Table of Contents

ISSUE TRACKER FEATURES AND FUNCTIONALITIES

1. DEVOPS FEATURES AND FUNCTIONALITIES

1.1. 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:

1.1.1. Perl syntax check call

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
```

1.1.2. Unit tests call

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
```

1.1.3. Integration tests call

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
```

1.1.4. Bash tests call

You can run all the bash functions in the tool by issuing the following command in the shell.

```
# if you set the previous 2 actions as those to be tested
bash src/bash/issue-tracker/test-issue-tracker.sh

2018-05-12 18:01:08      START test-issue-tracker test run report

result  start-time  stop-time   action-name
  ok    18:01:09 18:01:19 run-perl-unit-tests
  ok    18:01:20 18:01:30 run-perl-integration-tests



2018-05-12 18:01:30      STOP  test-issue-tracker test run report
```

1.2. Logging

Logging is implemented as follows:

1.2.1. 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:
```

1.2.2. 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:
```

1.3. development efficiency increasing actions

1.3.1. 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-
```

1.3.2. 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
```

1.3.3. 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
```

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

Single call export of the md and pdf documentation files

2. UI FEATURES

2.1. Common listing features

This section describes all the common listing related features.

2.1.1. Succesfull execution

You can use the pick=col1,col2,col3 url parameter to select for only desired attributes.
You could filter the result the same way the filters for the select page work (see bellow).

2.1.2. Error handling for unexisting db

If the db provided in the url pattern does not exist an error is shown.

2.1.3. Error handling for unexisting table

If the table requested does not exist an error is shown.

2.1.4. Error handling for unexisting column

If the column requested does not exist an error is shown.

2.1.5. The "pick" url param

You can use the pick=col1,col2,col3 url parameter to select for only desired attributes to be show in the ui control used for listing

2.1.6. The "hide" url param

If you do not specify any attribute to pick, you could hide specific attributes by using the "hide=col1,col2,col3" syntax

2.1.7. The "with=col-operator-value" filter

You can filter the result of the query by using the "with=col-operator-value"

2.2. List labels page

The list labels page lists all the rows from any table in the app db in form of "labels" - rectangular forms containing all the selected attributes (by default all) and their values.

2.3. List as table page

The list table page lists all the rows from any table in the app db in form of simple ui table , which is sortable by clicking with ..

2.3.1. table sorting

The listed table is sortable by clicking on the columns OR by navigating with the tab key on the keyboard on a column and hitting Enter.

2.3.2. table filtering

You can filter the already presented part of the result set in the page by using the search textbox. This is only an ui type of filtering for the already loaded data. To filter from the db use the where or

2.3.3. set table paging size

You can set the page size of the result set to be fetched from the database by using the "&page-size=<<page-size>>" url parameter or by clicking on the page sizes links right of the table search box.

2.3.4. set table page number

If the result-set requested is larger than the page size you can go to the next page number by using the "&page-num=<<page-num>>" url parameter.
You could go to the next page number by clicking on the links after the last row.

3. SHELL BASED 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.

3.1. The 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'
```

3.1.1. The 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
```

3.2. The 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
```

3.3. 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'
```

3.3.1. The 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

3.3.2. The xls-to-db action with 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 tables=Tests,ItemController,ItemModel,ItemView,ExportFile,UserStory,Requirement,DevOps,Feature,ReadMe,SystemGuide,Image,ExportFile;
export do_truncate_tables=1 ; export rdbms_type=mysql ; export load_model=nested-set ; perl src/perl/issue_tracker/script/issue_tracker.pl --do
xls-to-db --tables $tables
```

3.4. The 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'
```

3.4.1. 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
```

3.5. 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

3.6. 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

3.7. generate-docs

You can generate all the md and pdf docs by if you have a running instance of the isg-pub application accessible via single shell call by issuing the following command:

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

4. BACK-END FEATURES AND FUNCTIONALITIES

4.1. 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
```

4.1.1. 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
}
```

4.1.2. 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
}
```

4.2. 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>>
```

4.2.1. 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>>

4.2.2. 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.

4.2.3. 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
}
```

4.2.4. 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
}
```

4.2.5. 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>>
```

4.2.5.1. 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
}
```


4.2.5.2. 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
}
```

4.2.5.3. 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
}
```

4.2.6. Use pick url param 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
```

4.2.6.1. 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
```

4.2.6.2. 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
}
```

4.2.7. 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>>
```

4.2.7.1. 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
}
```

4.2.7.2. 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
}
```


4.2.7.3. 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
}
```


4.2.7.4. 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
}
```


4.2.8. Filtering by using the "with" url param

You can filter the result of the query by using the "with=col-operator-value"


```
// 20180605150216
// http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?as=cloud&with=prio-lt-6&o=prio&pick=name,prio&with=status-eq-02-todo

{
  "dat": [
    {
      "guid": "55a06b10-7bbf-4298-a1ee-804bbfd12570",
      "name": "poc for data-load testing",
      "prio": 5
    }
  ],
  "msg": "",
  "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?as=cloud",
  "ret": 200
```

```
}
```

4.2.8.1. Successful path

The application selects only the rows matching the generated where <<column>> <<operator>> <<value>> , where the operator could be also the like operator (set also if the value contains the % char)

4.2.8.2. Error handling for unexisting column

If the column used does not exist an error is provided as follows:

```
// 20180605150010
// http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?as=cloud&with=prio-lt-7&o=prio&pick=name,prio&with=foo-eq-02-todo

{
  "msg": "the foo column does not exist",
  "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?as=cloud",
  "ret": 400
}
```

4.2.9. the hide operator in the select web action

Whenever a hide=<<col-name>> operator is applied the values and the column names of the column to hide are not displayed in the result. Use command to as the separator for listing multiple columns to hide.

```
// using the following syntax:
<<web-host>>:<<web-port>>/<<database>>/select/<<table-name>>?hide=guid,prio
```

4.2.9.1. successfull execution for text types

The response of the url query presents all but the hidden attribute values.

```
// 20180511144541
// http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?like-by=category&like-
val=features&pick=name,prio,start_time,stop_time,category,status&o=stop_time&hide=guid

{
 "dat": [
   {
    "category": "issue-tracker-features",
    "name": "improve integration testing",
    "prio": 5,
    "start_time": null,
    "status": "09-done",
    "stop_time": null
   }
 ],
 "msg": "",
 "req": "GET http://192.168.56.120:3000/dev_issue_tracker/select/monthly_issues?like-by=category&like-
val=features&pick=name%2Cprio%2Cstart_time%2Cstop_time%2Ccategory%2Cstatus&o=stop_time&hide=guid",
 "ret": 200
}
```

4.2.9.2. error handling for unexistent column to hide

If the column which values are requested to be hidden does not exist the proper error message is retrieved.

```
// 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
}
```