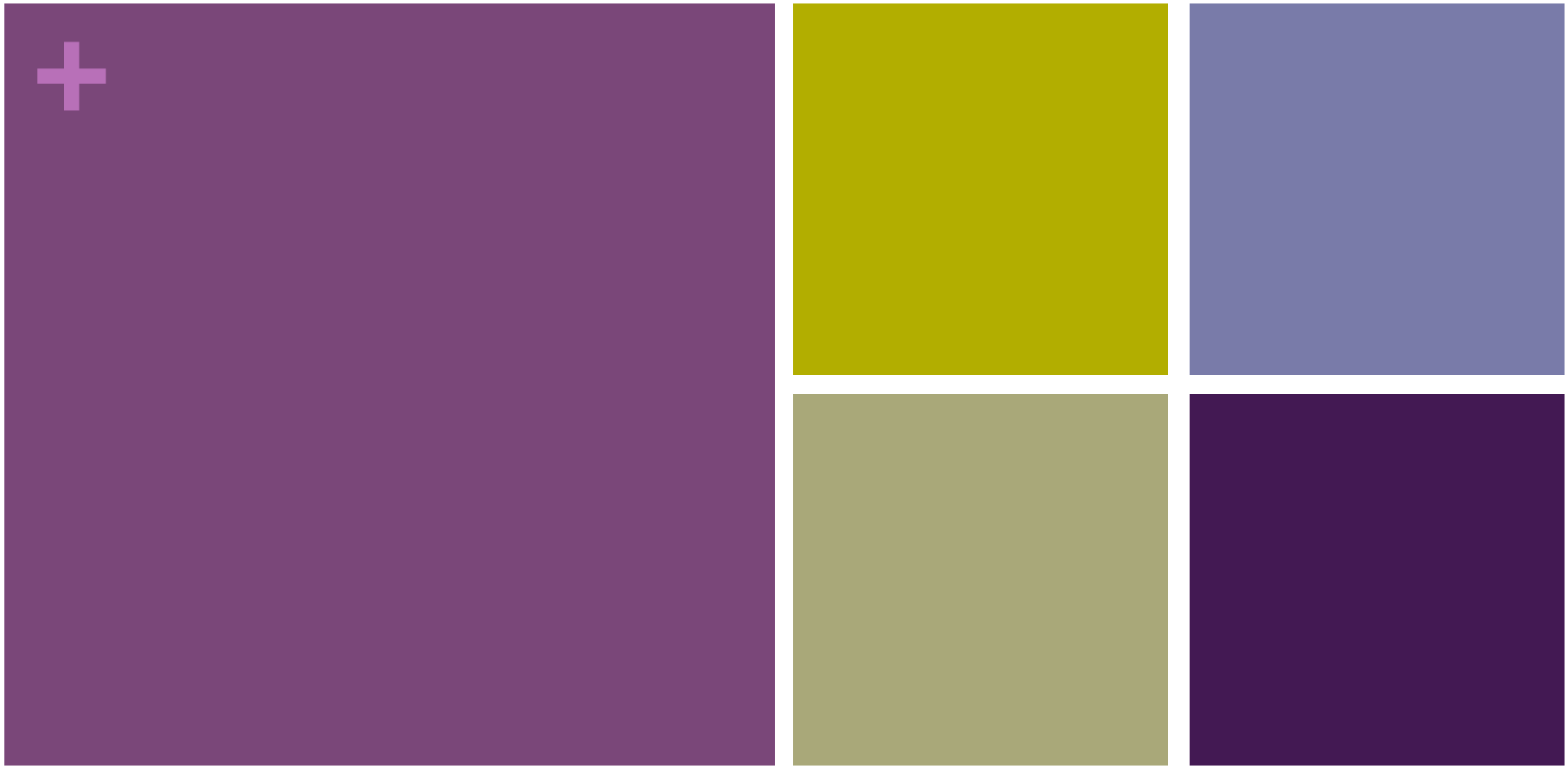
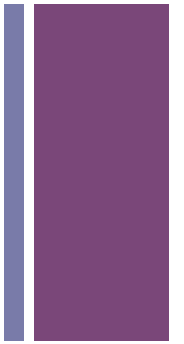


JaSST'15 Tokyo
テストコードクリニック

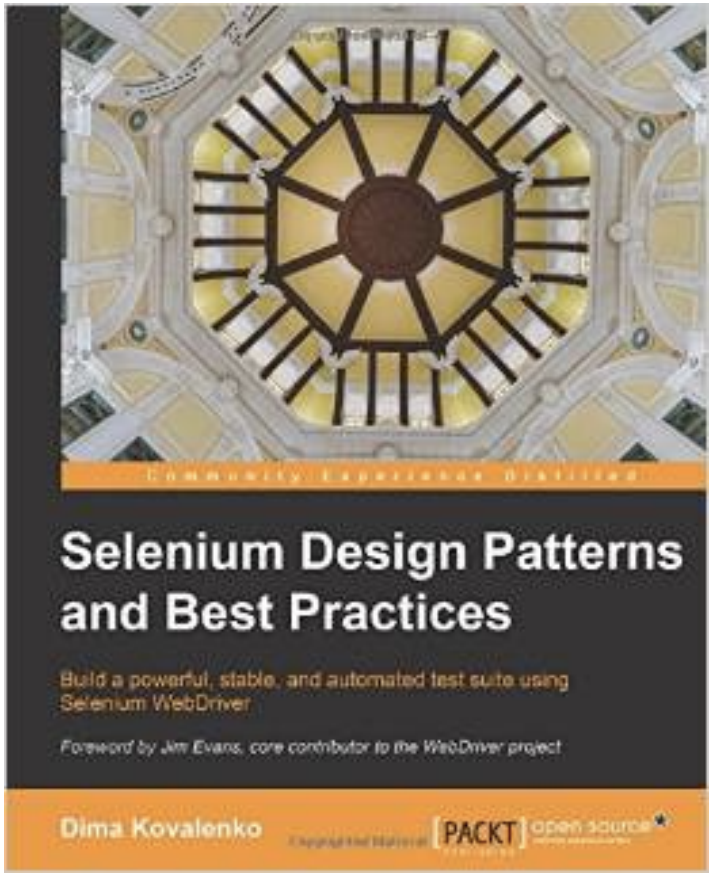
2015/02/20 太田健一郎
テスト自動化研究会　株式会社SHIFT

アジェンダ



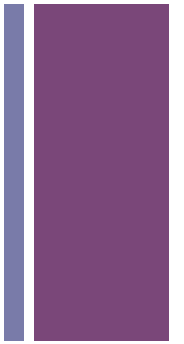
- Selenium Design Patterns and Best Practices
- あるある残念なテストコード
- 残念なテストコードを生み出すデザインパターン
- テストコードを改善するデザインパターン
- 更なる改善のために

Selenium Design Patterns and Best Practices



- Seleniumを使ったGUIテストのデザインパターンとベストプラクティスを解説した書籍です
- 残念なテストコードから始めて各種のデザインパターンとベストプラクティスを適用しながら品質の高いテストコードへと変えていく手法が見事です
- 書籍のコードはRubyですが、Web上にJavaリソースもあります
- 本セッションはそのJavaソースを使って最初のいくつかのデザインパターンを解説します
- Kindleなら¥1,424 !!
- 2015/08に太田&玉川監訳でオライリーさんから翻訳書が出版 !!

あるある残念なテストコード



- IntelliJ IDEAにて解説します
- このテストコードの問題点を会場からご指摘お願いします

残念なテストコードを生み出すデザインパターン

- Record and playback pattern
- Spaghetti pattern
- Chain Linked pattern
- Big Ball of Mud pattern

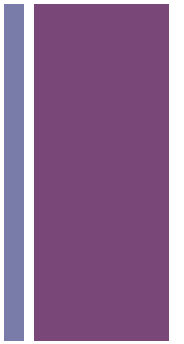
Record and playback pattern

- 概要
  - キャプチャ＆リプレイ・ツールを使って記録したスクリプトをそのまま利用するパターンです
  - 再実行のための一部定数の変数化や実行前の置換は実施します
- 利点
  - テストコードを素早く作成できます
  - プログラミングの経験が不要です
  - 画面要素の検索が不要です
- 欠点
  - 生成されるロケーターが分かりにくいです
  - テストコードの柔軟性がありません
  - テストデータが固定値で記録されます
  - 生成されるテストコードが構造化されていません
  - 生成されるコードが重複しています

Spaghetti pattern

- 概要
  - テスト自動化アーキテクチャと設計がありません。そのため、
    - テストコードが構造化されていません
    - テストケース依存関係がある場合があります
    - テストケースを個別に実行できない場合があります
  - Record and playback patternを使って生成されたテストコードの多くがこのパターンで実装されています
- 利点
  - 早くテスト自動化が始められます
  - テストコード自体が小さくなる可能性があります
  - スモークテストには向いているかもしれません
- 欠点
  - このデザインパターンは長期的なテスト自動化にはアンチパターンです
  - テストコード間の結合度が高すぎます
  - テストを並行実行できません
  - 最初の方のテストで失敗するとテストスィート全体が失敗します
  - 他のテストの失敗が別のテストの失敗に繋がります

Chain Linked pattern



- 概要
  - Spaghetti patternを改善し、テストコード自体はDRY testing patternを始めとする各種のパターンで構造化したものです
  - ただし、依然としてテストケース間に依存関係があります
  - 依存関係に伴うSpaghetti patternが抱える欠点を引き継ぎます

Big Ball of Mud pattern

- 概要
  - Spaghetti patternにHermetic test patternを適用し、テストコード自体は独立しています
    - このため、個別実行、並列実行、ランダム実行は可能です
  - しかし、テストコード全体としては構造化されておらず、テスト自動化アーキテクチャが不在のため、以下の問題が発生します
    - 部品となるテストコードやテストデータの変更によって、意図しないテストケースの失敗が発生します
    - 共有部品が保守困難なGod Class化する可能性があります

テストコードを改善するデザインパターン

- DRY testing pattern
- Hermetic test pattern
- Random order principle

DRY testing pattern

- 概要
  - テストコードにおいてDon't Repeat Yourselfを実践するパターンです
  - 必要なコードは一つの場所に正しくまとめます
- 利点
  - テストコードが部品で構築されるようになります
  - テストコードの重複が減ります
  - テストコードの修正が早くできるようになります
- 欠点
  - テストコードのプロジェクト構造が複雑になります
  - 適切なIDEのサポートが必須です
  - DRYを維持し続けるのは大変です
  - メンバー全員にプログラミングスキルが必要です

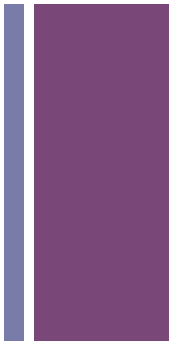
Hermetic test pattern

- 概要
  - テストケース間の依存関係を廃するパターンです
  - 直接管理ができない外部への依存関係を廃します
- 利点
  - 各テストケースはまっさらな状態で開始します
  - テストがまとめて失敗することがなくなります
  - テストコードが部品で構築されるようになります
  - テストのランダム実行が可能になります
  - テストの並列実行が可能になります
- 欠点
  - 事前に設計と部品の開発が必要になります
  - テストの実行時間が増加します
  - テスト実行時のリソースの消費が増加します

Random order principle

- 概要
  - テストケースを常にランダムに実行します
  - パターンと言うよりは原則といえます
- 利点
  - テストケース間の暗黙的な依存関係を防ぎます
  - 他のテストケースの結果が別のテストケースに影響を与えるのを防ぎます
  - テスティングフレームワークによっては標準で組み込まれています
- 欠点
  - テストコードに大量の修正が必要です
  - テストケースの失敗を追うのが困難です
  - チームのフラストレーションがたまる可能性が高いです
  - ランダム実行をサポートしていないテスティングフレームワークが多いです

更なる改善のために



- Data-driven testing
  - default values pattern
- Stabilizing the Tests
  - Action Wrapper pattern
  - Back Hole Proxy pattern
- Testing the Behavior
  - write once, test everywhere pattern
- Page Objects Pattern
  - test tool independence pattern
- Growing the Test Suite