DB-API 材料データ基盤 DB_クリープ試験検索機能追加

リリース ***1.0.0***

SIP-MI(A4)

2020 年 **02** 月 **26** 日

目次:

第 1 章

材料データ基盤 DB view 作成手順

1.1 システム構成

OS	CentOS 7.4
mySQL	5.7.20

1.2 関連ソース

<https://github.com/materialsintegration/DB-API/tree/master/sample/sample1/src/material_DB>

ファイル	内容
create_view_creep_test.sql	クリープ試験 view を作成するスクリプト

1.3 view 作成

1.3.1 システムログイン

材料データ基盤 DB サーバの mySQL に管理者権限にてログインする。(ログイン情報は材料データ基盤 DB のドキュメントを参照のこと)

ログインしたら、カレント DB を指定する。

```
mysql> use material;
```

1.3.2 ストアド登録

関連ファイル (create_view_creep_test.sql) を所定の場所に配置しておく。
sql コマンドラインから以下実行する。

```
mysql> source <create_view_creep_test.sql(フルパス)>
```

1.3.3 ストアド実行

ストアドプロシージャを呼び出す。

```
mysql> call create_view_creep_test()
```

1.3.4 view 確認

以下コマンド実施し、view の生成を確認する。

```
mysql> select * from view_creep_test;
```

データが表示されれば OK。

第 2 章

DB-API nodejs サンプルコード作成手順

2.1 概要

本手順書では、材料データ基盤 DB(MySQL) に対し、DB-API を用いて検索を行うための手順を示す。

2.2 システム構成

OS	CentOS 7.4
mySQL	v11.15.0

2.3 関連ソース

<https://github.com/materialsintegration/DB-API/tree/master/sample/sample1/src/DB-API>

ファイル	内容
get_test_NIMS_material.js	エンドポイントスクリプト
sql_view_test.js	検索用 SQL ファイル

2.4 DB-API nodejs 構成

DB-API の nodejs コード構成ファイルは以下。

```
[nodejs]
  |
  |- [DBAPI]
       |
       |- [app]
            |
            |- [db]                                  : DB に関するソースフォルダ
            |    |
            |    |- [granta]                         : GRANTA に関するソースフォルダ
            |    |
            |    |- dic_dblist.js                    : 検索対象の DB のリスト定義ファイル
            |    |- Mysql.js                         : mysql のモジュール定義ファイル
            |    |- Postgresql.js                    : postgresql のモジュール定義ファイル
            |
            |- [router]
            |    |
            |    |- [v1]
            |         |
            |         |- [get]
            |              |
            |              |- [element]              : 組成に関するエンドポイントフォルダ
            |              |- [property]             : 特性に関するエンドポイントフォルダ
            |              |- [structure]            : 構造に関するエンドポイントフォルダ
            |              |- [test]                 : 試験に関するエンドポイントフォルダ
            |                   |
```

(次のページに続く)

(前のページからの続き)

```
|                   |                   |- [NIMS_material]    : 材料データ基盤 DB に関するエンドポイントフォルダ
|                   |                   |     |
|                   |                   |     |- get_test_NIMS_material.js : エンドポイントスクリプト
|                   |                   |     |- sql_view_test.js : 検索用 sql ファイル
|                   |                   |
|                   |                   |- [GRANTA]           : GRANTA に関するエンドポイントフォルダ
|                   |                   |     |
|                   |                   |     |- get_test_GRANTA.js : エンドポイントスクリプト
|                   |
|                   |- app.js                                 : nodejs インデックスファイル
```

* DB に関する情報は [db] フォルダ以下に格納する。
* エンドポイントに関するコードは [get] フォルダ以下にサブフォルダを切って配置する。
* routing 情報を app.js に記述する。

2.5 nodejs 編集

2.5.1 エンドポイント作成

検索処理を登録するためのエンドポイントを作成する。

上記 nodejs 構成において、以下のフォルダを作成する。

nodejs/DBAPI/app/router/v1/get/test/NIMS_material

作成したフォルダに以下ファイルを配置する。

* get_test_NIMS_material.js
* sql_view_test.js

その後、以下ファイルを編集する。

```
$ vi ~/nodejs/DBAPI/app/app.js

> (63 行目以降) 追記
> var getTest_NIMSmaterial = require(router_get_path +
>     '/test/NIMS_material/get_test_NIMS_material.js');
>
> // routing
> app.use(rooturl + '/test/NIMS_material/', getTest_NIMSmaterial);
```

* getTest_NIMSmaterial にエンドポイントのパスとスクリプトを記述。
* app.use にて追加したエンドポイントスクリプトを定義する。

2.6 サービス再起動

nodejs のコード修正、ファイル追加等を行った場合、サービスを再起動する。

```
# systemctl restart nodejs
```

2.7 動作確認

ブラウザを起動し、以下 url をリクエストする。

```
http://<db-api サーバ>/db-api/v1/get/test/NIMS_material/?mimetype=json&test=creep_
↪rupture_test
```

DB 検索結果が表示されれば OK。

2.8 参考

2.8.1 検索用 sql スクリプト

エンドポイントパスの検索 sql 用スクリプトは以下。

(sql_view_test.js)

```
var sql = '';
sql = sql + 'select ';
sql = sql + '    vms.*, ';
sql = sql + '    vme.*, ';
sql = sql + '    vt.* ';
sql = sql + 'from __view__ vt ';
sql = sql + 'left join view_material_subset1 vms ';
sql = sql + 'on vt.material_id = vms.material_id ';
sql = sql + 'left join view_material_element vme ';
sql = sql + 'on vt.material_id = vme.material_id ';
sql = sql + 'order by vt.test_piece_id ';

module.exports = sql;
```

* 本 sql スクリプトにて、各 DB へ問い合わせを行う。
* 「__view__」は、置換用文字列 (問い合わせ先 view を動的に指定するもの)
  詳細は get_test_NIMS_material.js を確認のこと。

2.8.2 リクエスト処理スクリプト

上記で作成したエンドポイントパスに、指定した処理スクリプトを配置する。
必要に応じ、コード修正すること。

(get_test_NIMS_material.js)

```
// ------------------------------
// import library
// ------------------------------
const express  = require('express');
const app      = express();
const router   = express.Router();
const json2csv = require('json2csv');
const boom     = require('@hapi/boom');
const log4js   = require('log4js');

//const logErrors    = require('../../../error').logErrors;
const errorHandler = require('../../../error');

// ------------------------------
// local variable
// ------------------------------
//const node_root_path = '../../../../..';
const mid_db_path = process.cwd() + '/db';
const dblist = require(mid_db_path + '/dic_dblist.js');

const db = 'NIMS_material';

const mimetypes = ['json', 'csv'];
//const tests = ['tensile test', 'fatigue test', 'creep rupture test'];
const tests = {
    'creep_rupture_test': {
        view: 'view_creep_test',
        sql: 'sql_view_test.js',
        key: 'test_piece_id'
    },
    'tensile_test': {
        view: 'view_tensile_test',
        sql: 'sql_view_test.js',
        key: 'test_piece_id'
    }
```

(次のページに続く)

(前のページからの続き)

```
};

// ------------------------------
// local function
// ------------------------------

// ------------------------------
// main
// ------------------------------

// log setting
log4js.configure(process.cwd() + '/config/log4js.config.json');
const systemLogger = log4js.getLogger('system');
const httpLogger   = log4js.getLogger('http');
const accessLogger = log4js.getLogger('access');

router.use(require("../../../logger.js"));
router.use(log4js.connectLogger(accessLogger));
router.use((req, res, next) => {
    if (typeof req === 'undefined' || req === null ||
        typeof req.method === 'undefined' || req.method === null ||
        typeof req.header === 'undefined' || req.header === null) {
        next();
        return;
    }
    if (req.method === 'GET') {
        httpLogger.info(req.query);
    } else {
        httpLogger.info(req.body);
    }
    next();
});


// get /dbabi/v[n]/get/:table
router.get('/', function(req, res, next) {
    var query = req.query;

    // check query param
    var mimetype = 'json';
    if (query['mimetype']) {
        mimetype = req.query.mimetype;
    }
    var test = '';
    if (query['test']) {
        test = req.query.test;
    }

    // validation
    var msg = '';
    if (mimetypes.indexOf(mimetype) == -1) {
        msg = 'invalid mimetype';
        return next(boom.badRequest(msg));
    };
    if (test in tests) {
        msg = '';
    } else {
        msg = 'invalid test';
        return next(boom.badRequest(msg));
    };

    // import sql
    var sql_import = require('./' + tests[test]['sql']);
    var sql = sql_import.replace('__view__', tests[test]['view']);

    // get connection
    var dbobj = null;
    dblist.forEach(function( d ) {
        if (db != d.name) return

        switch (d.dbtype) {
            case 'mysql':
                var Mysql = require(mid_db_path + '/' + d.middle)
                dbobj = new Mysql(d.name)

                break
            case 'postgresql':
                var Postgresql = require(mid_db_path + '/' + d.middle)
                dbobj = new Postgresql(d.name)

                break
            default:
                console.log('no database type match')
```

(次のページに続く)

(前のページからの続き)

```
                return
                break
        }
    });

    // execute sql
    if (sql != '') {
        try {
            dbobj.getRecordset(sql, function(flds, result, next) {
                // output
                var output = '';
                if (mimetype == 'json') {
                    output = result;
                    res.setHeader('Content-Type', 'application/json');
                } else if (mimetype == 'csv') {
                    output = json2csv.parse(result, flds);
                    res.setHeader('Content-Type', 'text/csv; charset=UTF-8');
                }

                res.send(output);
            })

        } catch (error) {
            return next(error);
        }
    } else {
        res.send('');
    }
});

// errorHandler
//router.use(logErrors);
router.use(errorHandler);

module.exports = router;
```

* クエリパラメータ test の検索対象項目を 25 行目の tests に定義している。
  項目追加、変更はここを編集する。
* クエリパラメータの validation を 84 行目から実施している。
  パラメータ追加時に変更すること。
* sql ファイルの読み込みを 97 行目にて行っている。
  view 名置換もここで実施している。