

FaaO

ドメイン駆動設計で作るGitHub Issue Client

自己紹介

- Name : **azu**
- Twitter : @azu_re
- Website: [Web scratch](#), [JSer.info](#)



過去に作ったやつ

- azu/GithubReader: Github Notifications Client for OS X
- azu/github-reader: [node-webkit] GitHub client app - Viewer for Notifications and News Feed.
- azu/github-issue-teev: [NW.js] GitHub Issue Manager(Viewer)

2014-04-29T13:12:09Z
MartinKolarik
MartinKolarik
commented on
ractivejs/ractive#644

@jamesfisher Yeah that makes sense, but Ractive's isn't a library for input validation. If you need just a simple validation, you can use Ractive's features (observers) to create some validation rules, if you need something more complex, you are free to use any other library/plugin intended for that purpose.

2014-04-29T13:10:41Z
MindTooth
MindTooth
commented on
mllssohn/node-webkit-builder#9

Well, this is the current file:
<https://gist.github.com/MindTooth/11382714> Also added the "package.json" file within the same Gist. [screen shot 2014-04-29 at 15:07:50] (<https://cloud.githubusercontent.com/assets/35828/2829159/57375d38-cf9f-11e3-864b-c0220782b4ba.png>) Just

2014-04-29T13:08:41Z
hpique
hpique commented on
steipete/PSPDFTextView#9

Awesome. Thanks!

2014-04-29T13:07:39Z
reubano
reubano commented on
travis-ci/travis.rb#41

This is what worked for me
`travis-enc.sh` ``bash
#!/usr/bin/env sh -u
ENC_FILE='envs.yml' ENVS=\$1
USER=\$2 PROJECT=\$3
encout_file() { secret=\$1 file=\$2

GitHub Issue Teev

file:///Users/azu/.ghq/github.com/azu/github-issue-teev/index.html

Assigned issues

Explore Gist Blog Help

azu + ⌂ ⌂ ⌂ ⌂

My Issues

ECMAScript: BoundNamesとはなにか
<https://github.com/azu/azu/issues/52>

Repo Issues

ECMAScript: BoundNamesとはなにか
<https://github.com/azu/azu/issues/52>

更新履歴を追う方法
<https://github.com/efcl/efcl.github.io/issues/87>

testemのSourceMap対応
<https://github.com/azu/azu/issues/51>

Custom Elementは実用的なのかどうかの検証
<https://github.com/azu/azu/issues/32>

Fav Repository issues

MailplaneのようなGitHub Issue Todoクライアント #48

azu opened this issue 20 days ago · 4 comments

azu commented 20 days ago

Owner

Labels
None yet

Milestone
No milestone

Assignee
azu

Notifications
Unsubscribe

You're receiving notifications because you were assigned.

1 participant

Lock issue

アイデア募集中

- エディタをCodeMirror的なものへ
 - テキストエリアをMarkdownエディタに切り替えるGreasemonkeyを書いた | Web Scratch
- assign yourselfの自動化
- []の一覧を可視化
- インクリメンタルなIssue一覧のフィルタリング

アプリならでは常駐的な通知って何か必要なかなー

azu self-assigned this 20 days ago

azu commented 20 days ago

Owner

自分へassignすると<https://tacoapp.com/>にTodoとして出るという感じの仕組みがあるので、これ使って

Sub tasks

★ MailplaneのようなGitHub Issue Todoクライアント

- エディタをCodeMirror的なものへ
- assign yourselfの自動化
- []の一覧を可視化
- インクリメンタルなIssue一覧のフィルタリング
- TODO

FaaO

Quick Issue

Accounts

@azu

Created

Assigned

Mentioned

Review-
Requested

Queries

Faaoo

almin

immutable-
array-prototype

textlint

azu@todo

filter word - state:open

! almin: circular issue UseCase <-> UseCase...
`\$ depcruise --validate dependency-cruiser.json src...
almin/almin#220 updated 2017-07-03 20:07 by azu 0

! Main: always show content and searchbar
SearchBar and Content will be shown when the items i...
Status: Proposal Type: Bug
azu/faao#61 updated 2017-07-03 14:07 by azu 0

! almin: Transaction of UseCase
It related with Unit of work #186 Almin's unit of work ...
Status: Proposal
almin/almin#219 updated 2017-07-03 09:07 by azu 0

! Sync with gist
We want to add sync feature between A PC <-> B PC ...
Status: Proposal
azu/faao#59 updated 2017-07-02 18:07 by azu 1

! Release
First Release - [] Electron app - [] Mobile #51 - ...
Type: Maintenance
azu/faao#58 updated 2017-07-02 00:07 by azu 0

! Wide screen
![image](https://user-images.githubusercontent.com/1...
Priority: Medium Status: Proposal
azu/faao#57 updated 2017-07-03 14:07 by azu 0

! fromJSON shoule be catch

filter word - state:open

! almin: circular issue UseCase <-> UseCase...
`\$ depcruise --validate dependency-cruiser.json src...
almin/almin#220 updated 2017-07-03 20:07 by azu 0

! Main: always show content and searchbar
SearchBar and Content will be shown when the items i...
Status: Proposal Type: Bug
azu/faao#61 updated 2017-07-03 14:07 by azu 0

! almin: Transaction of UseCase
It related with Unit of work #186 Almin's unit of work ...
Status: Proposal
almin/almin#219 updated 2017-07-03 09:07 by azu 0

! Sync with gist
We want to add sync feature between A PC <-> B PC ...
Status: Proposal
azu/faao#59 updated 2017-07-02 18:07 by azu 1

! Release
First Release - [] Electron app - [] Mobile #51 - ...
Type: Maintenance
azu/faao#58 updated 2017-07-02 00:07 by azu 0

! Wide screen
![image](https://user-images.githubusercontent.com/1...
Priority: Medium Status: Proposal
azu/faao#57 updated 2017-07-03 14:07 by azu 0

! fromJSON shoule be catch

https://github.com/

GitHub

Search GitHub

Pull requests Issues Marketplace Gist

azu

3 minutes ago BridgeAR commented on pull request nodejs/node#13755
I guess in that case the the S should also be renamed but I can't think of what it stands for right now. Does it stand for Script ?

5 minutes ago bobheath33435 commented on issue ReactiveX/rxjs#2539
@bmayan My comments were and are intended to be constructive. I understand how inexperienced developers like yourself may not understand that promo...

6 minutes ago gaearon starred uanders/react-redux-cheatsheet 6 minutes ago

6 minutes ago simon04 commented on issue localForage/localForage#635
For reference, Firefox goes with a storage attribute on the option object to the open() function: <https://developer.mozilla.org/en-US/docs/Web/API/>

7 minutes ago olegdunkan commented on issue Microsoft/TypeScript#16898
You have constraint in Mixin function <T extends Constructor<any> therefore by default type of this has string index signature with any type / TKey

Contrib Friday Join our contribu

Repositories

jser/jser.ir jser/realti jser/sourc almin/almi asciidwang Show

Your repository

Find a repository

All Public

Faoao - Feature

- Support Modern browser/mobile/Electron(recommended)
- Support GitHub.com and GitHub Enterprise(GHE)
- Search Issue/Pull Request
 - Search Syntax is same with GitHub Search
- Mixed the result of search
 - e.g.) You can see the results of Created, assigned, mentioned as a single result
 - e.g.) You can see the results of repo:azu/todo on github.com and repo:azu-ghe/todo on GHE as a single result
- Support GitHub User Activity
- Quick to create issue
- Import/Export profile data

目的

- OOSでGitHub Issueをちゃんと扱うものがない
- 技術的目的
 - Admin + TypeScript + DDD ドメイン駆動設計 である程度の規模のアプリケーションを作りたかった

規模感(2017-07-03現在)

⚡ cloc src

212 text files.
208 unique files.
4 files ignored.

github.com/AlDanial/cloc v 1.72 T=1.31 s (159.3 files/s, 12962.9 lines/s)

Language	files	blank	comment	code
TypeScript	165	859	478	8487
JSON	7	1	0	6189
CSS	34	120	60	728
Markdown	2	3	0	6
SUM:	208	983	538	15410

作戦

- 「ちゃんと考へてちゃんとやる」
- 技術的ショーケースとしての意味合いを持つ
 - ちゃんとモデリングする
 - ちゃんとテストを書く
 - ちゃんとドキュメントを作る

DDD

ちゃんとモデリングモデルをやる

モデル ここでいうモデルはEntityとかValue Objectを含めたドメイン上のモデルクラス

クライアントサイドDDD

- [fao/domain.md at master · azu/fao](#)
- ドメインモデルの寿命が長い
 - 特にこういうクライアントアプリはずっと立ち上げっぱなし
- サーバ側の概念とクライアント側の概念は一致しないことがある
 - サーバ(GitHub)的にアカウントに対してGitHub APIのトークンが複数紐づく
 - クライアントからはTokenがあり、そのTokenに紐づくアカウントがいるように見える
 - Tokenがなければアカウントは分からず、アカウントだけ分かってもトークンがないと何もできない

モデリング

- AppUser: アプリケーションのユーザー
- GitHubSetting: TokenやAPI hostなどを含んだセッション情報
- GitHubUser: GitHubのAPIを叩いた結果取得できるGitHubユーザー情報

多くの処理(ユースケース)は

AppUserが**GitHubSetting**を使って ~~ する

のようになることが分かってくる

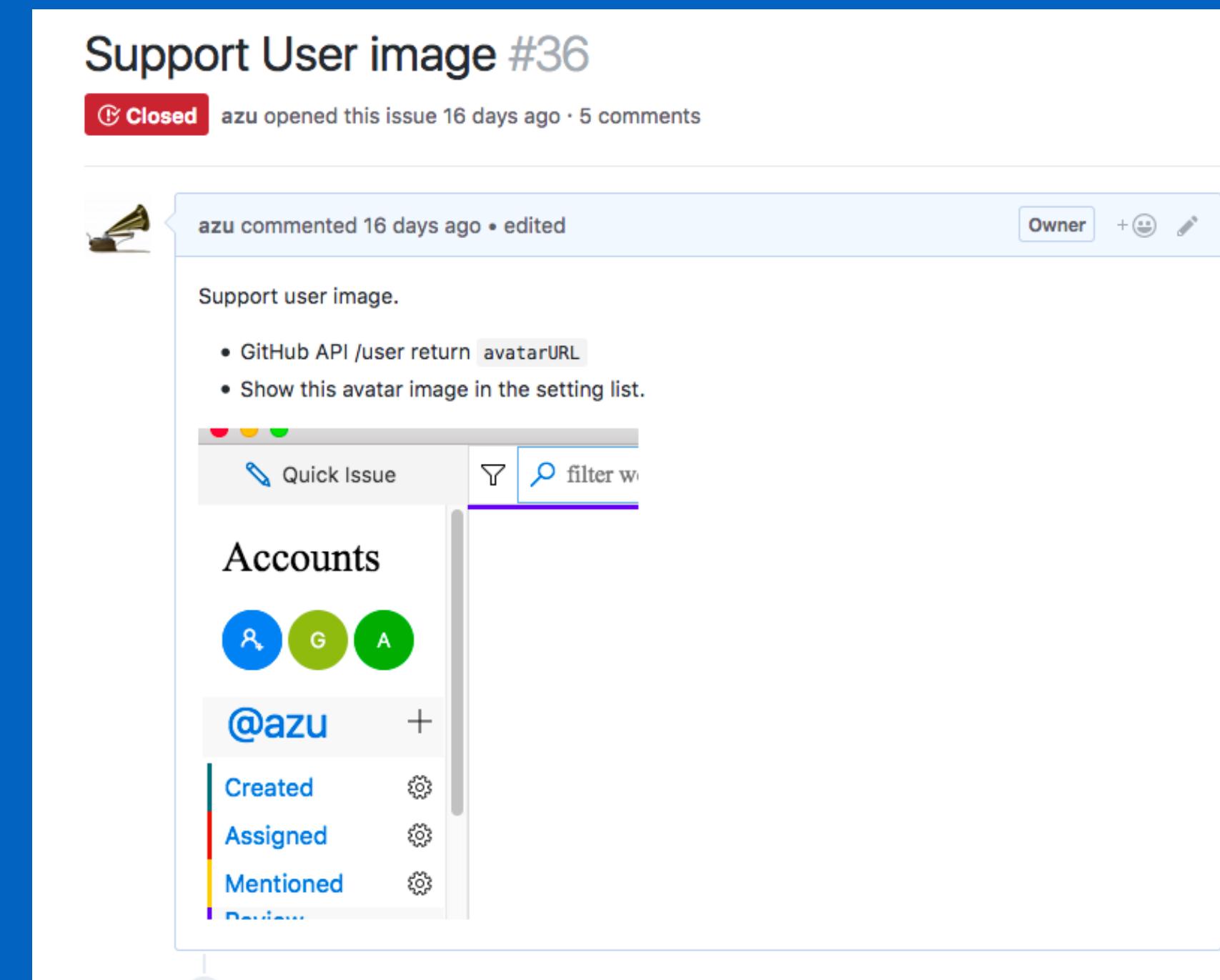
遠回りのモデリング

- 実際モデリングをしっかりやると進みが遅く感じる
 - 一つのモデルが大きくなりすぎないように気を配ったり
- 遠回りしてよかつた場合もある
 - 安易なUI起因の値がドメインに流れてくるのを防げる

遠回りの例

- GitHubSetting(Account)にアイコン画像を設定したいというIssue
- 安直にやるなら GitHubSetting へ avatarImageURL などを追加すれば終わり

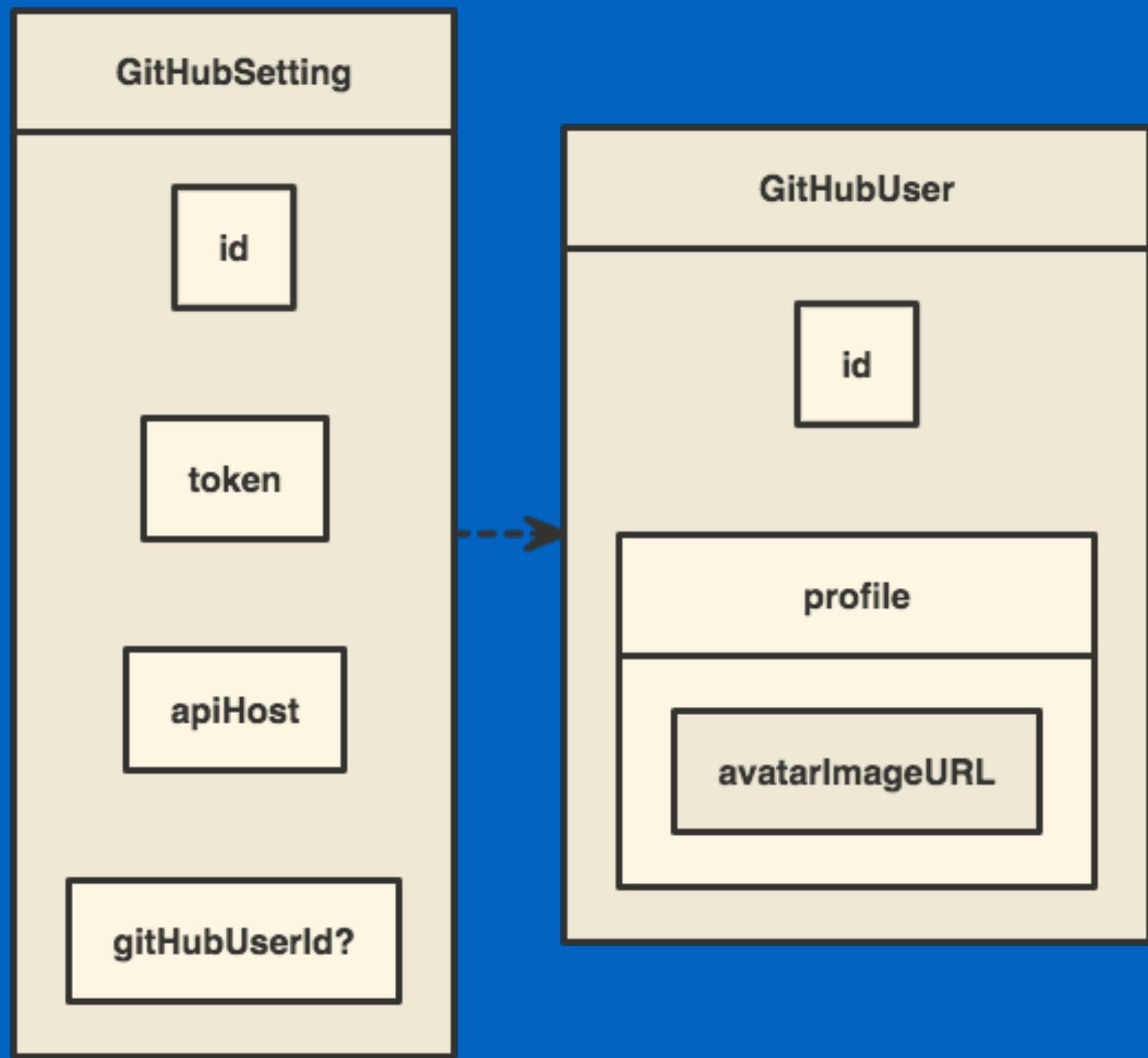
```
interface GitHubSetting {  
    id: Identifier<GitHubSetting>;  
    token: string;  
    apiHost: string;  
    webHost: string;  
    // ADD?  
    avatarImageURL?: string;  
}
```



遠回りの例 -> GitHubUser

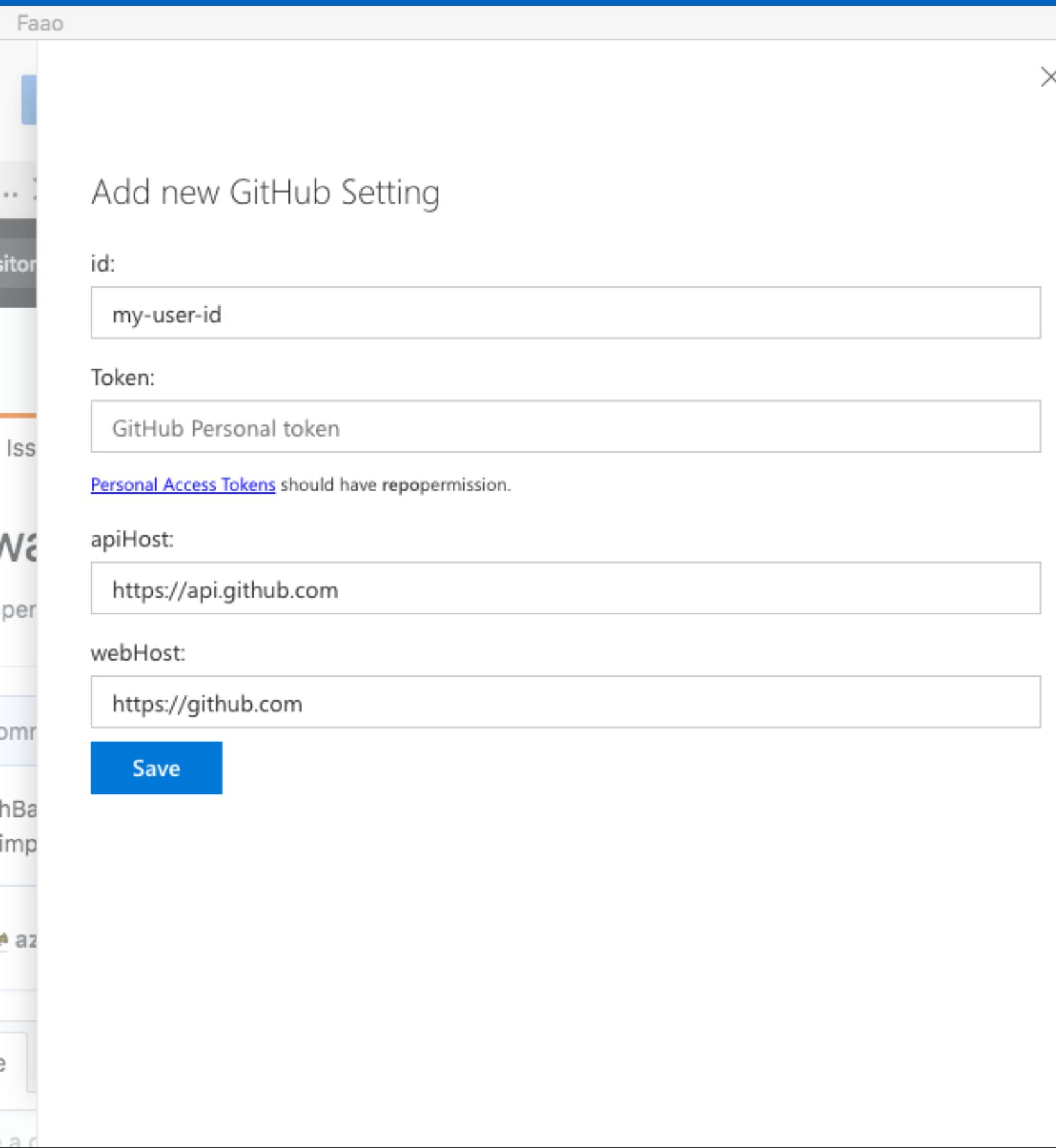
- 後回しにしていて、GitHubUserのActivityを表示したいと思った
- このときに、GitHubUserというモデルが必要で、avatarImageURLはこのGitHubUserのprofileに属するデータであると分かった
- 結果的にGitHubSettingに追加されたのはGitHubUserへのRelationship Id

```
interface GitHubSetting {  
    id: Identifier<GitHubSetting>;  
    token: string;  
    apiHost: string;  
    webHost: string;  
    // Relationship  
    gitHubUserId?: Identifier<GitHubUser>;  
}
```



遠回りのモデリング

- GitHubSettingとGitHubUserは想定するライフサイクルが異なった
- GitHubSettingで入力されたTokenを使って、/user APIを叩いて GitHubUserを作る
- 異なるライフサイクルを一つのモデルにまとめると破綻する未来が見えていた



ドメインモデル -> 永続化

Hard repository

このアプリの永続化してる部分

Faaoo

Quick Issue

Accounts

@azu

Created
Assigned
Mentioned
Review-
Requested

Queries

Faaoo
almin
immutable-
array-prototype
textlint
azu@todo

filter word - state:open

test: assert layer violation

Use [sverweij/dependency-cruiser: Validate and visualiz...
azu/faao#62 updated 2017-07-03 20:07 by azu 0

almin: circular issue UseCase <-> UseCase...

``` \$ depcruise --validate .dependency-cruiser.json src...  
almin/almin#220 updated 2017-07-03 20:07 by azu 0

Main: always show content and searchbar

SearchBar and Content will be shown when the items i...  
Status: Proposal Type: Bug  
azu/faao#61 updated 2017-07-03 14:07 by azu 0

almin: Transaction of UseCase

It related with Unit of work #186 Almin's unit of work ...  
Status: Proposal  
almin/almin#219 updated 2017-07-03 09:07 by azu 0

docs(UseCase): add UseCase architecture i...

Use http://www.nomnoml.com/ ![nomnoml](https://us...  
azu/faao#60 updated 2017-07-03 01:07 by azu 0

chore(travis): deploy task without PR

close #214  
almin/almin#218 updated 2017-07-03 05:07 by azu 0

Fix 'unused function argument' in StoreGro...

I used almin with flowtype. Flowtype error occurred in ...  
almin/almin#217 updated 2017-07-02 23:07 by Leko 3

Sync with gist

We want to add sync feature between A PC <-> B PC ...

Main: always show content and searchbar

azu / faao

Unwatch 2

Pull requests Issues Marketplace Gist

Code Issues 14 Pull requests 0 Projects 0 Wiki Settings Insights

# Main: always show content and searchbar #61

Open azu opened this issue 8 hours ago · 0 comments

azu commented 8 hours ago

SearchBar and Content will be shown when the items is empty.  
It will improve UX.

azu added Status: Proposal Type: Bug labels 8 hours ago

Write Preview

Leave a comment

# 永続化

- 検索履歴
- 検索クエリ
- 設定
- アクティビティ
- etc....
- 大体のモデルが永続化可能な形になってる

# 永続化はRepositoryの仕事だけど

- モデルのシリアル化/デシリアル化の定義をするのは誰?
- static fromJSONとtoJSONという安易な実装をモデルに生やしてる
- もっといい方法が欲しい(Decoratorはパス)

```
... static fromJSON(json: GitHubSettingJSON): GitHubSetting {
... return new GitHubSetting({
... id: new Identifier<GitHubSetting>(json.id),
... token: json.token,
... apiHost: json.apiHost,
... webHost: json.webHost,
... gitHubUserId: json.gitHubUserId
... ? new Identifier<GitHubUser>(json.gitHubUserId)
... : undefined
... });
... }

... toJSON(): GitHubSettingJSON {
... return Object.assign(target: {}, this, source2: {
... id: this.id.toValue(),
... gitHubUserId: this.gitHubUserId ? this.gitHubUserId.toValue() : undefined
... });
... }
```

# ドメインモデルは永続化(技術的制約)を知らずに済むか

## Patterns, Principles, and Practices of Domain-Driven Design より

- 妥協なしで行う
  - NHibernate<sup>読</sup>やEntity Frameworkなどのデータモデルとのマッピングできるものを使う
  - モデルをそのままJSONなどにシリアルライズして保存できるデータストアを使う
- 妥協ありで行う
  - リポジトリからデータを引くときに、Entityに対して外から値を指しながら復元させる
  - Mementoパターン - Entityのスナップショットとデータモデルをマッピング(今これ)

---

<sup>読</sup>えぬはいばーねいと

## Using a Persistence Framework That Can Map the Domain Model to the Data Model without Compromise

If you are mapping to a relational database in a greenfield environment and you are using an ORM that supports persistent ignorant domain objects, you will be able to map your domain model directly to the data model, as shown in Figure 21-2.

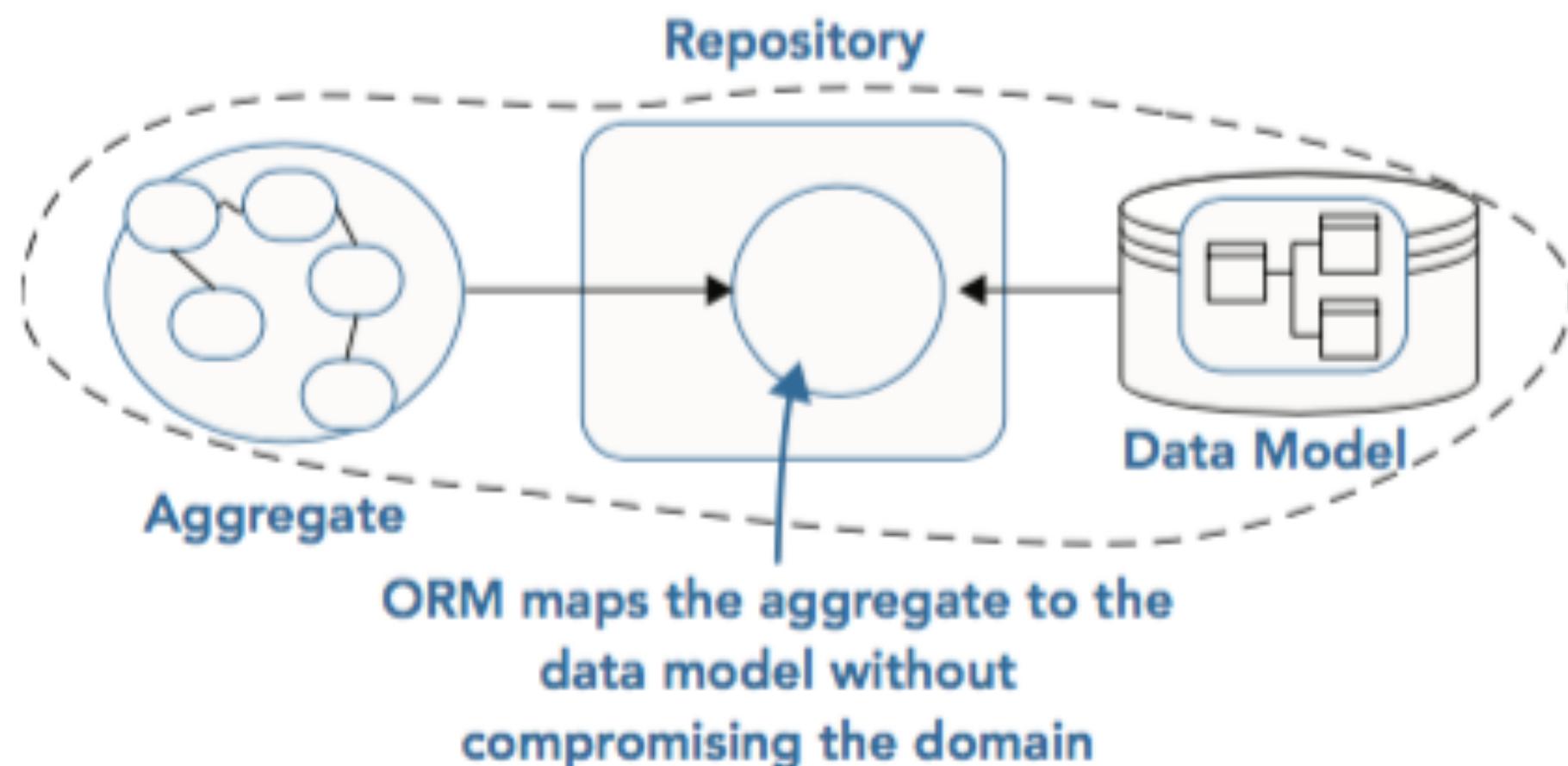
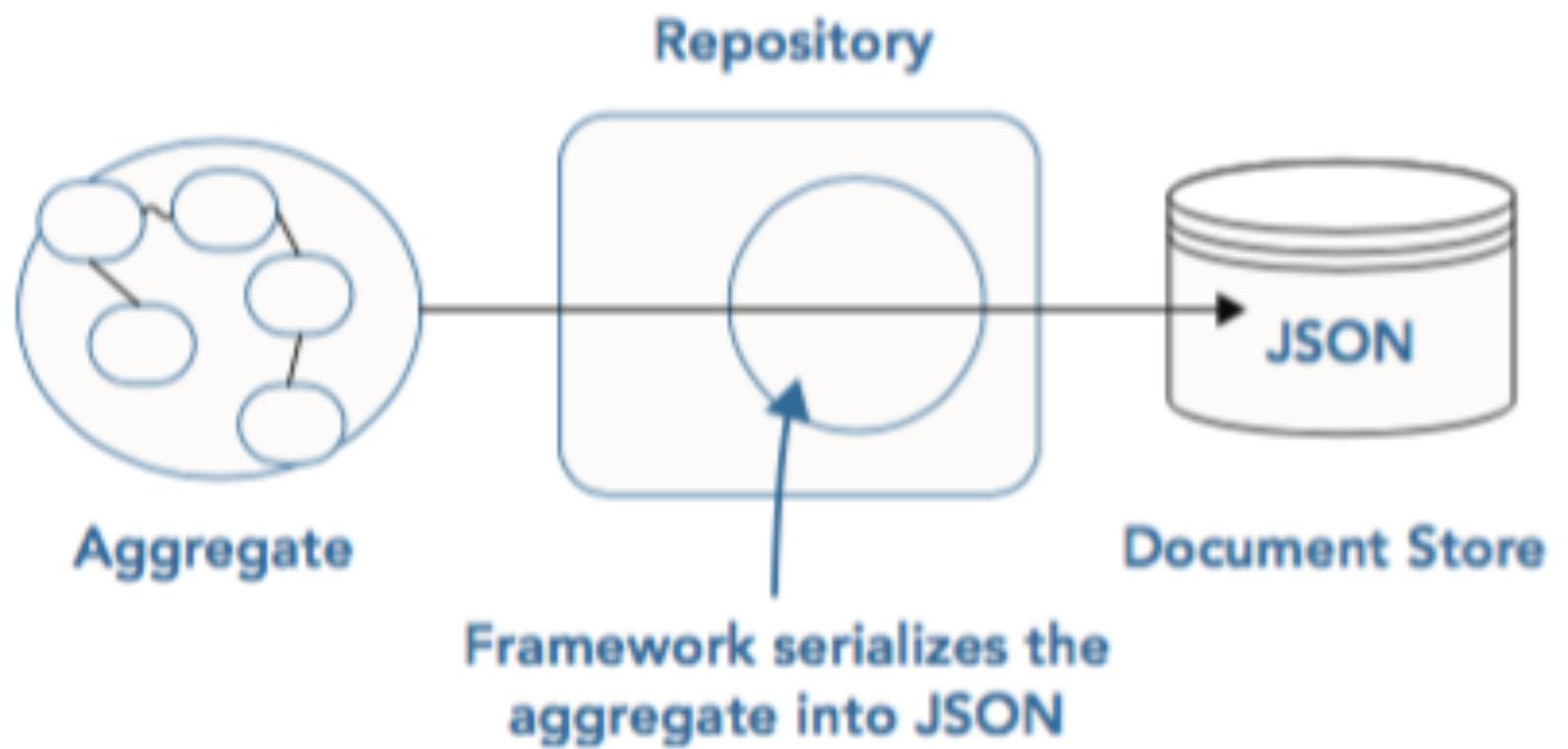


FIGURE 21-2: An ORM maps between the domain and the persistence model.



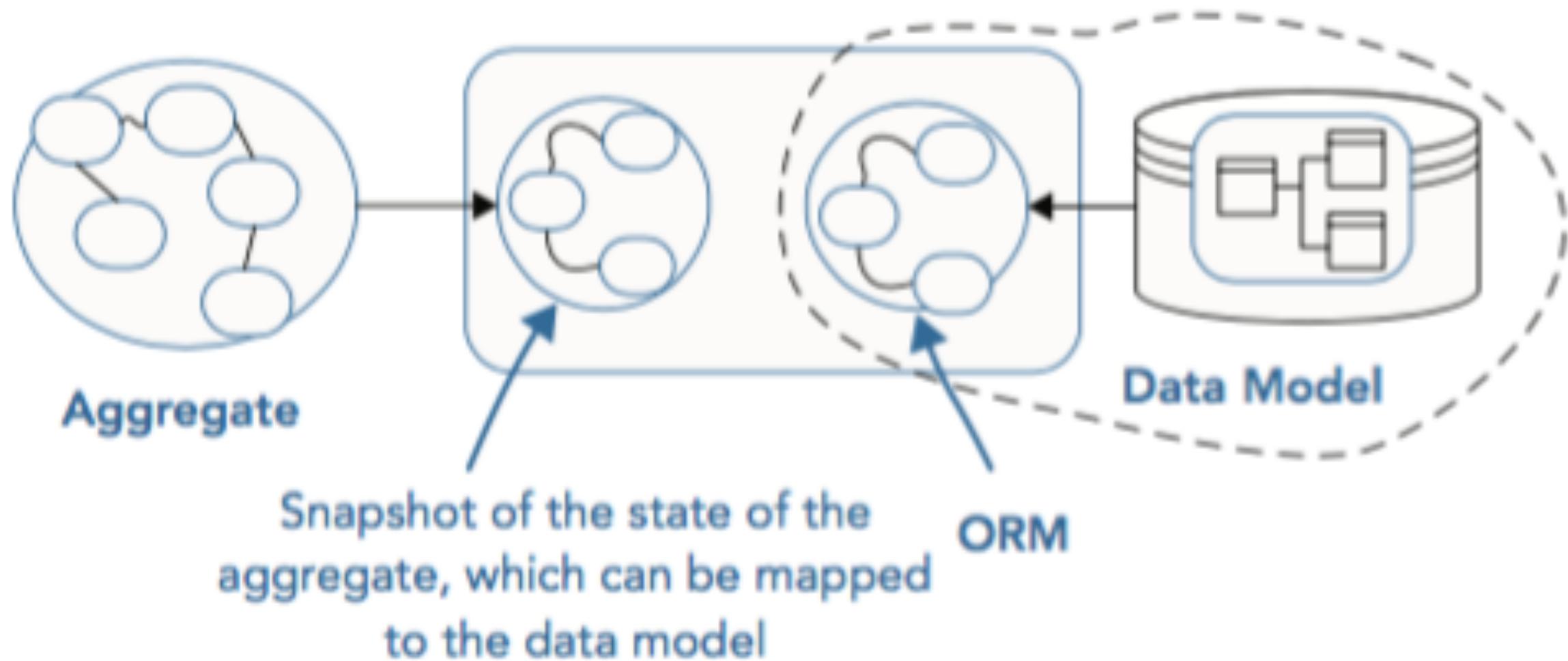
**FIGURE 21-3:** An aggregate can be serialized and stored.

## Using a Persistence Framework That Cannot Map the Domain Model Directly without Compromise

If you are using a persistence framework that does not allow your domain model to be persistence ignorant, you need to take a different approach to the way you persist and retrieve your domain objects so they remain free of infrastructural concerns. There are a number of ways that you can achieve this, but all affect the domain model and the shape of your aggregates. This is, of course, the compromise you need to make your application work.

## Repository

The repository mediates between the aggregate and the data model



**FIGURE 21-4:** The memento pattern enables you to map a snapshot of the domain model to the persistence model.

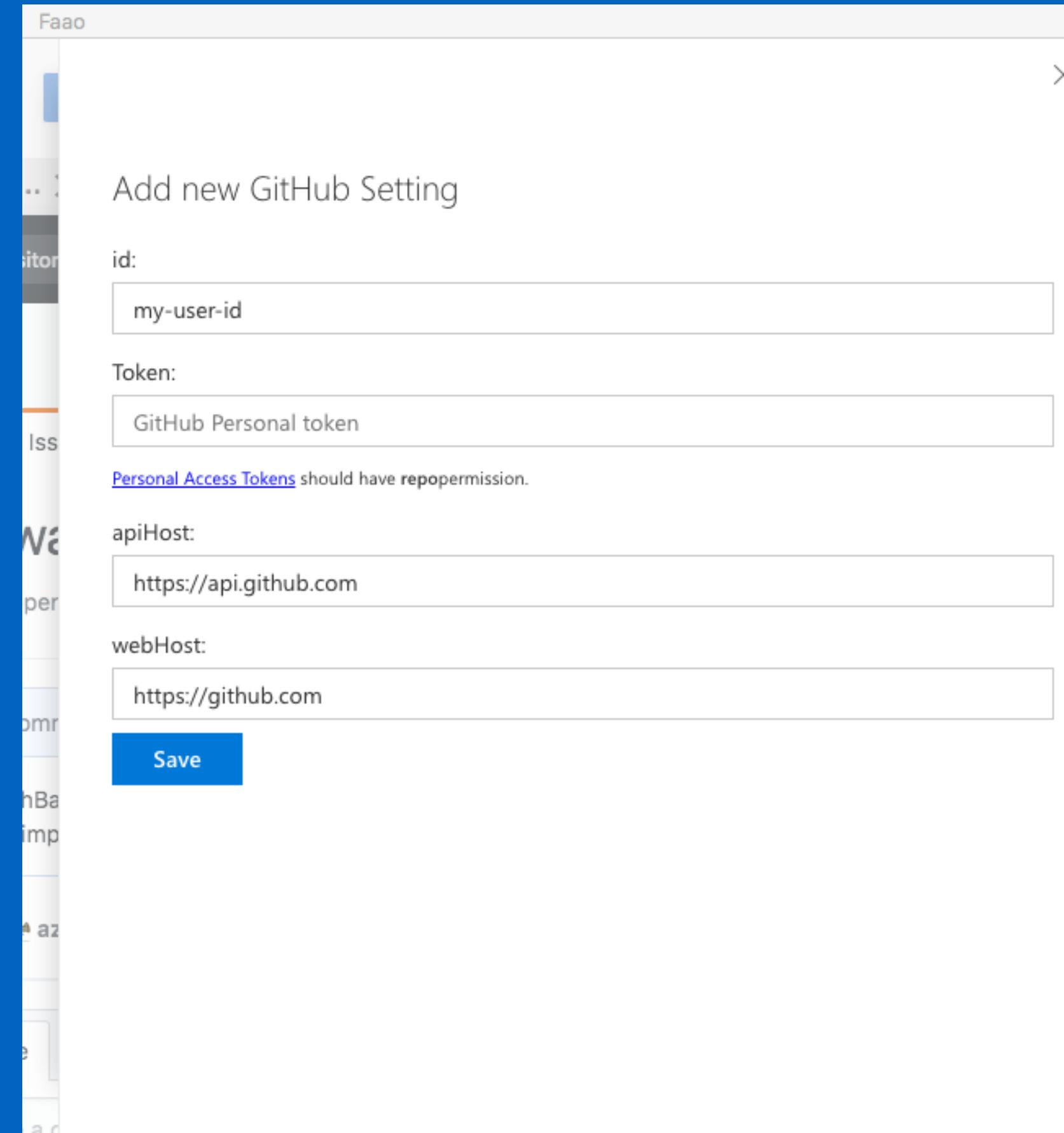
# 妥協あり/なしの永続化

- ドメインは軽く永続化されることを意識する必要はある
- constructorでincrement idをしていると不整合を生むので駄目
  - constructorでちゃんと{ id }なども受け取れるようにする
  - モデルの初期化は面倒になっていくのでFactoryが初期化を担当する

```
// 駄目なケース
let id = 0;
class User {
 constructor(){
 this.id = id++
 }
}
```

# スナップショットからの復元

- 今採用しているパターン
- 妥協ありパターンの一種である[TypeScript: Working with JSON](#)・[Choly's Blog](#)(Entityに対して外から値を指しながら復元させる)に比べると少し安全で何とか手で書いていけるレベル
- しかしスナップショットが現在のモデルと一致してるのは限らない
- スナップショットのバージョニングなどが必要となっていく
  - フレームワークになってないとそろそろ面倒



# Repository

- インメモリで終わる or データが常にサーバにある場合のRepositoryは単純なMap
- モデルの永続化を考えだしたときに大変になるのは、Repository
- モデルも永続化は全く意識はしていない場合、後から概念/構造に変更が出て大変となる
  - 影響度: 概念 > 構造 > 実装...
  - ついでに永続化するとIndexedDBなどを使うの非同期処理がやってくる
    - FaaOの実装では初期化と保存のみを非同期にして、Readは同期にした
    - Readを非同期にするとStoreも非同期にする必要がでてきて面倒そうだった

# UseCase

# UseCase

- アプリケーションのドメインを使った、やりたいことの流れを書くところ
- このアプリのユースケースは
  - GitHub APIを使ってSearch
  - GitHubSettingの作成、保存 などなど
- ユースケースの再利用性
  - 基本的にはしない、拡張ユースケースを使う
  - UseCaseの再利用性 - [yoskhdia's diary](#)

# ユースケース図

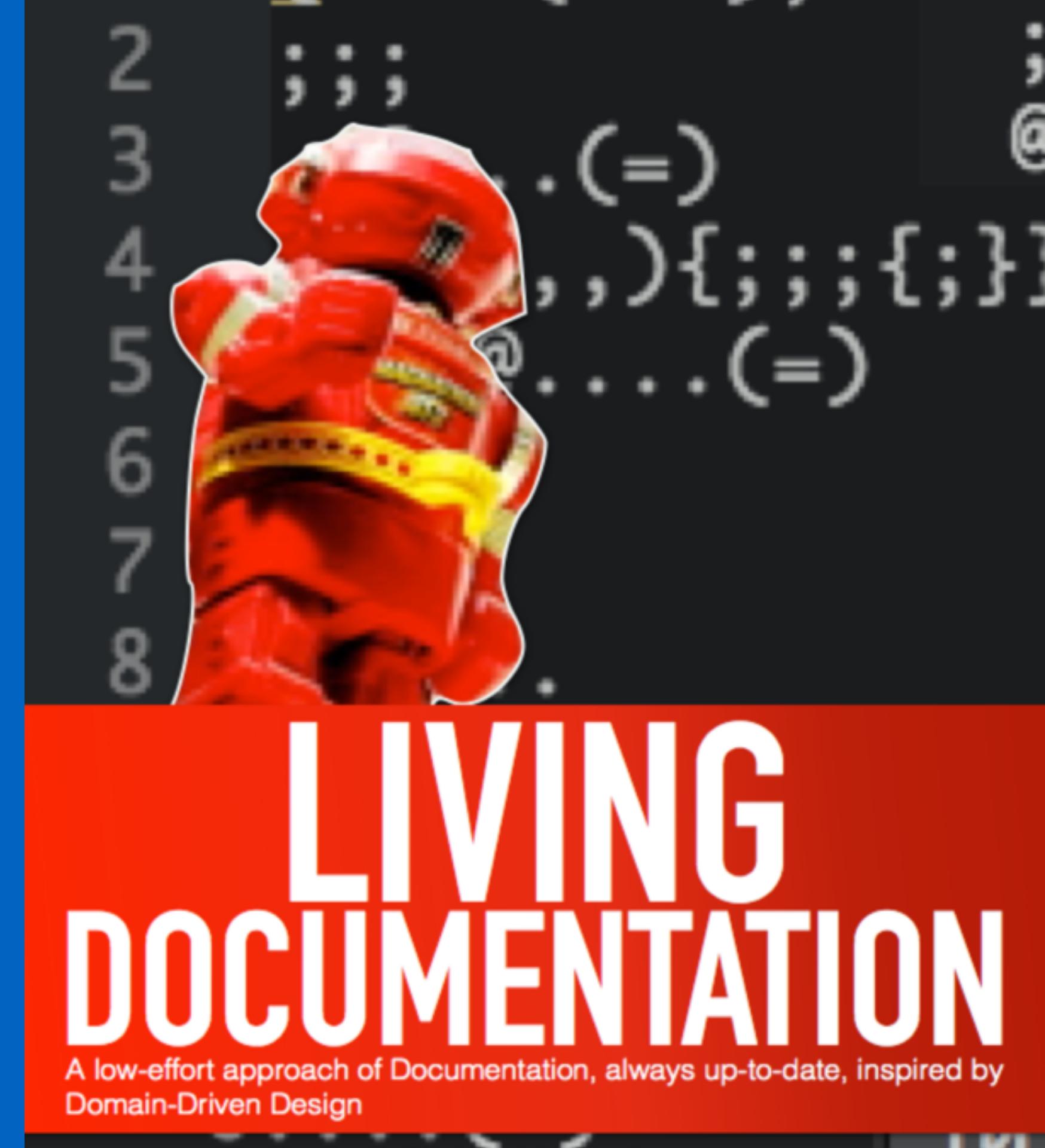
- FaaOのユースケース: [FaaO - UseCase architecture](#)

一点、注意が必要なのは、ユースケース記述とユースケース図は異なるということです。このガイドラインはユースケース記述のガイドラインです。

[UseCaseの再利用性 - yoskhdia's diary](#)

# Living Documentation

- Living Documentation by design, with Domain-Driven Design
- <https://leanpub.com/livingdocumentation> \$0～\$40で購入



# 知識の共有

KnowledgeにはGenericなものとSpecificなものがある。

会社やチーム、プロダクトにおけるSpecificな知識には次のような問題が生まれやすい

- アクセスできない
- 古すぎる
- フラグメント化してる
- 暗黙的になってる
- 理解できない
- 書かれてない

# Living Documentation

- これらの問題をLivingなドキュメントで解決するアプローチ
- ドキュメントもコードと同じ速度で成長する
- 良いドキュメントには良い設計が必要
- 良いドキュメントには自動化が必要
- 推測、憶測をドキュメント化しない

# LivingDocumentationのコア原則

- Reliable - 信頼性の高いドキュメント
  - single source of truth
  - reconciliation mechanism
    - ソースが複数の場所にあることを認め、それをテストする
- Low-Effort
- Collaborative
  - Conversations over Documentations
  - アクセスできる場所に知識は置く
- Insightful
  - 意図を残す

# 具体的な問題と対策

- ガイドラインを決めてもそれを自動で守れないと意味がない
  - ツールで検証する
  - コードで検証する
- 更新されない構成図
  - Living Diagram
  - 更新されないユビキタス言語
- etc..

# Living Documentationの4つのステップ

1. 何処かに保存されたデータの範囲を選択
2. データをドキュメントの目的に沿ってフィルター
3. フィルターした結果、各データのサブセットを抽出
4. ドキュメントを生成するためのフォーマットへ変換

# 例

- ユースケース図の自動生成
- レイヤーのバイオレーション検知
- Lint
- メタテスト

守られないルールは価値がない

# 守られないルールは価値がない

- 最も良いドキュメントはno document
- 必要となった時(ツールがエラーと言った時に初めて見ることができればいい
- ESLintがよくできている理由
- eslint, prettier, stylelint, webpack(case-sensitive-paths-webpack-plugin)などなど

例) ルール: ドメインはインフラ(repository)を参照してはいけない

dependency-cruiserを使ってルールをコード化し自動チェックする

```
{
 "forbidden": [
 {
 "name": "domain-not-to-depend-on-infra",
 "comment": "Don't allow dependencies from domain to infra",
 "severity": "error",
 "from": { "path": "^src/domain" },
 "to": { "path": "^src/infra" }
 }
]
}
```

破れないルールは価値を鈍化させる

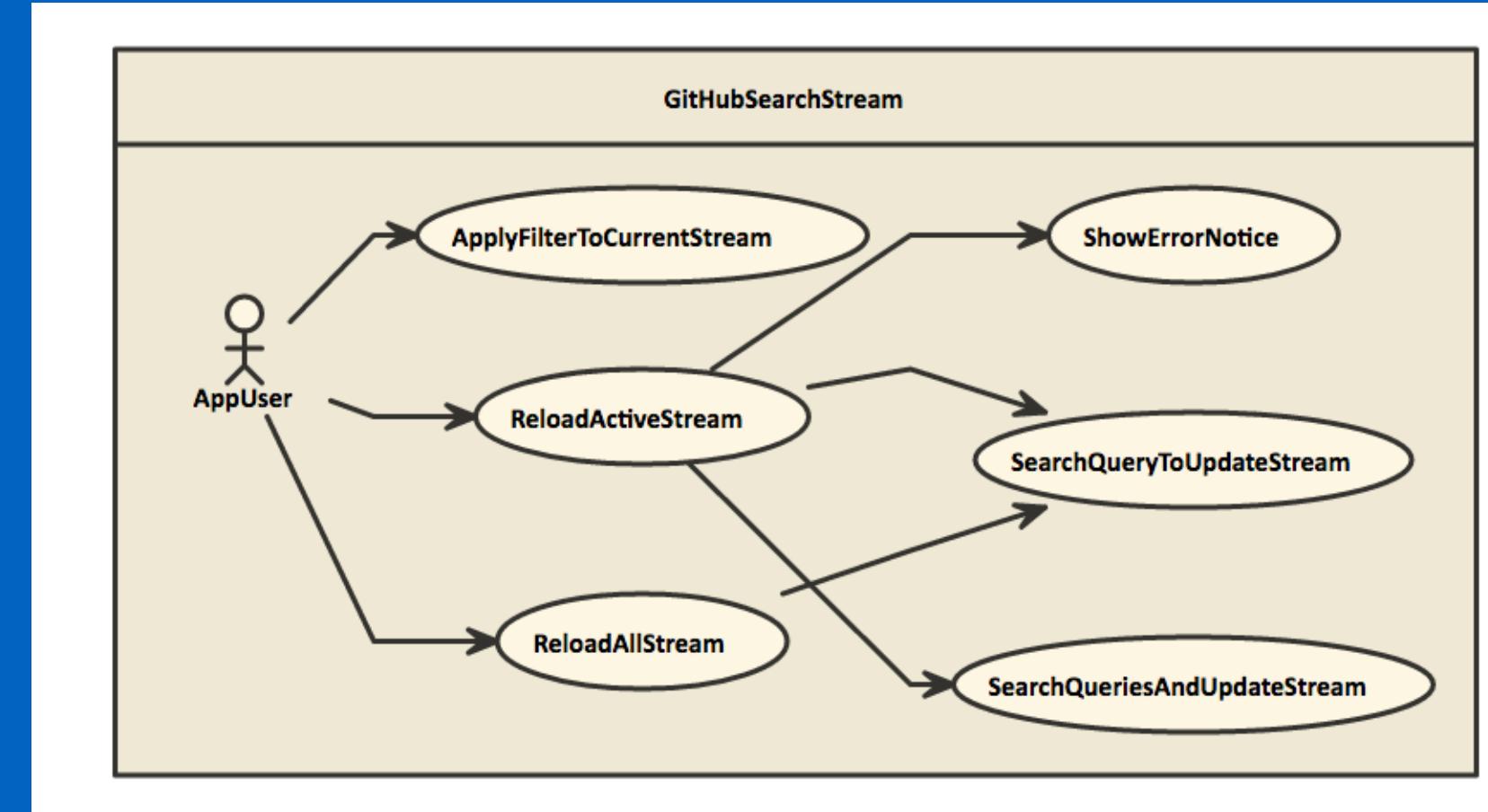
- ルールには例外がつきもの
- そのため、原則が守れないと崩壊してしまうルールよりは、例外を規定することで原則を守れるルールの方がよい。
- 厳密に守りたいルールはホワイトリストで管理できた方がいい

# Living Documentation

## Living Diagram

# ユースケース図のLiving Diagram

- Fao - UseCase architecture に全てのユースケース図が自動生成される
- Faoのソースコードから自動生成
- ファイルからuse-caseを抽出、Text to UMLの nomnoml が食べられる書式にして変換
- alminのUseCaseは拡張ユースケースを表現できる
  - ユースケースが別のユースケースを呼び出す
  - UseCaseの再利用性 - yoskhdia's diary



# Living Diagramの使いみち

- おかしなアクターを見つけることができる
  - 「名詞（主語） - 動詞 - 名詞（目的語）」(en)
  - 誰? がおかしいときがある。システムである場合など
- おかしなユースケースを見つけることができる
  - ユースケースは処理の流れを書く
  - そのため、省かれた処理を見つけ適切にキャッチすると多くのバグが解決できる

# 2/3のバグはカバレッジを上げると見つかる

- 適切なエラーハンドリングが行われるか、例外を無視しないかをテストしていくことで、全体の2/3のバグが発見できる(データ集約型分散システムの論文)

A majority of the production failures (77%) can be reproduced by a unit test.

– *Simple testing can prevent most critical failures | the morning paper*

# Living Documentationはドキュメンテーショ ンをコード化する

詳しくは本を読んで

- Living Documentation by... by Cyrille Martraire [PDF/iPad/  
Kindle]
- Living Documentation by design, with Domain-Driven Designを  
読んだ | Web Scratch

Almin



# Almin

- TypeScriptで書き直した
- Alminはフレームワークだが、今回のドメインやRepositoryは自分で書くところなので手出しじゃない
- あくまで思考を手助けする(そういう風にかけるというドキュメントがある)

限りなく薄いフレームワークとは



設計指針

- 安全
- 読むコード最小

# TypeScript

- まあ普通
- ツールのエコシステムに問題があったけどBabylon@7.0.0-beta.16でTypeScriptのパースができるようになった
- Release v7.0.0-beta.16 · babel/babylon

# Jest

- TypeScriptとの使い勝手がいいテストフレームワーク
  - ts-jestがよく出来る
  - TypeScript -> js -> babelなどもできる
- Assertion
  - expect 今回こっち
  - assert 後から気づいたけど普通にassertでもかける
  - power-assertもbabel変換とかができる

# Jest

- mock
  - Jestに依存したくないのでtypemoq
  - 実際のJSONや関数を使うので、anyを作るnullmock程度にしか使ってない
  - Almin的に基本的にコンストラクタDIとかできるように書けるので使う部分はあんまりなかった
- AutoMockはいらない
  - Painfulな機能

# Jest色々

- CLIは良く出来る
- 機能が多すぎる
- デフォルトでJSDomが入ってるのでNodeでもwindowがデフォルトで存在
  - "testEnvironment": "node"で回避
  - I found window is global in jest from StackOverflow, but not mention in docs? · Issue #3692 · facebook/jest
- 感想: Javascript unit testing tools
- Mochaはライブラリ向け、Jestはアプリ向け

# メタテスト

- azu/large-scale-javascript: 複雑なJavaScriptアプリケーションを作るために考えること
- 特定のクラスやディレクトリに対してルールを守ってるかのテスト
- StoreがちゃんとStoreGroupに登録されてるか、初期Stateを返せるかなど
- コードを書くと勝手にテストが増えて便利 ★

Work on everything

# FaaOの対応環境

- Browser
- Mobile(iOS Safari)
- Electron
- 最初はElectron向けに書いていたけど、これどこでも動くなと気づいてスイッチした

# GitHub API

- octokit.jsを使ってる
  - 0.9 Fetch APIベースになって壊れてる
- スキーマからAPIの対応を自動生成してるクライアント
- GitHubのdesktopも使ってたが
  - There can be only one API layer by joshaber · Pull Request #2080 · desktop/desktop 辞めた
- TypeScriptから扱いやすいやつ欲しい
  - レスポンスの型が面倒

# GraphQL

- GraphQLはGitHub Enterprise 2.10にも入った
- /users/:user/eventに相当するもののとり方がわからない
- <https://developer.github.com/v3/activity/events/#list-public-events-performed-by-a-user>

# GitHub API trap

- GHEだとRate Limitの機能が無効化されてるケースがある
  - 常に404を返す
  - どう見ても叩いたら壊れる
- GitHubのURLをパースするやつ
  - [github-url-to-object](#)を利用
  - GHEも対応してる
- GitHubのeventsをフォーマットするやつ
  - ダッシュボードの "pivotal-brian-croom opened issue on pivotal/cedar" みたいなメッセージを作るやつ
  - [parse-github-event](#)を書いた

# Philosophy

# Debuggability - 状態

- アプリケーションには様々な状態が存在する
- 全てはどこからでも現在の状態を見れるようになつてないと不便
  - 簡単に `window.faa` に参照突き刺しておけばいい
- Repository
- Store/State
- ViewのState

# Debuggability - データ

- 永続化したデータはいつでもメモリデータベースに切り替えできた方が良い
- テストの度に永続化したデータが消えるとテストしにくい
- FaaOではStorage.tsでいつでもメモリデータモードに入ることができる
  - localForageを使って動的にdriverを切り返す
  - データは元からメモリ上に載っていて、書き込み時にデータベースへアクセスする作りにしたため

# Debuggability - イベント

- Stateとイベントを見比べる
- almin-loggerやalmin-devtoolsで UseCaseの実行を確認する
- イベントを見ることは大事
  - Webの世界はイベント駆動
  - DOMには色々なイベントがそれはブラウザによっても違う
  - video-events-debugger

```
Console was cleared
[WDS] Hot Module Replacement enabled.
faao:AppRepository Save entity +0ms
▶ SystemReadyToLaunchAppUseCase(includes "UpdateAppNetworkStatusUseCase")
faao:GitHubSearchStreamRepository Save stream with query +10s
faao:AppRepository Save entity +25ms
faao:GitHubClient response ▶ Object +416ms
faao:SearchGitHubUseCase Searching result +0ms
▶ GitHubSearchResult
faao:SearchGitHubUseCase continueToNext +1ms true
faao:GitHubSearchStreamRepository Save stream with query +4ms
faao:SearchGitHubUseCase Searching Complete! Query:Issue +217ms
▶ SearchQueryAndOpenStreamUseCase(includes "AppUserOpenStreamUseCase, AppUserSelectFirstItemUseCase, SearchQueryToUpdateStreamUseCase")
> window.faao
< Object {repositories: Object, storeMapping: Object, debugOn: function, debugOff: function} ⓘ
 ▶ debugOff: function ()
 ▶ debugOn: function ()
 ▶ repositories: Object
 ▶ state: Object
 ▶ app: AppState
 ▶ gitHubSearchList: GitHubSearchListState
 ▶ gitHubSearchStream: GitHubSearchStreamState
 ▶ gitHubSetting: GitHubSettingState
 ▶ gitHubUser: GitHubUserState
 ▶ mobile: MobileState
 ▶ notice: NoticeState
 ▶ profile: ProfileState
 ▶ quickIssue: QuickIssueState
 ▶ __proto__: Object
 ▶ storeMapping: Object
 ▶ get state: function state()
 ▶ __proto__: Object
>
```

# まとめ

- ちゃんとやるにはちゃんとやる必要がある
- コードと共にテストやドキュメントも成長する
- それらは自動化されている部分もあればルール化されている部分もある
- モデリングをちゃんと行い、モデルから自動的にドキュメントが生成され、ドキュメントとしてみた時のモデルとしての不整合を検証する

# 参考

- azu/large-scale-javascript: 複雑なJavaScriptアプリケーションを作るために考えること
- Patterns, Principles, and Practices of Domain-Driven Design 1st Edition
- ユースケース駆動開発実践ガイド
- Living Documentation by... by Cyrille Martraire [PDF/iPad/Kindle]