xdataset_json 가이드

Last update: 2024/05/23

www.softbase.co.kr

목차

xdataset_json 가이드

데이터셋 및 XTranMap을 JSON문자열로 변환하고, JSON문자열을 데이터셋으로 변환하는 처리에 대한 샘플 화면이다.

예시

템플릿 위치: /HTML5/COMPONENT/XDATASET/xdataset_json

템플릿 파일

- xdataset_json.xml
- xdataset_json.js
- 새창으로 실행

화면 스크립트

```
function screen_on_load()
{
    grd01.setcheckedrow(1,true);
    grd01.setcheckedrow(3,true);

    var nRow = grd02.additemex(0, "aaaa", true, true);
    grd02.setitemtext(nRow, 1, "1111");
    nRow = grd02.additemex(0, "bbbb", true, true);
    grd02.setitemtext(nRow, 1, "2222");
}



/**
 * 화면에서 저장할 데이터셋 정보 배열을 리턴하는 콜백함수이다.
  * @return arrXDatasetInfoStr 저장 대상 데이터셋 정보 문자열 배열 또는 null (데이터셋 정보 문자열 형식은 아래 참조)
        데이터셋 정보 문자열 형식: 데이터셋ID + [":" + 데이터셋컬럼IDs + [":" + 레코드상태 + [":" + 레코드체크상태]]]
        데이터셋 이름이 공백 문자열인 경우 화면내 모든 데이터셋에 대해 처리한다.
        데이터셋 컬럼 이름이 공백 문자열인 경우, 데이터셋내 모든 컬럼에 대해 처리한다.
        데이터셋 컬럼 이름이 여러개인 경우, 콤마(",")로 구분하여 공백없이 나열한다.
        레코드 상태 정보가 없거나 공백문자열인 경우, "ALL" 처리한다.
        레코드 상태 정보가 "ALL"인 경우, 행 상태가 NORMAL/INSERT/UPDATE/DELETE MARK인 행만 저장된다.
        레코드 상태 정보가 "UPDATE"인 경우, 행 상태가 INSERT/UPDATE/DELETE MARK인 행만 저장된다.
        레코드 체크 상태 정보가 없거나 공백문자열인 경우, "ALL" 처리한다.
        레코드 체크 상태 정보가 "ALL"인 경우, 모든 행에 대해 처리한다.
```

```
       레코드 체크 상태 정보가 "CHECK"인 경우, "체크된 행만 저장된다.
       레코드 상태 정보와 레코드 체크 상태 정보는 AND 조건으로 비교된다.
       예시1: ["데이터셋이름1::"UPDATE|CHECK","데이터셋이름2:ALL","데이터셋이름3"]
// 저장 대상 데이터 셋 이름 배열 (이름에 ":"가 없는 경우, ALL로 처리)
       예시2: ["::UPDATE"]          // 화면내 모든 데이터셋에 대해 UPDATE로 처리
       예시3: ["::ALL"]             // 화면내 모든 데이터셋에 대해 ALL로 처리
 * @return JSON 문자열
 */
function SaveXDatasetToJson()
{
    return ["xDataSetID_01:::"];
    //return ["xDataSetID_01::UPDATE:CHECK"];
}

function btnXDataSetToJson_on_mouseup(objInst)
{
    var arrXDatasetInfoStr = SaveXDatasetToJson();
    var strJson = XDatasetListToJDatasetList(screen, arrXDatasetInfoStr);

    fld01.settext(strJson);
    return true;
}



/**
 * 대상 화면내 트랜맵에 연결된 입력 데이터셋 오브젝트 정보를 추출하여 JSON 문자열을 리턴한다.
 * @param objScreen 데이터셋 저장 대상 트랜맵이 포함된 screen 오브젝트
 * @param strXTranmapId 저장 대상 트랜맵 ID
 * @return JSON 문자열
 */
function SaveXDatasetOfXTranmapToJson()
{
    return "xTranMapID_01";
}

function btnXTranmapToJson_on_mouseup(objInst)
{
    var strXTranmapId = SaveXDatasetOfXTranmapToJson();
    var strJson = XTranmapToJDatasetList(screen, strXTranmapId);

    fld01.settext(strJson);
    return true;
}



function btnInitXDataSet_on_mouseup(objInst)
{
  // 데이터셋 초기화
    xDataSetID_01.init();
    xDataSetID_02.init();
}
```

```
function btnJSonToXDataset_on_mouseup(objInst)
{
    // JSON문자열을 데이터셋으로 변환
    JsonDatasetListToXDatasetList(screen, fld01.gettext());

    return true;

}









/*************************************************************************
************************/
/* 이하는 공통모듈 영역                                                    */

/**
 * XDataSet Object의 내용을 각행의 내용이 JSON 형식으로 구성된 어레이로 변환하여 리턴한다.
 * @param objXDataSet 변환할 XDataSet 오브젝트
 * @return JSON 데이터 어레이
 */
function getJsonDataArrayFromXDataSet(objXDataSet)
{
    var nRowCount = 0;
    var nColumnCount = 0;
    var nRow, nColumn;
    var arrColumnName = [];
    var arrJsonData = [];

  // 데이터셋의 행 및 칼럼의 수를 구함.
    nRowCount = objXDataSet.getrowcount();
    nColumnCount = objXDataSet.getcolumncount();

  // 데이터셋의 칼럼 이름을 어레이로 구함.
    for(nColumn = 0; nColumn < nColumnCount; nColumn++) {
        arrColumnName.push(objXDataSet.getcolumnid(nColumn));
    }

  // 데이터 셋의 각 행을 JSON 데이터롤 변환하고 어레이에 추가함
    for(nRow = 0; nRow < nRowCount; nRow++) {
        var objJsonData = new Object();
        for(nColumn = 0; nColumn < nColumnCount; nColumn++) {
            var strValue = objXDataSet.getdata(nRow, nColumn);
            objJsonData[arrColumnName[nColumn]] = strValue;
        }
        arrJsonData.push(objJsonData);
    }

    return arrJsonData;
```

```
}

/**
 * XDataSet Object의 내용을 각행의 내용이 JSON 형식으로 구성된 어레이로 변환하여 리턴한다.
 * @param objXDataSet 변환할 XDataSet 오브젝트
 * @param nRow 변환할 XDataSet의 Row Index (Zero-Base)
 * @return JSON 데이터 오브젝트
 */
function getJsonDataFromXDataSetRow(objXDataSet, nRow)
{
    var nColumnCount = 0;
    var nColumn;
    var arrColumnName = [];

    var objJsonData = new Object();

  // 데이터셋의 행 및 칼럼의 수를 구함.
    nRowCount = objXDataSet.getrowcount();

    if(nRow >= nRowCount) {
        return objJsonData;
    }

    nColumnCount = objXDataSet.getcolumncount();

  // 데이터셋의 칼럼 이름을 어레이로 구함.
    for(nColumn = 0; nColumn < nColumnCount; nColumn++) {
        arrColumnName.push(objXDataSet.getcolumnid(nColumn));
    }

  // 데이터 셋의 지정 행을 JSON 데이터롤 변환
    for(nColumn = 0; nColumn < nColumnCount; nColumn++) {
        var strValue = objXDataSet.getdata(nRow, nColumn);
        objJsonData[arrColumnName[nColumn]] = strValue;
    }

    return objJsonData;
}

/**
* 데이터셋 정보 JSON 문자열을 이용하여 데이터을 대상 화면에 복원한다.
 * @param objScreen 데이터셋 복원 대상 screen 오브젝트
 * @param strJson 데이터셋 복원 정보 JSON 문자열
 * @param bCreate 데이터셋 미존재시 데이터 셋 생성 처리 여부
 * @param bClear 기존 데이터를 데이터를 삭제할지 여부
 * @param bSetOp 레코드 상태 복원 여부
 */
function JsonDatasetListToXDatasetList(objScreen, strJsonData, bCreate,
bClear, bSetOp)
{
    var nDatasetCount, nPrevRowCount, nDataset, nRow, nCol, objXDataset,
```

```
arrRowData, objJDataset;
    var objScreenDataset;

    // JSON 문자을열 파싱하여 JSON 오브젝트를 구함, JSON 오브젝트 형식은 아래
XDatasetToJson 함수 참조
    objScreenDataset = factory.jsonparse(strJsonData);

    factory.consoleprint("objScreenXDataset.header.screenid = " +
objScreenDataset.header.screenid);
    factory.consoleprint("objScreenXDataset.header.screenurl = " +
objScreenDataset.header.screenurl);

    // JSON 데이터셋 배열 갯수를 구함
    nDatasetCount = objScreenDataset.dataset_arr.length;
    factory.consoleprint("nDatasetCount = " + nDatasetCount);

    // JSON 데이터셋 갯수만큼 Loop
    for (nDataset = 0; nDataset < nDatasetCount; nDataset++) {
        // JSON 데이터셋 오브젝트를 구함
        objJDataset = objScreenDataset.dataset_arr[nDataset];
        factory.consoleprint("objJDataset.id = " + objJDataset.id);

    // 화면에서 데이터셋 ID에 해당하는 XDataSet 오브젝트를 구함
        objXDataset = objScreen.getxdataset(objJDataset.id);

    // 화면에서 XDataSet 오브젝트을 구하지 못한 경우
        if (factory.isobject(objXDataset) == false) {
            // bCreate 파라미터가 false인 경우, continue
            if (bCreate == false) {
                factory.consoleprint("objXDataset is null");
                continue;
            }
            // bCreate 파라미터가 true인 경우, XDataSet 신규 생성
            else {
                // XDataSet 신규 생성
                objScreen.addxdataset(objJsonXDataset.id);

         // 화면에서 데이터셋 ID에 해당하는 XDataSet 오브젝트를 구함
                objXDataset = objScreen.getxdataset(objJsonXDataset.id);

         // 화면에서 XDataSet 오브젝트을 구하지 못한 경우
                if (factory.isobject(objXDataset) == false) {
                    factory.consoleprint("Fail to create XDataset");
                    continue;
                }

         // 생성된 XDataSet에 칼럼 생성
                factory.consoleprint("objJsonXDataset.column_count = " +
objJsonXDataset.column_count);
                for (nCol = 0; nCol < objJsonXDataset.column_count; nCol++)
{
```

```
objXDataset.addcolumn(objJsonXDataset.column_id_arr[nCol], "", 0);
                }
            }
        }

        // XDataSet 셋 정보 초기화
        if (bClear == true) { objXDataset.init(); }

    // 기존 행 갯수를 구함
        nPrevRowCount = objXDataset.getrowcount();
        factory.consoleprint("nPrevRowCount = " + nPrevRowCount);

        factory.consoleprint("objJDataset.column_count = " +
objJDataset.column_count);

    // 컬럼 이름에 대한 컬럼 인텍스 값 배열 설정 (아래 setdata 함수를 사용하기 위해서)
        column_index_arr = [];
        for (nCol = 0; nCol < objJDataset.column_count; nCol++) {
            factory.consoleprint(objJDataset.column_id_arr[nCol] + ", " +
objXDataset.getcolumn(objJDataset.column_id_arr[nCol]));
column_index_arr.push(objXDataset.getcolumn(objJDataset.column_id_arr[nCol])
);
        }

    // 행열 컬럼 이름 배열 설정
        factory.consoleprint("objJsonXDataset.row_count = " +
objJDataset.row_count);
        for (nRow = 0; nRow < objJDataset.row_count; nRow++) {
            arrRowData = objJDataset.row_data_arr[nRow];

            // XDataSet에 행 추가
            objXDataset.insertrow(nRow + nPrevRowCount);

       // 추가된 행에 컬럼 값 설정
            for (nCol = 0; nCol < objJDataset.column_count; nCol++) {
                factory.consoleprint("column_index_arr[nCol] = " +
column_index_arr[nCol] + ", arrRowData[nCol] = " + arrRowData[nCol]);
                objXDataset.setdata(nRow + nPrevRowCount,
column_index_arr[nCol], arrRowData[nCol]);
            }

       // 행 OP 값 설정
            if (bSetOp == true) {
                objXDataset.setrowoperation(nRow + nPrevRowCount,
objJDataset.row_opcode_arr[nRow]);
            }

            // objXDataset.setcheckedrow(nRow,
objJDataset.row_check_arr[nRow]);
        }
    }
```

```
    // factory.consoleprint(XDatasetToJson(objScreen, null));

    return;
}

/**
 * 대상 화면내 데이터셋 오브젝트 정보를 추출하여 JSON 문자열을 리턴한다.
 * @param objScreen 데이터셋 저장 대상 데이터셋이 포함된 screen 오브젝트
 * @param arrXDatasetInfoStr 저장 대상 데이터셋 정보 문자열 배열 또는 null (데이터셋 정보 문자열 형식은 아래 참조)
       데이터셋 정보 문자열 형식: 데이터셋ID + [":" + 데이터셋컬럼IDs + [":" + 레코드상태 + [":" + 레코드체크상태]]]
       데이터셋 이름이 공백 문자열인 경우 화면내 모든 데이터셋에 대해 처리한다.
       데이터셋 컬럼 이름이 공백 문자열인 경우, 데이터셋내 모든 컬럼에 대해 처리한다.
       데이터셋 컬럼 이름이 여러개인 경우, 콤마(",")로 구분하여 공백없이 나열한다.
       레코드 상태 정보가 없거나 공백문자열인 경우, "ALL" 처리한다.
       레코드 상태 정보가 "ALL"인 경우, 행 상태가 NORMAL/INSERT/UPDATE/DELETE MARK인 행만 저장된다.
       레코드 상태 정보가 "UPDATE"인 경우, 행 상태가 INSERT/UPDATE/DELETE MARK인 행만 저장된다.
       레코드 체크 상태 정보가 없거나 공백문자열인 경우, "ALL" 처리한다.
       레코드 체크 상태 정보가 "ALL"인 경우, 모든 행에 대해 처리한다.
       레코드 체크 상태 정보가 "CHECK"인 경우, "체크된 행만 저장된다.
       레코드 상태 정보와 레코드 체크 상태 정보는 AND 조건으로 비교된다.
       예시1: ["데이터셋이름1::"UPDATE|CHECK","데이터셋이름2:ALL","데이터셋이름3"]
// 저장 대상 데이터 셋 이름 배열 (이름에 ":"가 없는 경우, ALL로 처리)
       예시2: ["::UPDATE"]          // 화면내 모든 데이터셋에 대해 UPDATE로 처리
       예시3: ["::ALL"]             // 화면내 모든 데이터셋에 대해 ALL로 처리
 * @return JSON 문자열
 */
function XDatasetListToJDatasetList(objScreen, arrXDatasetInfoStr)
{
    var nXDatasetInfoCount, nXDatasetInfo, objJDataset;
    var arrXDatasetInfo, strXDatasetId, strUpdateOnly, strCheckOnly,
bUpdateOnly, bCheckOnly;

    var objScreenDataset = {
        header : {
            screenid: "",        // 화면 ID
            screenurl: ""        // 화면 URL
        },
        dataset_arr : []        // objJDataset 오브젝트 배열 (objJDataset 형식은 XDatasetToJDataset 함수 참조)
    };

    objScreenDataset.header.screenid = objScreen.getscreenid();
    objScreenDataset.header.screenurl = objScreen.getscreenurl();

    // arrXDataasetName 파라미터 값이 null인 경우, 화면의 모든 데이터 셋 정보 처리
    if (arrXDatasetInfoStr == null) {
        arrXDatasetInfoStr = objScreen.getxdatasetids();
    }
```

```
    nXDatasetInfoCount = arrXDatasetInfoStr.length;
    factory.consoleprint("nXDatasetInfoCount = " + nXDatasetInfoCount);

    for (nXDatasetInfo = 0; nXDatasetInfo < nXDatasetInfoCount;
nXDatasetInfo++) {
        // XDataset 정보를 ":" 문자로 스플릿
        arrXDatasetInfo = arrXDatasetInfoStr[nXDatasetInfo].split(":");

        strXDatasetId = arrXDatasetInfo[0];

        arrXDatasetColumnId = null;
        if (arrXDatasetInfo.length > 1 && arrXDatasetInfo[1].length > 0) {
            arrXDatasetColumnId = arrXDatasetInfo[1].split(",");
            if (arrXDatasetColumnId.length == 0) { arrXDatasetColumnId =
null; }
        }

        // bUpdateOnly, bCheckOnly 값 설정 처리
        strUpdateOnly = "ALL";
        if (arrXDatasetInfo.length > 2) { strUpdateOnly =
arrXDatasetInfo[2]; }

        strCheckOnly = "ALL";
        if (arrXDatasetInfo.length > 3) { strCheckOnly = arrXDatasetInfo[3];
}

        bUpdateOnly = false;
        bCheckOnly = false;

        if (strUpdateOnly == "UPDATE") { bUpdateOnly = true; }
        if (strCheckOnly == "CHECK") { bCheckOnly = true; }

        factory.consoleprint("bUpdateOnly = " + bUpdateOnly + ", bCheckOnly
= " + bCheckOnly);

    // 하나의 XDataSet을 JDataSet으로 변환
        objJDataset = XDatasetToJDataset(objScreen, strXDatasetId,
arrXDatasetColumnId, bUpdateOnly, bCheckOnly);
        if (objJDataset != null) {
       // 하나의 JSON 데이터셋 정보 오브젝트(objJDataset)를 dataset_arr 배열에 추가
            objScreenDataset.dataset_arr.push(objJDataset);
        }
    }

    factory.consoleprint(factory.jsonstringify(objScreenDataset));

    return factory.jsonstringify(objScreenDataset);
}

/**
 * 대상 화면내 데이터셋 오브젝트 정보를 추출하여 JSON 문자열을 리턴한다.
```

```
 * @param objScreen 데이터셋 저장 대상 데이터셋이 포함된 screen 오브젝트
 * @param strXDatasetId 데이터셋 ID
 * @param arrColumnName 데이터셋 컬럼 이름 배열 또는 null (null 인 경우, 모든 컬럼 처리
함)
 * @param bUpdateOnly true이 경우, 행 상태가 INSERT/UPDATE/DELETE MARK인 행만 처리
 * @param bCheckOnly true이 경우, 체크된 행만 처리
 * @return JSON 데이터셋 오브젝트
 */
function XDatasetToJDataset(objScreen, strXDatasetId, arrColumnId,
bUpdateOnly, bCheckOnly)
{
    var nRowCount, nColumnCount, nRow, nCol, objXDataset, arrRowData,
objJDataset;
    var strRowOp, bCheckRow, arrColumnIndex, nColumnIndex;

  // 하나의 데이터셋 정보를 저장할 JSON 오브젝트
    objJDataset = {
        id: "",                     // 데이터셋 ID
        row_count: 0,               // 행 갯수
        row_opcode_arr: [],         // 행별 OP 코드 배열
        row_check_arr: [],          // 행별 체크 상태 배열
        column_count: 0,            // 열 갯수
        column_id_arr: [],          // 열 ID 배열
        row_data_arr: []            // 행 데이터 배열
    };

    factory.consoleprint("strXDatasetId = " + strXDatasetId);
    if (arrColumnId == null) { factory.consoleprint("arrColumnId is null");
}
    else { factory.consoleprint("arrColumnId.length = " +
arrColumnId.length); }
    factory.consoleprint("bUpdateOnly = " + bUpdateOnly + ", bCheckOnly = "
+ bCheckOnly);

  // 화면에서 XDataSet 오브젝트를 구함
    objXDataset = objScreen.getxdataset(strXDatasetId);
    if (factory.isobject(objXDataset) == false) {
        factory.consoleprint("Invalid XDataset ID = " + strXDatasetId);
        return null;
    }

  // 데이터셋의 모든 컬럼 이름을 저장하는 배열 생성
    if (arrColumnId == null) {
        arrColumnId = [];
        nColumnCount = objXDataset.getcolumncount();
        for (nCol = 0; nCol < nColumnCount; nCol++) {
             arrColumnId.push(objXDataset.getcolumnid(nCol));
        }
    }

    nColumnCount = arrColumnId.length;
```

```
    factory.consoleprint("nColumnCount = " + nColumnCount);

    objJDataset.id = strXDatasetId;
    factory.consoleprint("objJDataset.id = " + objJDataset.id);

    for (nCol = 0; nCol < nColumnCount; nCol++) {
        factory.consoleprint(nCol + ": " + arrColumnId[nCol]);
    }

  // 컬럼 이름 배열 설정 및 컬럼 이름에 대한 컬럼 인덱스 값 배열 설정 (아래 getdata 함수를 사용하기 위해서)
  // (유효하지 않은 컬럼 이름 제외 처리)
    arrColumnIndex = [];
    for (nCol = 0; nCol < nColumnCount; nCol++) {
        nColumnIndex = objXDataset.getcolumn(arrColumnId[nCol]);
        if (nColumnIndex >= 0) {
            objJDataset.column_id_arr.push(arrColumnId[nCol]);
            arrColumnIndex.push(nColumnIndex);
        }
    }

    // XDataSet 컬럼 갯수를 구함
    objJDataset.column_count = arrColumnId.length;
    factory.consoleprint("objJDataset.column_count = " + objJDataset.column_count);

  // 행열 데이터 설정
    nRowCount = objXDataset.getrowcount();
    for (nRow = 0; nRow < nRowCount; nRow++) {
        strRowOp = objXDataset.getrowoperation(nRow);
        bCheckRow = objXDataset.ischeckedrow(nRow);

        factory.consoleprint(nRow + ": strRowOp = [" + strRowOp + "], bCheckRow = " + bCheckRow);

        if (bUpdateOnly == true && strRowOp == XFD_ROWOP_NONE) { continue; }
        if (bCheckOnly == true && bCheckRow == false) { continue; }

    // 행의 OP 코드 저장
        objJDataset.row_opcode_arr.push(strRowOp);
        objJDataset.row_check_arr.push(bCheckRow);

    // 한 행의 데이터를 보관할 배열 생성
        arrRowData = [];
        for (nCol = 0; nCol < objJDataset.column_count; nCol++) {
            arrRowData.push(objXDataset.getdata(nRow, arrColumnIndex[nCol]));
        }

    // 한 행의 데이터 배열(arrRowData)를 행 데이터 베열에 추가
        objJDataset.row_data_arr.push(arrRowData);
    }
```

```
    // JSON 데이터셋 행 갯수 값 설정
    objJDataset.row_count = objJDataset.row_data_arr.length;
    factory.consoleprint("objJDataset.row_count = " +
objJDataset.row_count);

    return objJDataset;
}

/**
 * 대상 화면내 트랜맵에 연결된 입력 데이터셋 오브젝트 정보를 추출하여 JSON 문자열을 리턴한다.
 * @param objScreen 데이터셋 저장 대상 트랜맵이 포함된 screen 오브젝트
 * @param strXTranmapId 저장 대상 트랜맵 ID
 * @return JSON 문자열
 */
function XTranmapToJDatasetList(objScreen, strXTranmapId)
{
    var nInputKind;
    var arrXDatasetId, nXDatasetIdCount, nXDatasetId, strXDatasetId,
objJDataset, bUpdateOnly, bCheckOnly;

    var objScreenDataset = {
        header : {
            screenid: "",        // 화면 ID
            screenurl: ""        // 화면 URL
        },
        dataset_arr : []        // objJDataset 오브젝트 배열 (objJDataset 형식
은 XDatasetToJDataset 함수 참조)
    };

    if (strXTranmapId == null || strXTranmapId.length == 0) {
        return null;
    }

    objScreenDataset.header.screenid = objScreen.getscreenid();
    objScreenDataset.header.screenurl = objScreen.getscreenurl();

    arrXDatasetId = objScreen.getxtranmapinputxdataset(strXTranmapId, true);
    nXDatasetIdCount = arrXDatasetId.length;
    factory.consoleprint("nXDatasetIdCount = " + nXDatasetIdCount);

    for (nXDatasetId = 0; nXDatasetId < nXDatasetIdCount; nXDatasetId++) {
        strXDatasetId = arrXDatasetId[nXDatasetId];

        arrXDatasetColumnId = objScreen.getxtranmapinputcols(strXTranmapId,
strXDatasetId);
        nInputKind = objScreen.getxtranmapinputkind(strXTranmapId,
strXDatasetId);
        switch(nInputKind) {
            case 1 :    // UPDATE
                bUpdateOnly = true;
                bCheckOnly = false;
```

```
                break;
            case 3 :    // ALL
                bUpdateOnly = false;
                bCheckOnly = false;
                break;
            case 4 :    // CHECK
                bUpdateOnly = false;
                bCheckOnly = true;
                break;
            default :
                bUpdateOnly = true;
                bCheckOnly = true;
                break;
        }

    // 하나의 XDataSet을 JDataSet으로 변환
        objJDataset = XDatasetToJDataset(objScreen, strXDatasetId,
arrXDatasetColumnId, bUpdateOnly, bCheckOnly);
        if (objJDataset != null) {
       // 하나의 JSON 데이터셋 정보 오브젝트(objJDataset)를 dataset_arr 배열에 추가
            objScreenDataset.dataset_arr.push(objJDataset);
        }
    }

    factory.consoleprint(factory.jsonstringify(objScreenDataset));

    return factory.jsonstringify(objScreenDataset);
}
```