第1原則: 最適化するな
- ✓ 最適化の第2原則: まだ最適化するな
- ✓ プロファイリング


(9)

9.テスト全体の実行速度に気を配る

- ✓ 頻繁に実行したいじゃない?
- ✓ 実行速度の目安:
  - ✓ Red/Green/Refactorのサイクルで1秒
  - ✓ ユニットテスト全体で20〜30秒
  - ✓ 「10分ビルド」
- ✓ 遅いテストは別実行(それCIで)

(10)

10.依存関係を少なくすることに気を配る

- ✓ だってテストしづらいもん
- ✓ 小さなクラスをたくさん定義することを躊躇しない
- ✓ 協調を担当するクラスと、実装を担当するクラス

(11)

11. “わかりやすい”ことがカッコイイ

- ✓ “わかりやすさ”は“巧妙さ”に200倍勝る
- ✓ だってテストしづらいもん
- ✓ リファクタリングしやすい
- ✓ 意図が明瞭なテストコード

(12)

12. テスト実行を支援するIDEが良いIDE

- ✓ 開発とはテストの実行である
- ✓ モダンIDE
  - ✓ Eclipse,NetBeans,IDEA
  - ✓ Visual Studioも?
- ✓ Emacs, Vim

まとめ

- ✓ TDDは仕事と思考のスタイル
- ✓ TDD脳チェックリスト12項目
- ✓ 短いので原文も読んでね
- ✓ 営業自重

ご清聴
ありがとう
ございました

Happy Testing!