S2DaoでもN:Nできます

自己紹介

- 名前:木村　聡(きむら　さとし)
- Seasarプロジェクトコミッタ:
  - S2Struts
  - S2Mai
  - 舞姫
- 仕事
  - (株)フルネス
  - フレームワーク
  - 自動生成ツール

これまで書いたものとか

- 書籍:
  - Eclipseで学ぶはじめてのJava
  - Seasar入門　～はじめてのDI&AOP～
- 雑誌、Web記事
  - CodeZine
  - DB Magazine
  - WEB+DB
    - 「Seasar2徹底攻略」(Vol.31)
  - JavaWorld
    - 「開発者にとって“易しく、優しい”軽量コンテナSeasar2の実力を探る」(2005/05)

デモ

テーブル

- EMP
  - EMPNO
  - ENAME
  - DEPTNO　　DETPと関連
  - その他

- DEPT
  - DEPTNO
  - DNAME
  - その他

SQL

```
SELECT
        EMP.DEPTNO
      , DNAME
      , EMPNO
      , ENAME
      ～略～
FROM
      ～略～
WHERE
      ～略～
ORDER BY
      EMPNO
```

DEPTNO	DNAME	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	TSTAMP	LOC
10	ACCOUNTING	7934	MILLER	CLERK	7782	1982-01-23	1300	(null)	2000-01-01 00:00:00.000000000	NEW Y
10	ACCOUNTING	7839	KING	PRESIDENT	(null)	1981-11-17	5000	(null)	2000-01-01 00:00:00.000000000	NEW Y
10	ACCOUNTING	7782	CLARK	MANAGER	7839	1981-06-09	2450	(null)	2000-01-01 00:00:00.000000000	NEW Y

マッピング

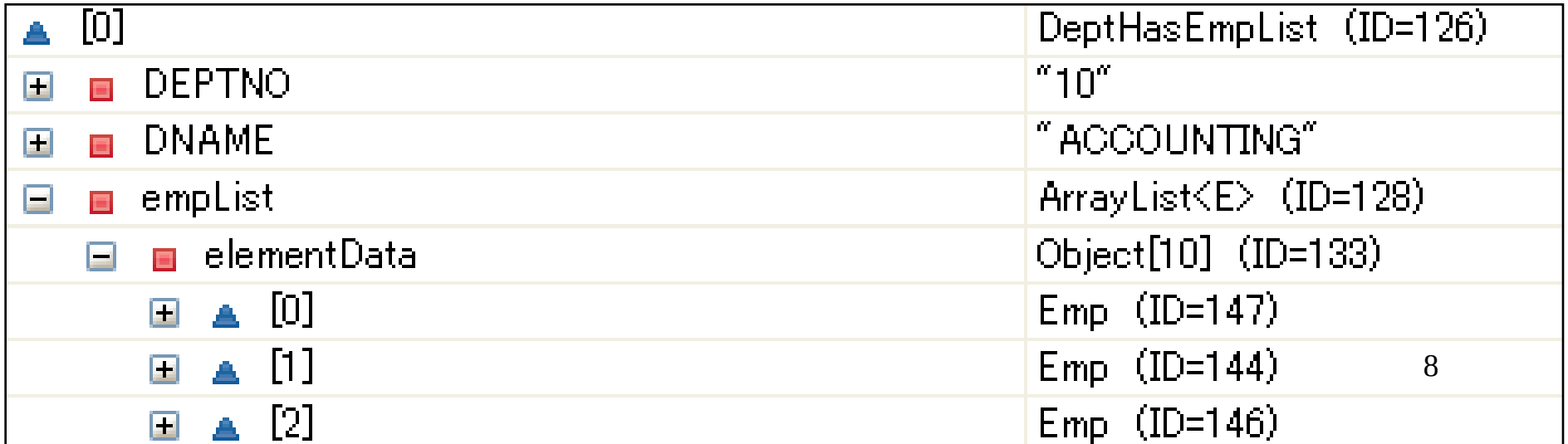
- S2Dao
  - フラットなList

DEPTNO	DNAME	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	TSTAMP	LOC
10	ACCOUNTING	7934	MILLER	CLERK	7782	1982-01-23	1300	(null)	2000-01-01 00:00:00.000000000	NEW YO
10	ACCOUNTING	7839	KING	PRESIDENT	(null)	1981-11-17	5000	(null)	2000-01-01 00:00:00.000000000	NEW YO
10	ACCOUNTING	7782	CLARK	MANAGER	7839	1981-06-09	2450	(null)	2000-01-01 00:00:00.000000000	NEW YO

マッピング

- S2Dao拡張版
  - 階層あり

DEPTNO	DNAME	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	TSTAMP	LOC
10	ACCOUNTING	7934	MILLER	CLERK	7782	1982-01-23	1300	(null)	2000-01-01 00:00:00.000000000	NEW Y
10	ACCOUNTING	7839	KING	PRESIDENT	(null)	1981-11-17	5000	(null)	2000-01-01 00:00:00.000000000	NEW Y
10	ACCOUNTING	7782	CLARK	MANAGER	7839	1981-06-09	2450	(null)	2000-01-01 00:00:00.000000000	NEW Y



使い方

- パラメータ化されたListのプロパティーを持たせる
  - 例えば
    - List<Dept>
    - List<Emp>
- Daoのselectメソッドに@S2DaoExtをつける
- diconファイル赤文字を追記

```
<components>
  <include path="kimu_dao.dicon"/>
  <component
    class="jp.dododododo.dao.ext.s2dao.interceptor.TestDao">
    <aspect>s2DaoExtInterceptor</aspect>
    <aspect>dao.interceptor</aspect>
  </component>
</components>
```

Selectの結果をそのままマッピング

- テーブル構造は関係ない
  - ResultSet→Object
    - SQLの実行結果に素直
  - 検索結果を見たまま素直にObjectにする
    - セレクト項目
    - ORDER BY
    - Objectの構造

まとめ(その1)

- Genericsを使った型情報など
  Classファイルから取れる情報が増えた
  - どう使うかが今後の課題
    →フレームワーク・ライブラリ提供者

kimu-daoを作った動機

- S2Daoのような機能を保守中のシステムでも使いたい
  - Connection渡すだけで使いたい
- ついでにいろいろな機能を付けた
  - 1:NやN:Nも扱いたい

つまり

- S2Daoのコマではないです

使い方

Daoのインスタンス

- コンストラクタにコネクションを渡す

```
Dao dao = new DaoImpl(connection);
```

- コンストラクタにデータソースを渡す

```
Dao dao = new DaoImpl(dataSource);
```

- データソースをセットする

```
DaoImpl dao = new DaoImpl();
dao.setDataSource(dataSource);
```

CRUD

- insert

```
dao.insert(entity);
```

- update

```
dao.update(entity);
```

- delete

```
dao.delete(entity);
```

バッチ

- insert

Collection<ENTITY>

```
dao.insert(entities);
```

- update

```
dao.update(entities);
```

- delete

```
dao.delete(entities);
```

select

- 1件

複数件数が返ってきたら例外

```
Emp emp = dao.selectOne(sql, Emp.class);
```

・sqlファイル（S2Daoと同じ）のパス
or
・S2Daoがsqlファイルに書いていたファイルの中身
(SQLを動的に組み立てられる)

```
"select * from emp
 /*BEGIN*/.../*END*/"
```

select

- 複数件

```
List<Emp> emp = dao.select(sql, Emp.class);
```

select

- 1件(Map)

```
Map<String, Object> m = dao.selectOneMap(sql);
```

select

- 複数件(Map)

```
List<Map<String, Object>> l = dao.selectMap(sql);
```

条件渡し

名前付の値

```
import static jp.dododododo.dao.util.DaoUtil.*;
...

Emp emp =
      dao.selectOne(sql,args("id",10), Emp.class);
```

```
Emp emp =
      dao.selectOne(sql,
      args("id", query.getId(), "name",query.getName()),
      Emp.class);
```

int

String

まとめ(その2)

- 可変長引数を使うとJavaっぽくないAPIが出来る
  - 新しいAPIの可能性

一件づつまわす

```
dao.select(sql, Emp.class, new FooCallback())
```

```
public class FooCallback
    implements IterationCallback<Emp> {

    public void iterate(Emp e) {
        // ここに処理を書く
    }

    public List<Emp> getResult() {
        return null;
    }
}
```

このカラムだけ

```
import static jp.dododododo.dao.columns.ColumnsUtil.*;
...

dao.update(entity,pc("col1","col2"));
```

Persistent Columns

このカラムを除く

```
import static jp.dododododo.dao.columns.ColumnsUtil.*;
...

dao.update(entity,npc("col1","col2"));
```

No Persistent Columns

ロック

- 楽観的ロック

```
import static jp.dododododo.dao.columns.ColumnsUtil.*;
...

dao.update(entity, OPTIMISTIC_LOCKING);

dao.update(entity,pc("col1","col2"), OPTIMISTIC_LOCKING);

dao.update(entity,npc("col1","col2"), OPTIMISTIC_LOCKING);
```

select数字

```
BigDecimal num = dao.selectOneNumber(sql);


int i = dao.selectOneNumber(sql).intValue();
```

独自機能

可変のオーダーバイ

```
SELECT
    *
FROM
    EMP
/*ORDER BY @SqlUtil@orderBy(orderBy)*/
    ORDER BY EMPNO
/*END*/
```

```
List<OrderByArg> orderBy = new ArrayList<OrderByArg>();
orderBy.add(new OrderByArg("ENAME", SortType.ASC));
orderBy.add(new OrderByArg("EMPNO", SortType.DESC));

dao.select(sql, args("orderBy", orderBy), Emp.class);
```

BEGIN

```
SELECT * FROM EMP
/*BEGIN*/
where
  /*IF false*/
    /*IF true*/
      EMPNO = 1
    /*END*/
  /*END*/
/*END*/
```

- 一つのIFがtrueでもwhere句が空なら BEGIN～ENDは除く
- S2Daoの場合エラーになる
  - SELECT * FROM EMP where

ID生成

```
public class Entity {

@Id(@IdDefSet(type = Sequence.class,name = "seqName"))
public void setId(int id) {
      this.id = id;
}

}
```

ID生成

```
public class Entity {

@Id({
  @IdDefSet(type = Sequence.class,name = "seqName"),
  @IdDefSet(db = HSQL.class,
    type = Sequence.class,name = "seqName", ),
  @IdDefSet(db = Oracle.class,
    type = RandomUUID.class)
})
public void setId(int id) {
      this.id = id;
}

}
```

ID生成

- DB毎に指定可能

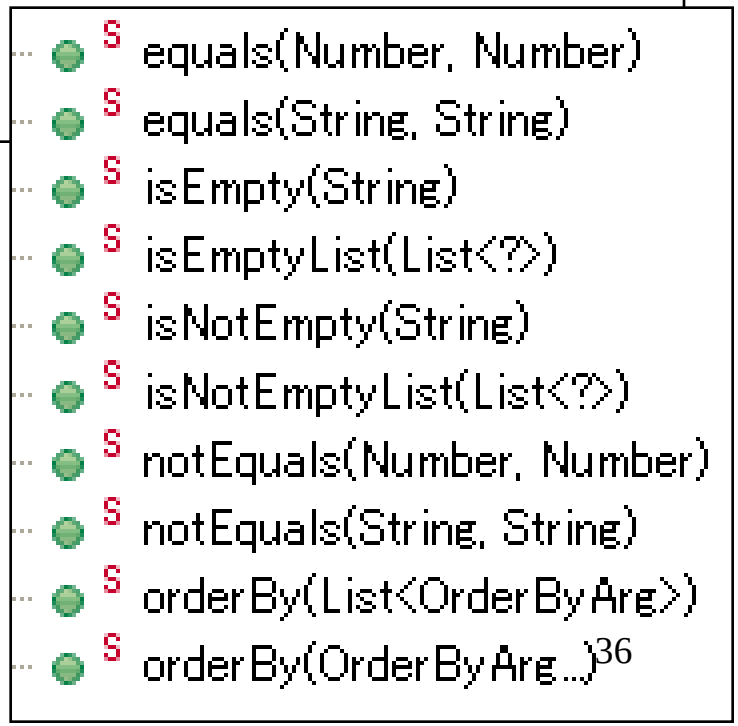
- 生成ロジック
  - sequence
  - Identity
    - Derby : IDENTITY_VAL_LOCAL()
  - UUID
  - 独自ロジックを作成可能
    - IdGeneratorをimplしたenumを作成
    - @IdDefSetのtypeで指定

更新、セレクトでのDBとオブジェクトのどちらを尊重するか

- 更新はDB
- セレクトはJava
  - できるだけ型変換する
    - PreparedStatementのsetXxx
    - ResultSetのgetXxx
      を使い分ける

SqlUtilをデフォルトパッケージに置いた

```
SELECT *
FROM EMP
/*ORDER BY @SqlUtil@orderBy(orderBy)*/
    ORDER BY EMPNO
/*END*/
```



- 割り切った

方言

- S2Daoと同じ＋α

```
public class Bean {
  @Dialects(
    dialect = { HSQL.class, Oracle.class },
    value = { "CURRENT_TIMESTAMP", "SYSDATE" })
  public String sysDate;
  @Dialects(
    dialect = { HSQL.class, Oracle.class },
    value = { "INFORMATION_SCHEMA.SYSTEM_TABLES WHERE table_name =
'SYSTEM_TABLES'", "DUAL" })
  public String dual;
```

```
dao.selectMap("select /*$bean.sysDate*/SYSDATE from
/*$bean.dual*/DUAL ", args("bean", bean));
```

```
select CURRENT_TIMESTAMP from INFORMATION_SCHEMA.SYSTEM_TABLES
WHERE table_name = 'SYSTEM_TABLES'
```

```
select SYSDATE from DUAL
```

LimitOffset

```
List<Map<String, Object>> l =
  dao.selectMap(
    sql,
    args(LimitOffset.KEYWORD,new LimitOffset(10, 0)));
```

```
List<Map<String, Object>> l =
  dao.selectMap(
    sql,
    args(LimitOffset.KEYWORD,new Paging(10, 0)));
```

limit, offset

limit, pageNumber

publicフィールド対応

- も、してあります

S2Dao拡張

- @S2DaoExtアノテーション付けるだけ
- アノテーションが無ければS2Dao
- 全部の機能は使えない
  - selectだけ

```
public interface TestDao {
  @S2DaoExt
  @Arguments("deptno")
  List<DeptHasEmpList> selectByKimDao(int deptno);
}
```

使い方

- 大きく2つ
  - 直接使う
  - S2Daoを拡張して使う

今後

- S2Daoにフィードバックしたい
- 機能追加はあまり考えていない
  - 特殊なDBでも拡張可能だし、SQL書けばよいので
  - 近いうちにVer1.0.0をリリース
  - S2JDBCのようにJava上でSQLを組み立てるような仕組みは提供は予定していない
    - SQLエディタを使えば
      - 補完できる
      - SQLの即時実行もできる
    - Eclipseプラグインの紹介
      - DBViewer Plugin
      - http://sourceforge.jp/projects/dbviewer/

プロダクト情報

- 名前
  - kimu-dao
- URL
  - http://code.google.com/p/kimu-dao/

まとめ

- S2Daoでは不可能な機能を提供
- S2Daoの仕組み、知見が利用できる
  - 例えば、doltengを使った自動生成
    - Entity
    - Dao
    - Sql
- 「Java5以降導入された機能」の可能性

終わり

- ご清聴ありがとうございました