Migrate a DOC Application from 4.0.0 FP3 to 4.0.0 FP4

October 30, 2020

Migration Steps

1. Export all the scenarios that should be kept using the Web Application for instance in excel format.

    1.1. Delete them using the UI Since we need to reset Data Service Database after migration Scenario Service will reference scenarios which will not have the related data

2. Export Application Configuration in a json file

3. Stop services

```
$ cd deployment/docker/infra
$ docker-compose down
```

4. Remove Postgres database volume Warning! Be sure to have exported scenarios data you need to keep

    4.1. List all the docker volumes and locate the one named <project>-pgdata

```
$ docker volume ls
DRIVER              VOLUME NAME
local               52d4bd725f3d6c27af141135e38a163d965980a1ac46e889de66a71c1d9d24a8
local               ac3ca8c89d4c2a4983ad575d1db3b16609db4a2c291edbdc40fa00e2d154919f
local               capacity-planning-mongovolume
local               capacity-planning-pgdata
local               f725b649b661f6637086c070fc9a884b7ea9d412479b86abf17fcf9fe927318f
```

    4.2. Delete this volume

```
$ docker volume rm capacity-planning-pgdata
capacity-planning-pgdata
$
```

5. Proceed to a full project generation with 4.0.0-FP4 version to a clean folder using following procedure :

   5.1. create a new folder called “fp4”

   5.2. copy in that folder

   - the `entities.jdl` file that describes your data model (it can be found in `gene-model/src/main/resources`)
   - the `generator.sh` file provided with the platform(see section Script for Application Generation of the Installation section of the documentation
   - the `.yo-rc.json` file produced when you originally generated your application.

     Example of `.yo-rc.json` file:

```
{
  "@gene/generator-gene": {
    "promptValues": {
      "projectName": "aircraft-maintenance",
      "projectTitle": "Aircraft Maintenance Optimization",
      "destination": ".",
      "package": "com.aircraftmaintenance",
      "collectorClass": "AircraftMaintenance",
      "inputFileType": "jdl",
      "inputFile": "./entities.jdl",
    }

  }
}
```

If you do not have `.yo-rc.json` file, create one using the following information:

- `projectName`: this is the `xxx` value that has been used to prefix sub-projects like xxx-libs or xxx-services
- `projectTitle`: this is the title you have defined for your project toolbar
- `package`: this is the java package you have defined
- `collectorClass`: this is the java class name you have defined. You can find it at the beginning of your entities.jdl file

Generate An Empty Application

Use the `generator.sh` command to generate your application.

```
$ cd fp4


$ ls -al
  .yo-rc.json
  entities.jdl
  generator.sh


$ ./generator.sh -v 4.0.0-fp4
```

6. Be sure to have a clean starting point with no unversioned files in your existing fp3 project repository and all changes committed git status

should show no changes and no untracked files, if necessary you can use “git clean -dfx” to clean all files not present on the current branch.

7. Delete all files and folders from the folder keeping .git and .gradle .idea and other potentially useful hidden folders , you can do that with “rm -Rf *” which will not include hidden directories

8. Copy all the files generated at step 5 in the now empty folder of your existing FP3 project

9. Run ./gradlew clean docker at the root of the project which should execute all generation and build steps

10. At this stage the build should succeed but you have a project that does not contain all the custom code.

    In the git status you should now see some new untracked files, those you can add, they are new and needed by the application.

    Obviously you have many changed files, you should go through them resolving the conflicts with your previous code. At this stage all your custom files appear as deleted, you can now revert those files and make the application compile with them.

    Folders you should have the most work in should be

    /extensions

    /web.

    See the detailed code changes in this document to guide you in making the project compile again.

11. Restart infrastructure services with “docker-compose up -d”

12. Once conflicts are resolved and the web interface can be launched re-import the application configuration and the scenarios using the UI.

Dependencies changes

Version Changes

Backend

- Jackson: 2.9.9 → 2.11.0
- Postgresql 10.0 → 12.0
- OpenAPI generator: 4.2.2 → 4.3.1

Frontend

- @angular/cdk: 7.0.0 → 9.1.0
- @ng-select/ng-select: 4.0.0 → 4.0.4
- angular-tree-component@8.5.6 → @circlon/angular-tree-component@9.0.3
- echarts: 4.4.0 → 4.8.0
- ngx-color-picker: 8.2.0 → 9.1.0
- ngx-moment: 3.4.0 → 5.0.0
- ngx-toastr: 12.0.1 → 12.1.0
- moment: 2.24.0 → 2.27.0
- lodash: 4.17.15 → 4.17.19
- corejs: 2.4.1 → 3.6.5

New Dependencies

Backend

- mapstruct: 1.3.1.Final
- univocity-parsers: 2.8.4
- kotlin: 1.3.50

Python

- python: 3.6
- pandas: 1.1.0
- xvik/gradle-use-python-plugin: 2.2.0

- pip: 20.1
- setuptools: 46.1.3
- wheel: 0.34.2
- twine: 3.1.1pytest: 5.4.1
- pytest-cov: 2.10.0

Public classes/api changes

1. Field “groups” removed from following classes, it is not relevant anymore due to the new permissions system :

- com.decisionbrain.gene.application.model.GeneUser
- com.decisionbrain.scenario.dto.GeneUserDTO
- GeneUser from @gene/core (Typescript)

2. Argument scope removed from:

- all scenario-service endpoints defined in controllers under com.decisionbrain.scenario.controller

- related api clients, generated under com.decisionbrain.gene.scenario.api

UI and ScenarioService “Path” events

1. Event types emitted by com.decisionbrain.scenario.service.notification.NotificationService on the scenario-server websocket topic were removed :

```
PATH_NODE_LOCKED

PATH_NODE_UNLOCKED

PATH_NODE_OWNER_UPDATED

PATH_NODE_GROUP_UPDATED

PATH_NODE_PERMISSIONS_UPDATED

PATH_NODE_GROUP_AND_PERMISSIONS_UPDATED

PATH_NODE_MODIFIED
```

```
PATH_NODE_DELETED

PATH_NODE_CREATED
```

As a consequence in the Web UI they are not processed and (re)-emitted by GeneScenarioNotificationsService anymore.

If you relied on those events you have to use new events introduced in 4.0.0-FP4 which are more specific to the application element type :

```
SCENARIO_LOCKED,

SCENARIO_UNLOCKED,

SCENARIO_MODIFIED,

SCENARIO_DELETED,

SCENARIO_RESTORED,

SCENARIO_CREATED,


VIEW_DASHBOARD_RESTORED,

WORKSPACE_RESTORED,


WORKSPACE_DELETED,

WORKSPACE_CREATED,

WORKSPACE_MODIFIED,


VIEW_DASHBOARD_DELETED,

VIEW_DASHBOARD_CREATED,

VIEW_DASHBOARD_MODIFIED,


APPLICATION_SETTINGS_CHANGED
```

```
APPLICATION_PERMISSIONS_CHANGED
```

2. More information on elements in GeneScenarioEvent

The “uuids” member is deprecated, starting with 4.0.0-FP4 GeneScenarioEvents will provide a new member “elements” containing an array of objects providing uuid, name and parentUuid of the impacted objects

```
class GeneScenarioEvent {
       // @Deprecated, use 'elements' member with IdWithNameAndParent
       public uuids: string[],
       public type: GeneScenarioEventType,
       public userId: string = null,
       public userName: string = null,
       public pathNodes: PathNode[] = [],
       public data: any = null,
       // since FP4
       public elements: IdWithNameAndParent[] = []
       }


      interface IdWithNameAndParent {
       /**
         * A string that is not empty, used as an identifier
         */
       uuid: string;
       /**
```

```
          * The element\'s name
          */
        name?: string;
        /**
          * The element\'s parent uuid
          */
        parentUuid?: string;
}
```

Widgets

1. New job button

The job button and its configurator have been updated and enriched with additional features such as defining wether parameters should be locked, asked upon execution etc.

As a consequence the widgets configurations may no longer be valid, and you should have to edit the widgets configurations again with this new version.

2. Keycloak

New “manage-users” role was added to the scenario service, for the realm-management client.

This role allows the scenario service to modify attributes of a user. This is used to detect the first user’s login into the application.

Code generation change

A breaking change was introduced due to a bugfix on the handling of entities with consecutive uppercase characters.

Here an example of the change. Consider A business model entity with the following property : MYTestUTILITYFunction

In previous versions this name would convert to the following column name: m_ytest_ut_il_it_yfunction

WIth FP4 it will be : my_test_utility_function

It also impacts generated DbDomObjects and JPA entities fields

mYTestUTILITYFunction -> myTestUTILITYFunction

Performance Improvement-related changes

Collector Serialization

Collector serialisation using CSV format which uses postgresql COPY operations

- To use CSV format, you need to use withFormat() on ScenarioDataExpression (see example below).
- To use CSV format as result from DBOS worker, you need to use (see exemple below):
    - in the task definitionwithOutputScenario("outputCollector", ScenarioDataFormat.CSV,..)
    - in the Worker implementation emitOutputCollector(output, collector, DbDomCollectorSerializerFormat.CSV);
- Performance improvements in deleting a complete scenario.
- Performance improvements in duplicating a complete scenario.
- Improvements in Memory/Cpu usage with collectors.

Note that DBRFC serialization of collectors remains the default serialisation mechanism, so that upgrading to FP4 without modifying Task / Worker should not have any impact. CSV is the the default serialisation method in FP4.

Date Database Mapping

JPA Instant attributes are now mapped to SQL BIGINT instead of TIMESTAMP to avoid formatting and parsing Date String (see bellow script to migrate existing databases).

Improvements in Criteria API Usage

The criteria API is used to implement all our generic queries (GraphQL or REST), this release brings two changes that reduce the execution time of all queries.

- Reduce over-fetching by using Multi Select instead of Select. Instead of fetching objects, we now fetch individual object properties.
- Fix wrong construction of Join queries when filters are set leading to duplicate CROSS JOIN Sql query generation, instead of single LEFT JOIN
- As a consequence DataService REST API is now aligned with GraphQL and no longer returns plain DTO objects but dynamic JSON content (through JsonNode and JsonArray objects). requestedFieldNames is now a mandatory parameter for all the endpoints.

Usage of Postgres block range indexes BRIN

So far, the generated default indexes were using standard btree indexes for *db_gene_internal_scenario_id*. However our data have common properties that fit very well with BRIN indexes:

- same values for all rows of a same scenario,
- rows of a same scenario are likely to be contiguous because written most of the time by bulk (collector read/write)

Usage of BRIN indexes reduces both indexes size and query response time.

Data Table Partitioning

Introduced Postgresql Partitioning by LIST. The idea of partitioning is to sustain the same level of performance on queries no matter of how many rows / scenarios are stored in the database. When the size of tables grows, indexes grow as well.

When performing queries, the pgsql execution engine has to reduce as fast as possible the rows involved. It uses as part of this process indexes fully loaded in working_memory. When the table size hits the working memory limit then the engine has to process the query using batch operations which heavily impacts response times.

Also when performing INSERT & DELETE postgres uses one trigger per FK involved * rows involved.

By splitting tables per scenario we make sure we limit the number of rows to a certain volume depending on the project data size.

Migration to Postgresql 12

By introducing Data Table Partitioning, which is a breaking change in the initially generated SQL, we also upgraded to the latest stable version of Postgresql which improves greatly partition management when having thousands of partitions.

It also improves COPY performances with large datasets

https://www.2ndquadrant.com/en/blog/postgresql-12-partitioning/

- NOTE: Postgresql data server user now requires SUPERUSER privileges (see in deployment *create-user.sh*)