

DB設計者にも うれしいDBFlute

久保 雅彦 (jflute)

私はだれ？

- 久保 雅彦
 - はてなブログ、Twitterでは **jflute**
- フリーランスのオープンソースプログラマ
 - DBFluteメインコミッタ
 - Seasar.NETリーダー
 - その他S2Dao, Teedaなどのコミッタ
- **株式会社レイハウオリ**
 - 技術コンサル・Javaチーム全体教育
- **株式会社ビズリーチ・株式会社ルクサ**
 - 技術コンサル・主にDB周りの教育

メインテーマ

DBFluteとは？

すいません...

5分じゃ無理！

(> <)

ぶつくさぶつくさ

えーつと...Seasarプロダクト...
オープンソース...O/Rマッパ...

ぶつくさやめっ！

なんて、

しのごいってないで！

とにかく

DB設計者にも

うれしいO/Rマッパ

というか

自分がDB設計者だったら...

プログラマに使って欲しい

O/Rマッパ **No.1**

ポリシー

DB変更に強い！

またぶつくさぶつくさ...

自動生成ドリブンでジェネレー
ションギャップでプログラム上
でタイプセーフでどうのこう
の...

ぶつくさやめっ！

やっぱ5分じゃ無理！

(> <

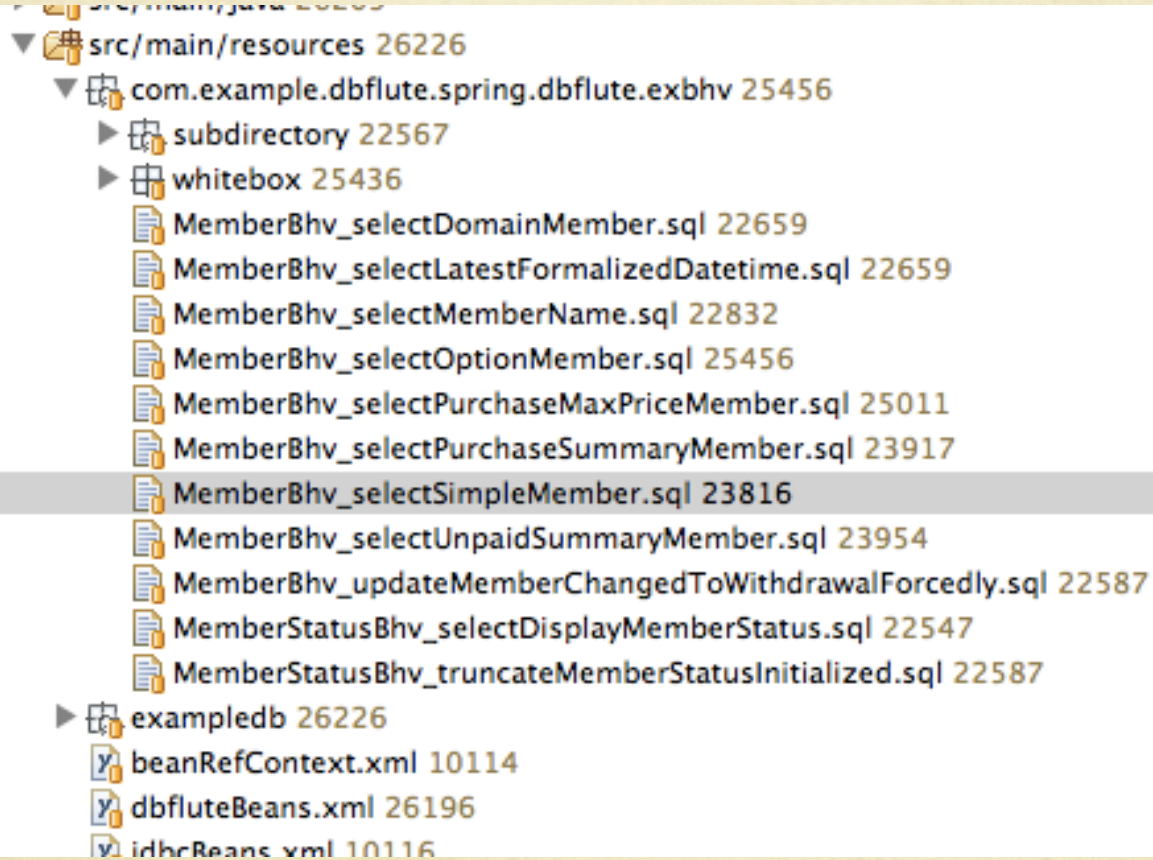
わりと便利

外だしSQLの一括テスト

(デモ)

- 2Way-SQLでSQLはいつでも実行できるように
- 一括実行でDB変更の影響範囲すぐに特定
- (実行することでSQL対抗DTOも自動生成できる)

デモ (のつもり)



A screenshot of a file explorer window showing a project structure. The root directory is `src/main/resources`. Inside it is a package `com.example.dbflute.spring.dbflute.exbhv`. This package contains a subdirectory `subdirectory` and a directory `whitebox`. The `whitebox` directory contains several SQL files, with `MemberBhv_selectSimpleMember.sql` highlighted. Below the `whitebox` directory is a directory `exampledb` which contains XML files: `beanRefContext.xml`, `dbfluteBeans.xml`, and `idhrBeans.xml`.

- ▼ `src/main/resources` 26226
 - ▼ `com.example.dbflute.spring.dbflute.exbhv` 25456
 - ▶ `subdirectory` 22567
 - ▶ `whitebox` 25436
 - `MemberBhv_selectDomainMember.sql` 22659
 - `MemberBhv_selectLatestFormalizedDatetime.sql` 22659
 - `MemberBhv_selectMemberName.sql` 22832
 - `MemberBhv_selectOptionMember.sql` 25456
 - `MemberBhv_selectPurchaseMaxPriceMember.sql` 25011
 - `MemberBhv_selectPurchaseSummaryMember.sql` 23917
 - `MemberBhv_selectSimpleMember.sql` 23816**
 - `MemberBhv_selectUnpaidSummaryMember.sql` 23954
 - `MemberBhv_updateMemberChangedToWithdrawalForcedly.sql` 22587
 - `MemberStatusBhv_selectDisplayMemberStatus.sql` 22547
 - `MemberStatusBhv_truncateMemberStatusInitialized.sql` 22587
 - ▶ `exampledb` 26226
 - `beanRefContext.xml` 10114
 - `dbfluteBeans.xml` 26196
 - `idhrBeans.xml` 10116

DB変更されました！

```
-- !df:pmb!  
-- !!Integer memberId!!  
-- !!String memberName:likePrefix!!  
-- !!Date birthdate!! // used as equal
```

```
select member.MEMBER_ID  
      , member.MEMBER_NAME  
      , member.MEMBER_BIRTHDAY  
      , memberStatus.MEMBER_STATUS_NAME  
from MEMBER member  
  left outer join MEMBER_STATUS memberStatus  
    on member.MEMBER_STATUS_CODE = memberStatus.MEMBER_STATUS_  
/*BEGIN*/  
where  
  /*IF pmb.memberId != null*/  
  member.MEMBER_ID = /*pmb.memberId*/3
```

MEMBER_BIRTHDAY が BIRTHDATE に



OutsideSqlTest実行！

```
jflute    staff    310    1    7    2012  manage.bat
jflute    staff    234    1    7    2012  manage.sh
jflute    staff    136    1    7    2012  output
jflute    staff    279    1    7    2012  outside-sql-test.ba
jflute    staff    299    1    7    2012  outside-sql-test.sh
jflute    staff    340    3    5    13:32  playsql
jflute    staff    272    1    7    2012  replace-schema.bat
jflute    staff    292    1    7    2012  replace-schema.sh
jflute    staff    204    4    12   11:21  schema
jflute    staff    265    1    7    2012  sql2entity.bat
jflute    staff    285    1    7    2012  sql2entity.sh
dbflute_exampleadb jflute$ sh outside-sql-test.sh
```

SQLの実行エラー

```

/* ****
Failed to execute the SQL!

[SQL File]
../src/main/resources/com/example/dbflute/spring/dbflute/exbhv/MemberBhv_selectSimpleMember.sql

[Executed SQL]
/*
[df:title]
Example for Simple Select

[df:description]
This SQL is the most basic example for outsideSql.
It uses CustomizeEntity and ParameterBean.
*/
-- #df:entity#

-- !df:pmb!
-- !!Integer memberId!!
-- !!String memberName:likePrefix!!
-- !!Date birthdate!! // used as equal

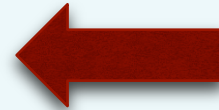
select member.MEMBER_ID
      , member.MEMBER_NAME
      , member.MEMBER_BIRTHDAY
      , memberStatus.MEMBER_STATUS_NAME
from MEMBER member
left outer join MEMBER_STATUS memberStatus

```

エラー内容は詳しく

エラーを直せば

```
select member.MEMBER_ID  
      , member.MEMBER_NAME  
      , member.BIRTHDATE  
      , memberStatus.MEMBER_STATUS_CODE  
from MEMBER member  
left outer join MEMBER_STATUS  
on member.MEMBER_STATUS_CODE
```



成功！

```
[df-outside-sql-test] url      = jdbc:mysql://localhost:4
[df-outside-sql-test] schema  = {exampledb.$$NoNameSchem
[df-outside-sql-test] user    = exampledb
[df-outside-sql-test] props   = {}
[df-outside-sql-test] additionalSchema = nextexampledb.
[df-outside-sql-test] repsEnvType      = ut
[df-outside-sql-test] refreshProject   = dbflute-mysql-
[df-outside-sql-test] _/_/_/_/_/_/_/_/_/_ {OutsideSqlTest}
```

BUILD SUCCESSFUL

Total time: 2 seconds

jflute-mac-air:dbflute_exampledb jflute\$ █

少なくとも

アプリの(外だし)SQLへの
DB変更の構造上の影響は
すぐに検知

地味に便利

DB変更履歴の自動生成

(デモ)

- プログラマへのDB変更の通知をスムーズに
- 「このカラムいつ頃追加されたんだっけ？」

デモ (のつもり)

Diff Date: 2010/11/26 19:52:15

for many pattern tests
table count: 13 -> 16

Add Table

- MEMBER_SERVICE
- PRODUCT_CATEGORY
- SERVICE_RANK

Change Table

- MEMBER_SECURITY

Add Column

- REMINDER_USE_COUNT

- PRODUCT

Add Column

- PRODUCT_CATEGORY_CODE
- PRODUCT_REGULAR_PRICE

Add FK

- FK_PRODUCT_PRODUCT_CATEGORY

Add Index

- FK_PRODUCT_PRODUCT_CATEGORY_INDEX_1

HistoryHTML

=> DB変更の履歴一覧

Diff Date: 2010/06/15 16:41:11

restore the dollar table (wrong deleted)
table count: 12 -> 13

Add Table

今のDB構造

```
1  
2 create table MEMBER(  
3     MEMBER_ID INTEGER IDENTITY NOT NULL P  
4     MEMBER_NAME VARCHAR(200) NOT NULL,  
5     MEMBER_ACCOUNT VARCHAR(50) NOT NULL,  
6     MEMBER_STATUS_CODE CHAR(3) NOT NULL,  
7     FORMALIZED_DATETIME DATETIME,  
8     BIRTHDATE DATE,  
9     REGISTER_DATETIME DATETIME NOT NULL,
```


DB変更！

NotNull制約を外して、
サイズを50から80に

```
2 create table MEMBER(  
3     MEMBER_ID INTEGER IDENTITY,  
4     MEMBER_NAME VARCHAR(200) NOT NULL,  
5     MEMBER_ACCOUNT VARCHAR(80),  
6     CLUB_DB2_MEMO VARCHAR(200),  
7     MEMBER_STATUS_CODE CHAR(3) NOT NULL
```

カラム追加

ReplaceSchemaでDB反映！

```
ite/dbflute-0.9.9.6/build-torque.xml
```

```
2012-07-13 12:46:17,372 INFO - +-----
2012-07-13 12:46:17,375 INFO - |
2012-07-13 12:46:17,375 INFO - | ReplaceSchema
2012-07-13 12:46:17,376 INFO - |
2012-07-13 12:46:17,376 INFO - +-----
2012-07-13 12:46:17,378 INFO - ...Waiting for your GO SIGN
/
- - - - -
Database: jdbc:h2:file:../src/main/resources/exampledb/exampledb
Schema: EXAMPLEDB.PUBLIC
. - - - - - /
(input on your console)
The schema will be initialized. Are you ready? (y or n):
```


JDBCタスクとDocタスク



変更履歴を自動生成！

Diff Date: 2012/07/13 12:55:17

Change Table

- MEMBER

Add Column

- CLUB_DB2_MEMO

Change Column

- MEMBER_ACCOUNT

Size	50 -> 80
Not Null	true -> false

HistoryHTMLに新しい
履歴が追記される

相当便利

DB環境構築の自動化

(デモ...既にやった)

- DB変更をプログラマのDB環境にスムーズに反映させる
- DBをReplaceしてもテストデータも復元される
- 新規開発メンバーのDB環境もバッチ一発で作成

DBFluteのこだわり

DB変更はテーブル・カラムの
変更だけじゃない！

ストアドも

ストアドプロシージャの変更もDB変更の一つ



ストアドプロシージャ対応のクラスの自動生成



ストアドの名前、引数、SQL結果セットが変わったら、
再自動生成すればコンパイルエラーで検知...

区分値も

区分値の変更もDB変更の一つ



区分値対応のメソッドやENUMを自動生成



...

テストデータも

テーブルやカラムが変わったら

テストデータも変わらなきゃ



ReplaceSchemaでテストデータの一元管理



テストデータの変更を即座に横展開

やっぱり

DBは長生き

良い構造にしていきたい

そして

DB設計は国家資格じゃない

設計者は設計しながら成長していく

まずは

こういったDB設計の勉強会に
出席することが大事

もひとつ

DB変更を許容できるフレームワークを
アプリでは使って欲しい

DBFlute

DBFluteで、

DBのリファクタリングを

今日のお楽しみ

今日得たDB設計のスキルを
現場で発揮するために

おわり

ありがとうございました

- DBFluteはDB2にもしっかり対応
 - withRR, withRS, row_number() over() など
 - DB2のストアードプロシージャも簡単実行