

# TDDで(自分の)開発がどう変わったか

【Qiita Bash】 Qiita Tech Festa お疲れ様会



Ryosuke Tomita (sigma)

# 自己紹介

名前: 富田涼介 (sigma)

職業: セキュリティエンジニア NRIセキュアテクノロジーズに所属

個人的ニュース: 聽くエンジニアtypeに出演。Podcast聴いてください!

一言: そろそろ常連になってきた? 運営様、スポンサー様ありがとうございます。



# 今日話したいこと

- TDDとはなにか?
- TDD導入のきっかけ
- 実際の導入時の壁
- AIとTDD

# TDDとは？（超簡単版）

テスト駆動開発 = Test Driven Development

1. テストを書く → 失敗を確認
2. 最小限の実装 → テストを通す
3. リファクタリング → 設計改善
4. 繰り返し 

※ Kent Beckの著書「テスト駆動開発」はおすすめ

# TDD導入のきっかけ 「おらこんな環境いやだ～TDDするだ～」

## 自動テストが欲しい



手動テストから解放されたい



デプロイしてから動作確認をするのは時間がかかる

# | 実際の導入時の壁

1

開発環境の整備

2

どんなテストを書いたらいいかわからない

# 開発環境の整備

テストを実行するために労力が必要だとTDDを始めるのが億劫になる

現在の開発環境を段階的に整備した。

## レベル1: デプロイの自動化(CI/CD)

- ・テスト実行環境を作るよりは手がつけやすい

## レベル2: ローカルの開発環境構築手順の整備

- ・Docker化する
- ・READMEに動く手順を書く。情報の集約

## レベル3: テスト実行環境の整備

# どんなテストを書いたらいい?

ある程度の基礎を勉強したら、手を動かして直感を磨くしかなさそうな気がしている。

- ✓ 古典派的アプローチ: 振る舞いをテストする。テストが壊れにくくなる
- ✓ 粗結合なコードをつくる
  - DDD
- ✓ TDDですべてのテストケースを網羅しようとしている
  - リリース前テストは別で作ったほうが良さそう

 単体テストの考え方/使い方(書籍)はおすすめ

# TDD導入で得られた4つの効果 ✨

## 1. 集中力UP

TDDのフローに沿うことで今やるべきことに集中できる

## 2. テストのある安心感

テストがあることで勇気を持って変更できる

## 3. 設計意識の向上

テストしやすいコード = 良い設計を意識するように  
コードをいきなり書かなくなった。

## 4. 開発環境をエンハンスするモチベーション

TDDを進める中で、開発環境をより良くしようとする意欲が湧いてくる

# AIとTDD(AIと並走する文脈)

例: GitHub Copilot

TDDのテストを一つ書いて一つ通すというやり方はAIと並走しやすい

- ✓ AIと一緒にテストコードを書く(文法とか曖昧でも書きやすい)
- ✓ AIと一緒にテストコードを通す
  - テストが実施できるまではAIにまかせてもいいかも
  - テストが通るように実装すれば最低限の品質は保証される
- ✓ リファクタリングのアイデアを試しやすい

# AIとTDD(AIに全部委託する文脈)

例: Claude Code + Kiro(AWS)

要求定義書(requirements.md)

設計書(design.md)

タスクリスト(tasks.md)

3で作成したタスクリストをベースに「t-wadaさんのTDDで実装して」とClaude Codeにお願いすると割といい感じで作ってくれる。

単純に仕様書駆動開発するよりは、タスクをもとにテストを作るほうが良さそう。(少なくとも人間の負担は少ない)

これが最良の方法なのかはわからない

# まとめ

- 1 TDDを導入したことでエンジニアとして視野が広がるきっかけになった。
- 2 少なくともAIにはテストというガードレールは必要だとは思う。TDDが最良とは思はないので、手を動かしながらいろいろ模索していきたい。

# Thanks

今日話しきれなかった部分についてのQiita記事と本日の発表資料のリンクです。



Qiita記事

TDDに関する詳細記事



今日の登壇資料

スライド資料のダウンロード



プロフィールページ

お問い合わせはこちら

※発言はすべて個人の見解であり、所属組織を代表するものではありません