Table of Contents

ISSUE TRACKER FEATURES AND FUNCTIONALITIES

1. ISSUES DATA TRANSFORMATION ACTIONS

You can load your issues data from different sources into different targets, whenever those sources and targets comply with the syntax and format of the issue tracker.
A single call performing the transformation of one issues source data into another taget data instance artifact are called actions.
This section contains the description of this feature-set per action.

1.1. Run the txt-to-db action

You can load you issues from an "issues txt file" , having a specic syntax into a PosgtreSQL issue table, by issueing the shell.
This call with truncate the issue table from the db and convert all the issues data from the issues txt file into the issue table.

```
# pre-load the vars of an issue-tracker project
doParseIniEnvVars /vagrant/csitea/cnf/projects/issue-tracker/issue-tracker-issues.dev.doc-pub-host.cnf

# ensure there is no data in the issue table
psql -d "$db_name" -c 'TRUNCATE TABLE issue ;'

# run the txt-to-db action
bash src/bash/issue-tracker/issue-tracker.sh -a txt-to-db

# check the data by :
psql -d "$db_name" -c 'SELECT issue_id , category , name FROM issue order by name'
```

1.1.1. Run the txt-to-db action period handling

Issues txt files are stored in a daily folder with the following naming convention:
<<project>>.<<current_date>>.<<period>>.txt
The tool knows to correctly fetch the issues files for the configured period (by export period=weekly) and copy its data in to the <<period>>_issue table.

```
ysg-issues.2017-06-03.daily.txt
ysg-issues.2017-06-03.monthly.txt
ysg-issues.2017-06-03.weekly.txt
ysg-issues.2017-06-03.yearly.txt
```

1.2. Run db-to-xls action against postgres

You can unload your already stored ANY xls table with unique id's and load them into a xls file.

```
# pre-load the vars of an issue-tracker project
doParseIniEnvVars /vagrant/csitea/cnf/projects/issue-tracker/issue-tracker-issues.dev.doc-pub-host.cnf



# check the data by :
psql -d "$db_name" -c 'SELECT issue_id , start_time , stop_time , category , name FROM issue order by prio'

# run the db-to-xls action
bash src/bash/issue-tracker/issue-tracker.sh -a db-to-xls
```

1.3. Run the xls-to-db action

You can load the latest produced xls file (note as long as your xls sheet headers match the columns in your db table ANY xls is compatible)
You can control whether or not the loadable table should be truncted by setting the do_truncate_tables environment variable to 1 or 0.

```
# check the data by :
psql -d "$db_name" -c 'SELECT issue_id , start_time , stop_time , category , name FROM issue order by prio'
```

```
# run the db-to-xls action
bash src/bash/issue-tracker/issue-tracker.sh -a xls-to-db

# check the updated data
psql -d "$db_name" -c '
SELECT issue_id , start_time , stop_time , category , name FROM issue order by start_time'
```

1.3.1. Run the xls-to-db action without passing xls file

if you do not provide a xls file the newest xls file from the mix data dir will be used

1.3.2. Run xls-to-db in nested-set mode against mysql

You could run the xls-to-db action against mysql or mariadb rdbms so that the issue-tracker will arrange your table to be compatible with the nested-set hierarchy model.

```
export load_model=nested-set
export rdbms_type=mysql # or mariadb
export tables=Feature,Test
bash src/bash/issue-tracker/issue-tracker.sh -a xls-to-db -t $tables
```

1.4. db-to-txt action

You can load your already stored in the issue table issues and load them into the same issues txt file

```
# check the data by :
psql -d "$db_name" -c '
SELECT issue_id , start_time , stop_time , category , name FROM issue order by prio'

# run the db-to-xls action
bash src/bash/issue-tracker/issue-tracker.sh -a db-to-txt

# check the updated data
psql -d "$db_name" -c '
SELECT issue_id , start_time , stop_time , category , name FROM issue order by start_time'
```

1.4.1. db-to-txt action with pre-defined sorting attribute

You can load your already stored in the issue table issues and load them into the same issues txt file by using a pre-defined sorting attribute.

```
export issues_order_by_attribute=start_time

bash src/bash/issue-tracker/issue-tracker.sh -a db-to-txt
```

1.5. run-pgsql-scripts

You can create a preconfigured <<env>>_<<db_name>> postgres via a single shell call. The scripts will fail if any of the sql scripts have a syntax error - all the ddl events will be displayed in the STDOUT and stored in the shell log file for later audit

1.6. run-mysql-scripts

You can create a preconfigured <<env>>_<<db_name>> in mariadb via a single shell call. The scripts will fail if any of the sql scripts have a syntax error - all the ddl events will be displayed in the STDOUT and stored in the shell log file for later audit

1.7. generate-docs

You can generate all the md and pdf docs via single shell call by issuing the following command:

```
bash src/bash/issue-tracker/issue-tracker.sh -a generate-docs
```

2. DEVOPS FEATURES AND FUNCTIONALITIES

2.1. Testability

The issue-tracker app has a good and improving all the time test coverage. You can all the tests in the application as follows:

2.1.1. Perl syntax check for the whole project

You can check the perl syntax for each perl code file in the whole project by issing the following shell call:

```
bash src/bash/issue-tracker/issue-tracker.sh -a check-perl-syntax
```

2.1.2. Testability for unit testing

You can run the unit tests of the application by issuing the following single shell call:

```
bash src/bash/issue-tracker/issue-tracker.sh -a run-perl-unit-tests
```

2.1.3. Testability for integration testing

You can run the integration tests of the application by issuing the following single shell call:

```
bash src/bash/issue-tracker/issue-tracker.sh -a run-perl-integration-tests
```

2.2. Logging

Logging is implemented as follows:

2.2.1. Bash logging

The issue-tracker.sh bash entry point loggs both to STDOUT and file. You cold easily defined your own log levels.

```
doLog "INFO an info msg"
[INFO ] 2018.05.08-21:05:25 EEST [issue-tracker][@host-name] [29667] an info msg:
```

2.2.2. Perl logging

The perl logger could be configured to log to file and std outputs.

```
doLog "INFO an info msg"
[INFO ] 2018.05.08-21:05:25 EEST [issue-tracker][@host-name] [29667] an info msg:
```

2.3. development efficiency increasing actions

2.3.1. morph-dir action

You can recursively search and replace strings in both file and dir paths and their contents (as soon as they non-binary , txt files) by issuing the following commands:

```
export to_srch=WriterTxtDaily
export to_repl=WriterTxtTerm
export dir_to_morph=`pwd`
bash src/bash/issue-tracker/issue-tracker.sh -a morph-dir
fg
history | cut -c 8-
```

2.3.2. work against different projects

The issue-tracker could be used against many different projects as soon as they have the needed file and dir structure , configuration file and dedicated db in the PostgreSQL.

```
# pre-load the vars of an issue-tracker project
doParseIniEnvVars /vagrant/csitea/cnf/projects/issue-tracker/issue-tracker-issues.dev.doc-pub-host.cnf
```

2.3.3. issue-tracker tool perl code syntax check

You can check the perl code syntax with the following command:

```
bash src/bash/issue-tracker/issue-tracker.sh -a check-perl-syntax
```

2.4. Web based routes

2.4.1. single item data fetch in json via

web

You can get the data of a single item in db by guid in json format via the web interface , for example: http://doc-pub-host:3000/dev_stockit_issues/get/company_eps/727cf807-c9f1-446b-a7fc-65f9dc53ed2d

```
# run for the loaded items
while read -r guid ; do
curl "http://doc-pub-host:3000/dev_stockit_issues/get/company_eps/$guid" ;
done < <(psql -d dev_stockit_issues -t -c "
SELECT guid FROM company_eps")
```

2.5. Documentation related

2.5.1. Single call export of the md and pdf documentation files

Single call export of the md and pdf documentation files

3. BACK-END FEATURES AND FUNCTIONALITIES

3.1. select-tables web action

An http-client could get the select of all the tables of a database to which the issue-tracker has connectivity to (that is not only the one configured in the application layer)

```
<<web-host>>:<<web-port>>/<<database>>/select-tables
```

3.1.1. successfull execution

An http-client could get the select of all the tables of a database to which the issue-tracker has connectivity to

```
// 20180505205212
// http://192.168.56.120:3000/dev_issue_tracker/select-tables

{
  "dat": {
    "1": {
      "row_id": "1",
      "table_catalog": "dev_issue_tracker",
      "table_name": "confs",
      "table_schema": "public"
    },
    "2": {
      "row_id": "2",
      "table_catalog": "dev_issue_tracker",
      "table_name": "daily_issues",
      "table_schema": "public"
    },
    "3": {
      "row_id": "3",
      "table_catalog": "dev_issue_tracker",
      "table_name": "decadally_issues",
      "table_schema": "public"
    }
  },
  "msg": "SELECT tables-select OK ",
  "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select-tables",
  "ret": 200
}
```

3.1.2. error handling for failed connect to db in the select-tables web action

If the http-client points to a db to which the app layer does not have a connection (might be a non-existing one) the proper response is generated.

```
// 20180503234141
// http://192.168.56.120:3000/non_existent/select/daily_issues
```

```
{
 "msg": "cannot connect to the non_existent database: FATAL:  database \"non_existent\" does not exist",
 "req": "GET http://192.168.56.120:3000/non_existent/select/daily_issues",
 "ret": 400
}
```

3.2. select web action

An http-client could get the contents of ANY table of a database to which the issue-tracker has connectivity to (ie not only the one configured in the application layer but also other databases in the same postgres instance) by using the following syntax:

```
<<web-host>>:<<web-port>>/<<database>>/select/<<table-name>>
```

3.2.1. successfull execution

An http-client could get the contents of ANY table of a database to which the issue-tracker has connectivity to by calling the following url:
<<web-host>>:<<web-port>>/<<database>>/select/<<table-name>>

3.2.2. apply multiple operators on the select properly

All the operators bellow could be combined and the result set is the one "translated" with the AND operator in the back-end side.

3.2.3. error handling for failed connect to db in the select web action

If the http-client points to a db to which the app layer does not have a connection (might be a non-existing one) the proper response is generated.

```
// 20180503234141
// http://192.168.56.120:3000/non_existent/select/daily_issues

{
 "msg": "cannot connect to the non_existent database: FATAL:  database \"non_existent\" does not exist",
 "req": "GET http://192.168.56.120:3000/non_existent/select/daily_issues",
 "ret": 400
}
```

3.2.4. error handling for non-existent table in the select-tables web action

if a table does not exist a proper error msg containing response is generated.

```
// 20180505205015
// http://192.168.56.120:3000/dev_issue_tracker/select/non_existent

{
 "msg": " the table non_existent does not exist in the dev_issue_tracker database ",
 "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/non_existent",
 "ret": 400
}
```

3.2.5. filter functionality in select table web action

The response cold be filtered by ANY attribute with any valid value.

```
// using the following syntax:
<<web-host>>:<<web-port>>/<<database>>/select/<<table-name>>?fltr-by=<<filter-attribute-to-filter-by>>&fltr-val=<<filter-value-to-filter-by>>
```

3.2.5.1. successfull execution

The response of the select web action could be filtered by using the syntax bellow:
Those are eventual transated to a where clause in the db select part.

```
// 20180505204531
// http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?fltr-by=prio&fltr-val=1
```

```
{
  "dat": {
    "c89d3283-0a9f-4b8d-9dcc-84a63e64276b": {
      "actual_hours": null,
      "category": "issue-tracker-features",
      "description": "add the web select controller “\r\n - implementation code\r\n - tests \r\n - documentation additions for :\r\n-- requirements\r\n--
userstories\r\n-- tests \r\n-- features and functionalities",
      "guid": "c89d3283-0a9f-4b8d-9dcc-84a63e64276b",
      "id": 180402,
      "level": 2,
      "name": "add the web select controller",
      "owner": "ysg",
      "planned_hours": "3.00",
      "prio": 1,
      "seq": 1,
      "start_time": "2018-04-02 18:00",
      "status": "07-qas",
      "stop_time": null,
      "tags": "feature",
      "type": "feature",
      "update_time": "2018-05-04 23:18:45.104771"
    }
  },
  "msg": "SELECT OK for table: monthly_issues",
  "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?fltr-by=prio&fltr-val=1",
  "ret": 200
}
```


3.2.5.2. error handling for wrong filtering syntax by missed fltr-by or fltr-va url params

If the request does not have either one of the url params the following response is produced.

```
// 20180505204734
// http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?fltr-by=prio

{
  "msg": "mall-formed url params for filtering - valid syntax is ?fltr-by=<<attribute>>&fltr-val=<<filter-value>>",
  "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?fltr-by=prio",
  "ret": 400
}
```

3.2.5.3. error handling for unexisting filter name

If the syntax is correct but an unexisting filtering attribute is provided (that is the table columns and the attriute name do not match) the following error msg is returned:

```
// 20180506220030
// http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?fltr-by=foo&fltr-val=sdklfj

{
  "msg": "the foo column does not exist",
  "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?fltr-by=foo&fltr-val=sdklfj",
  "ret": 400
}
```

3.2.6. pick functionality in select table web action

Works for both a single colum and a comma separated select of columns. Obeys the following syntax

```
// using the following syntax:
<<web-host>>:<<web-port>>/<<database>>/select/<<table-name>>?pick=col1,col2,col3
```

3.2.6.1. successfull execution

if the request contains the "pick" url parameter only the picked column values are selected.

```
// 20180506230955
```

```
// http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?pick=name,prio

{
 "dat": {
  "0daa3447-42f5-4792-aca2-bd1cb06e2a78": {
   "guid": "0daa3447-42f5-4792-aca2-bd1cb06e2a78",
   "name": "define REST API response structure",
   "prio": 3
  },
  "3c3aff5d-8246-4893-acc4-4853904f1d40": {
   "guid": "3c3aff5d-8246-4893-acc4-4853904f1d40",
   "name": "add the pick in url to select in db reader control flow for Select.pm controller",
   "prio": 3
```

3.2.6.2. error handling if a picked column does not exist

if a picked column does not exist the following error is displayed.

```
// 20180506230926
// http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?pick=non_existent_column

{
 "msg": "the non_existent_column column does not exist",
 "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?pick=non_existent_column",
 "ret": 400
}
 },
 "msg": "SELECT OK for table: monthly_issues",
 "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?pick=name%2Cprio",
 "ret": 200
}
```

3.2.7. Use filtering with the like operator in select table web action

The response cold be likeed by ANY attribute with any valid value.

```
// using the following syntax:
<<web-host>>:<<web-port>>/<<database>>/select/<<table-name>>?like-by=<<like-attribute-to-like-by>>&like-val=<<like-value-to-like-by>>
```

3.2.7.1. successfull execution for number types types

The like operator could be used with numbers as well.

```
// 20180508191656
// http://192.168.56.120:3000/dev_issue_tracker/select/yearly_issues?like-by=prio&like-val=1&pick=name,prio

{
 "dat": {
  "46533749-1c00-4688-9cdd-1cc276ca40ac": {
   "guid": "46533749-1c00-4688-9cdd-1cc276ca40ac",
   "name": "implement upsert in DbWriterPostgres",
   "prio": 21
  },  "msg": "SELECT OK for table: monthly_issues",
 "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?like-by=prio&like-val=1",
 "ret": 200
}
```

3.2.7.2. successfull execution for text types

The response of the select web action could be likeed by using the syntax bellow:
Those are eventual transated to a where clause in the db select part.

```
// 20180505204531
// http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?like-by=prio&like-val=1

{
```

```
 "dat": {
   "c89d3283-0a9f-4b8d-9dcc-84a63e64276b": {
     "actual_hours": null,
     "category": "issue-tracker-features",
     "description": "add the web select controller “\r\n - implementation code\r\n - tests \r\n - documentation additions for :\r\n-- requirements\r\n--
userstories\r\n-- tests \r\n-- features and functionalities",
     "guid": "c89d3283-0a9f-4b8d-9dcc-84a63e64276b",
     "id": 180402,
     "level": 2,
     "name": "add the web select controller",
     "owner": "ysg",
     "planned_hours": "3.00",
     "prio": 1,
     "seq": 1,
     "start_time": "2018-04-02 18:00",
     "status": "07-qas",
     "stop_time": null,
     "tags": "feature",
     "type": "feature",
     "update_time": "2018-05-04 23:18:45.104771"
   }
 },
 "msg": "SELECT OK for table: monthly_issues",
 "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?like-by=prio&like-val=1",
 "ret": 200
}
```


3.2.7.3. error handling for wrong syntax in the filtering by the like operator by missed like-by or like-val url params

If the request does not have either one of the url params the following response is produced.

```
// 20180505204734
// http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?like-by=prio

{
 "msg": "mall-formed url params for likeing - valid syntax is ?like-by=<<attribute>>&like-val=<<like-value>>",
 "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?like-by=prio",
 "ret": 400
}
```

3.2.7.4. error handling for unexisting like table's attribute

If the syntax is correct but an unexisting like operator's attribute is provided (that is the table columns and the attriute name do not match) the following error msg is returned:

```
// 20180506220030
// http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?like-by=foo&like-val=sdklfj

{
 "msg": "the foo column does not exist",
 "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?like-by=foo&like-val=sdklfj",
 "ret": 400
}
```