Developer Guide

Table of Contents

Contribution

Contribution

There are several ways in which you may contribute to this project.

- File issues
- Submit a pull requests

Found a bug or missing feature?

Please file an issue in our issue tracking system.

Submit a Pull Request

If you found a solution to an open issue and implemented it, we would be happy to add your contribution in the code base. For doing so, please create a pull request. Prior to that, please make sure you

- rebase against the `develop` branch
- stick to project coding conventions
- added test cases for the problem you are solving
- added docs, describing the change
- generally comply with codeacy report

Project Setup

Project Setup

If you are interested in developing and building the project please follow the following instruction.

Version control

To get sources of the project, please execute:

```
git clone https://github.com/holunda-io/camunda-bpm-taskpool.git
cd camunda-bpm-taskpool
```

We are using gitflow in our git SCM. That means that you should start from `develop` branch, create a `feature/<name>` out of it and once it is completed create a pull request containing it. Please squash your commits before submitting and use semantic commit messages, if possible.

Project Build

Perform the following steps to get a development setup up and running.

```
./mvnw clean install
```

Integration Tests

By default, the build command will ignore the run of `failsafe` Maven plugin executing the integration tests (usual JUnit tests with class names ending with ITest). In order to run integration tests, please call from your command line:

```
./mvnw integration-test failsafe:verify -Pitest -DskipFrontend
```

Project build modes and profiles

Camunda Version

You can choose the used Camunda version by specifying the profile `camunda-ee` or `camunda-ce`. The default version is a Community Edition. Specify `-Pcamunda-ee` to switch to Camunda Enterprise edition. This will require a valid Camunda license. You can put it into a file `~/.camunda/license.txt` and it will be detected automatically.

Skip Frontend

Tip Components for production use of camunda-bpm-taskpool are backend components only. Frontend components are only created for examples and demonstration purpose.

If you are interested in backend only, specify the `-DskipFrontend` switch. This will accelerate the build significantly.

SQL scripts

The project uses Flyway for versioning of database changes. In doing so we provide the required SQL scripts for initialization of required database objects (including Camunda BPM schema, Axon schema and some example schema). If you change any of those you will need to create SQL scripts describing your change. For doing so, you can re-generate the scripts running:

```
./mvnw -Pgenerate-sql
```

Note The existing scripts must not be replaced or changed, but new additional scripts needs to added.

Documentation

We are using Orchid for generation of a static site documentation and rely on AsciiDoc as much as possible.

Tip If you want to develop your docs in 'live' mode, run `./mvnw -f docs -Pserve-docs` and access the http://localhost:8080/ from your browser.

For creation of documentation, please run:

```
./mvnw -f docs orchid:build
```

Warning This operation requires special permissions. You need to replace `GITHUB_TOKEN` by the token of the github pages repository, allowing to publish the pages.

In order to publish documentation to github pages, please run from command line

```
./mvnw -f docs -Pdeploy-docs -DgithubToken=GITHUB_TOKEN
```

Examples

Taskpool provides a series of examples demonstrating different features of the library. By default, the examples are built during the project build. If you want to skip the examples, please add the following parameter to your command line or disable the `examples` module in your IDE.

```
./mvnw clean package -DskipExamples
```

Local Start

Important If you want to run examples locally, you will need `docker` and `docker-compose` .

Pre-requirements

Before starting the example applications, make sure the required infrastructure is set up and running. Please run the following from your command line:

```
./.docker/setup.sh
```

This will create required docker volumes and network.

Start containers

In order to operate, the distributed example applications will require several containers. These are:

- Axon Server
- PostgreSQL Database
- Mongo Database (if used in projection)

Please start the required containers executing the corresponding command from `examples/scenarios/distributed-axon-server`:

```
cd ./examples/scenarios/distributed-axon-server
docker-compose up
```

Starting application (distributed scenario)

For the distributed scenario, the containers from the previous section needs to be started. To start applications, either use your IDE and create two run configurations for the classes (in this order):

- `io.holunda.camunda.taskpool.example.process.ExampleTaskpoolApplicationDistributedWithAxonServer`
- `io.holunda.camunda.taskpool.example.process.ExampleProcessApplicationDistributedWithAxonServer`

Alternatively, you can run them from the command line:

```
./mvnw spring-boot:run -f examples/scenarios/distributed-axon-server/taskpool-applicatior
./mvnw spring-boot:run -f examples/scenarios/distributed-axon-server/process-application
```

Continuous Integration

Travis CI is building all branches on commit hook. In addition, a private-hosted Jenkins CI is used to build the releases.

Release Management

Release management has been set-up for use of Sonatype Nexus (= Maven Central)

What modules get deployed to repository

Every module is enabled by default. If you want to change this, please provide the property

```
<maven.deploy.skip>true</maven.deploy.skip>
```

inside the corresponding `pom.xml`. Currently, all examples are *EXCLUDED* from publication into Maven Central.

Trigger new release

Warning	This operation requires special permissions.

We use gitflow for development (see A successful git branching model for more details). You could use gitflow with native git commands, but then you would have to change the versions in the poms manually. Therefore we use the mvn gitflow plugin, which handles this and other things nicely.

You can build a release with:

```
./mvnw gitflow:release-start
./mvnw gitflow:release-finish
```

This will update the versions in the `pom.xml`s accordingly and push the release tag to the `master` branch and update the `develop` branch for the new development version.

Trigger a deploy

Warning This operation requires special permissions.

Currently, CI allows for deployment of artifacts to Maven Central and is executed using github actions. This means, that a push to `master` branch will start the corresponding build job, and if successful the artifacts will get into `Staging Repositories` of OSS Sonatype without manual intervention.

Run deploy from local machine

Warning This operation requires special permissions.

If you still want to execute the the deployment from your local machine, you need to have GPG keys at place and to execute the following command on the `master` branch:

```
export GPG_KEYNAME="<keyname>"
export GPG_PASSPHRASE="<secret>"
./mvnw clean deploy -B -DskipTests -DskipExamples -Prelease -Dgpg.keyname=$GPG_KEYNAME -D
```

Release to public repositories

Warning This operation requires special permissions.

The deploy job will publish the artifacts to Nexus OSS staging repositories. Don’t forget to close and release the repository to enable it’s sync with Maven Central.