
TDD勉強会
Vol.01

自己紹介

- 芝浦工業大学卒(応用化学科)
- WEB制作会社にバックエンドエンジニアとして就職
- 現フリーランスエンジニア

- SQL, PHP, JS, HTML, CSS
- 設計・開発とノンストップでできるのが強み

- シーシャがだいすき

- Github↓↓↓
- https://github.com/kenta0629

TDDとは

- テスト駆動開発(**T**est-**D**riven **D**evelopment)

- 最小限のコード(主にメソッド)に対してバグが起こらないようにテストコードを作成する
- テストコードで単体テストを行うようなイメージ

- 基本的な開発サイクル
- 1) 失敗するテストを書く
- 2) できる限り早く、テストに通るような最小限のコードを書く
- 3) コードの重複を除去する(リファクタリング)

- **実際のPJだと開発コストの関係でコード作成→テストコードでテストのフローになりやすい**

メリット

- テストパターンを洗い出すことでメソッドの役割が明確になる
- 単体テスト時のコストが減る(仕様書作成も含めて)
- 要件を再度見直す機会が増える
- 追加要望・追加機能の予測につながる
- 自動でテストを気軽に試すことができる
- テスト済みであると安心して実コードを使える
- メソッドの役割を意識してコードを書く練習になる
- ファットController・Modelの解消につながる

- **テストパターンを事前に洗い出しておくことで、ケアレスミスや想定外な要件パターンを格段に減らせる！**

デメリット

- 工数を確保する時間がない
- テストできる状態を整えるコストがある
- テストデータ、テストコードの管理がタスクとして増える
- テストパターンの洗い出しに時間がかかる
- PHP-Unitの構文に慣れるコストがかかる
- 仕様が変わるとテストコードも変えるコストがかかる
- Model, Controller, Service, Templateなど、テストする箇所が多い

- **人的コストが増える点が一番のデメリット...**

PHPUnit

- 単体テストを行うためのフレームワーク
- PHPのフレームワークならだいたい使える ※Laravel, Cakephp, Zend Frameworkは検証済

- アノテーション(@test, @dataProvider)
- アサーション(assertSame, assertBool)

- **PHPUnit マニュアル↓↓↓**
- **https://phpunit.readthedocs.io/ja/latest/**

テスト手順(Laravel)

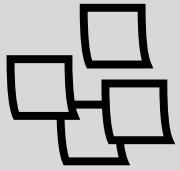
- 1) テストデータの作成
- 2) テストしたいメソッドの実装
- 3) テストパターンを用意してUnitテスト実行
- 4) テストで失敗した場合は期待値に近づくようにメソッドを修正
- 5) テストがOKになる(green)まで3,4を継続

デモでテストの仕方を見てみよう！

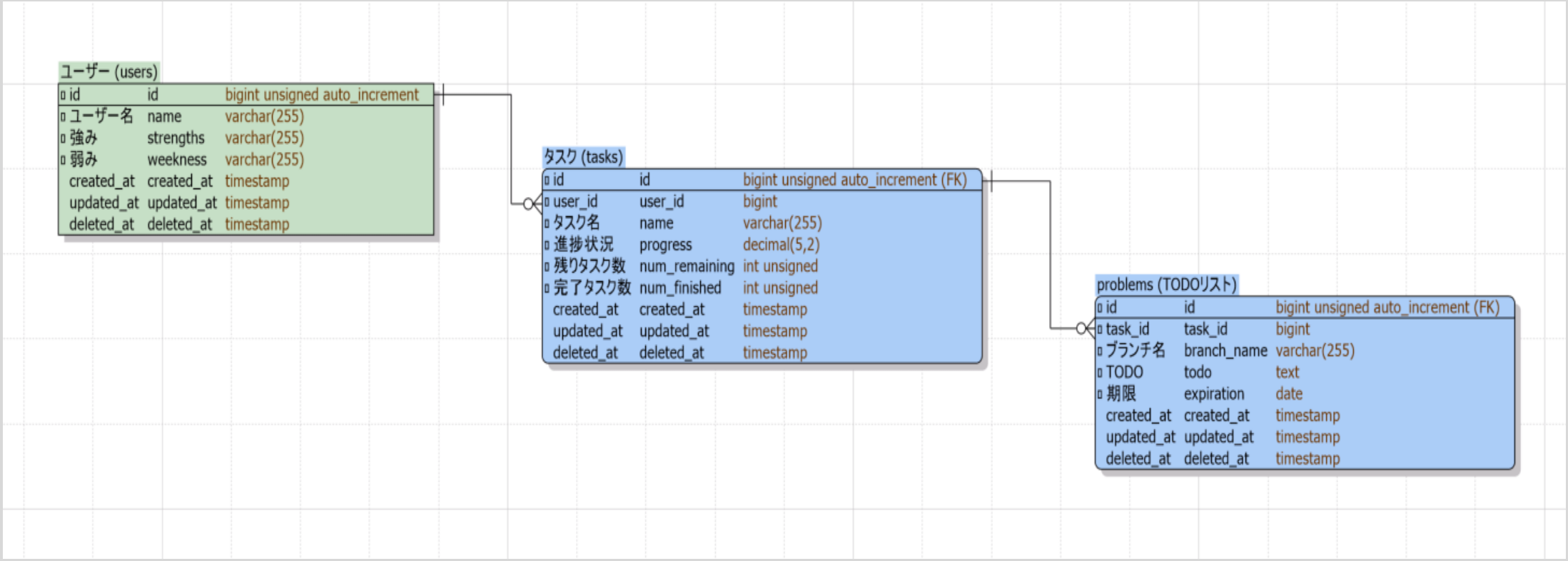
デモシステム要件

- **TODOリスト管理システム**

- ユーザーが抱えるタスクのTODOを作成・管理して進捗状況を管理していくシステム
- タスクの進捗状況を%、完了したTODOリスト数を管理
- TODOリストデータにはそれぞれ期限が設けられており、それを過ぎるとメールが届く
- ユーザーはタスクデータを作成する際にTODOリストデータも合わせて作成する
- 完了したTODOリストデータは論理削除して、タスクの進捗を進行させる



デモDB構造


ユーザー (users)
id id bigint unsigned auto_increment
ユーザー名 name varchar(255)
強み strengths varchar(255)
弱み weekness varchar(255)
created_at created_at timestamp
updated_at updated_at timestamp
deleted_at deleted_at timestamp
タスク (tasks)
id id bigint unsigned auto_increment (FK)
user_id user_id bigint
タスク名 name varchar(255)
進捗状況 progress decimal(5,2)
残りタスク数 num_remaining int unsigned
完了タスク数 num_finished int unsigned
created_at created_at timestamp
updated_at updated_at timestamp
deleted_at deleted_at timestamp
problems (TODOリスト)
id id bigint unsigned auto_increment (FK)
task_id task_id bigint
ブランチ名 branch_name varchar(255)
TODO todo text
期限 expiration date
created_at created_at timestamp
updated_at updated_at timestamp
deleted_at deleted_at timestamp

実際にテストの仕方を見てみよう！

まとめ

- 要件の穴を見つけるのは早いが、人的コストがかかる
- テスト結果はアサーションで判定する
- 要件の穴を見つけるのは早いが、人的コストがかかる
- PJでテストコードを書くなら、Modelのみなどのように絞って行う
- メソッドの役割を意識してコードを書く練習にはなる！