DHIS2 APP for Data import from Excel Files

History

This app was originally developed by HISP Vietnam <http://hisp.vn/> in collaboration with the Malaria Department in WHO. We forked the code in order to apply slight changes in the core management of the Excel file and configure the models of our Excel files.

Overview

As the main part of the app was developed outiside our team there are still some features we are still under investigation to really work on them. The orginal code does a lot of things, it tracks erros, show summaries, save logs but we are currently using the app practically and only importing the data.

The process of reading the excel files is manage by the excel-reader javascript library (available in the code). From uploading the file to save data in DHIS2 system the data goes trhough this steps:

* The file is uploaded in the input element in the main page **index.html** with **id=uploadFile** which only accepts .xlsx and .xls files.
* Once the file is selected the function **changeFile() (index.html)** is triggered.
* The function reads the file and triggers the function **handleFile() (index.html).**
* From there the data flows to the file **funcxl.js** to the functions **process\_wb() -> to\_formulae().**
* In the function **to\_formulae()** is where we associate the dta extracted from the file with a variable called **resultArray** and the function **processExcellSheet() (main.js)** is triggered.
* The function **processExcellSheet()** is the core of the digestment of the excel and is where we use the metadta definitions previously stored in a variable **templateObject** to properly read the file.
* The variable **templateObject** is filled with the model of the Excel file read from **template\_bu02.js (CHANGE).**
* For each sheet indicated by **templateObject** the function read which type of data has inside and prepare an import to the DHIS2 API.
* Note that to access the data fo the excel fiel the function **processExcelSheet** does not use the variable **resultArray** directly. The process use external functions (**getCellData() or getLastRowNumer())** to encapsualte the queries to the excel file.
* Once all the data is gathered and ready for the import the function **importData() or importEventData()** is triggered depending on the model of the excel.

Function getCellData()

Args: Number of sheet, Sheet cell  
Process: It access the variable resultArray where the first index is for the sheets and the second is for the sheet cells. A sheet cels must be in the format = A6, B89, DF87.

Function processExcelSheet()

TODO

SHEET TYPES

AGGREGATED\_EVENT

Excel files with aggreated and individual data in the same sheet.

* Collects the year and month from the excell sheet (only one cell per each).
* Collects tye organisation unit from the sheet (onlye one cell).
* Validates the year and the month.
* Prepares the data value with period (year + month), dataElement, coc, orgUnit and value.
* Pushes all the data values to a list and import that list.
* Looks for individual data in the same sheet.
* From the first line to the last line of the individual line-listing it imports the eventDataValues.

Example of a configuration file:

{

sheet\_type: "AGGREGATE\_EVENT",

sheet\_no: 2,

oucode\_cell: "D7",

month\_cell: "I5",

year\_cell: "I4",

last\_data\_column : "ZZ",

agg\_des:

[

{ deuid: "EaISNtwxqUl", cocuid: "TKUeFoc6Ogt", cell\_no: "E9"}

],

program: "Km4KB0dcgxu",

key\_coulmn: "A", *//for validation*

event\_date\_col: "O", *//should be a letter*

data\_starting\_row: "39", *//should be a number*

event\_des:

[

{ deuid: "R6TMiedD6Vt", cocuid: "SdOUI2yT46H", column: "B"},

{ deuid: "CxxjS8wT5jY", cocuid: "S34ULMcHMca", column: "C"},

{ deuid: "iBlf6ogCGDH", cocuid: "jOkIbJVhECg", column: "D"}

}

AGGREGATED\_STATIC

Excel files with only aggregated data

* Collects the year and month from the excell sheet (only one cell per each).
* Collects tye organisation unit from the sheet (onlye one cell).
* Validates the year and the month.
* Prepares the data value with period (year + month), dataElement, coc, orgUnit and value.
* Pushes all the data values to a list and import that list.

Example of a configuration file:

{

sheet\_type: "AGGREGATE\_EVENT",

sheet\_no: 2,

oucode\_cell: "D7",

month\_cell: "I5",

year\_cell: "I4",

last\_data\_column : "ZZ",

agg\_des:

[

{ deuid: "EaISNtwxqUl", cocuid: "TKUeFoc6Ogt", cell\_no: "E9"}

],

}

MULTIPLE\_DE\_OU\_PE

Import of Excel files with aggregated data from different organisation units and periods in the same sheet. A table with aggregated metadata as headers but each rows referes to certain organisation unit and a certain period.

* From the first row of data to the last one it collects all the aggregated values, organisation unit and period, and import them.
* Providing a SINGLE\_OU mode it will only ask once the organistation unit in the provided cell
* You can configure that some of the cells in the excel aggregate to a single data value

Example of a configuration file:

{

name: "Rabies template",

file: "rabies\_template.xlsx",

sheets: [

{

sheet\_type: MULTIPLE\_DE\_OU\_PE,

sheet\_no: 1,

orgUnitIdScheme: ORG\_CODE\_SCHEME,

dataElementIdScheme: ORG\_UID\_SCHEME,

idScheme: ORG\_UID\_SCHEME,

mode: "SINGLE\_OU",

oucode\_col: "B3",

month\_col: "A",

year\_col: "A",

data\_starting\_row: 11,

data\_ending\_row: 13,

last\_data\_column: "M",

key\_column: "A",

attr\_oc: "QIFdMRrb22m",

agg\_des: [

*//HUMAN CASES, AGE, DIAGNOSIS, GENDER AND TRANSMISSION*

*//AGE UNKNOWN*

{deuid: "Ujm8qoGj0fo", cocuid: "kek1YXjDq70", col\_no: "B"},

*//DIAGNOSIS UNKNOWN*

{deuid: "it8UvG94yDu", cocuid: "a14nFkqYZ9i", col\_no: "B"},

*//GENDER UNKNOWN*

{deuid: "wJvwACm3zzR", cocuid: "jNbFhhnUsQv", col\_no: "B"},

*//TRANSMISSION UNKNOWN*

{deuid: "RmmNGZSrcLa", cocuid: "U8blRfoPG9x", col\_no: "B"},

*//BITE CASES IN HUMANS, GENDER, AGE, CATEGORIES AND ANIMAL*

*//GENDER UNKNOWN*

{deuid: "UgOTAOvaLbD", cocuid: "jNbFhhnUsQv", col\_no: "D,E,F,G"},

*//AGE UNKNOWN*

{deuid: "nnCmUHUtUCS", cocuid: "kek1YXjDq70", col\_no: "D,E,F,G"},

]

}

]

}

EVENTS

Importing individual line-listings.

* From the first row with data to the last one it creates an event per row.
* One program per file
* Each row contains the orgUnit, date, and all the values
* To know the dataElement to which each value refers the process looks into the first row of the file.

Example of configuration file:

{

name: "BU02 data template",

file:"BU02\_template.xlsx",

sheets: [

{

sheet\_type: EVENTS,

sheet\_no: 1,

data\_starting\_row : "5",

key\_column : "B",

ou\_col : "B",

program : "lAu94BiaY5s",

event\_date\_col: "C", *//should be a letter*

event\_des\_length : "33",

}

]

}

MULTIPLE\_PERIODS\_AND\_FACILITIES