Developer Guide

Table of Contents

Developer Guide

Contribution

Contribution

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

- File new issues if you found a bug or missing feature
- Submit a pull request, if you have a solution for on of the issues

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

- rebased against the `develop` branch
- sticked to project coding conventions
- added test cases for the problem you are solving
- added docs, describing the change
- generally complied 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 checkout sources of the project, please execute from your command line:

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

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

The default build command above will NOT run `failsafe` Maven plugin executing the integration tests (These are JUnit tests with class names ending with ITest). In order to run integration tests, please call from your command line:

```
./mvnw -Pitest
```

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.

Examples

If you want to skip the build of examples, please specify the `-DskipExamples` switch in your command line.

Documentation

We are using JavaEden Orchid for generation of a static site documentation and rely on Markdown 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
```

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
```

Starting example application

To start applications, either use your IDE and create run configuration for the classes:

- `io.holunda.camunda.bpm.data.example.CamundaBpmDataProcessApplication`
- `io.holunda.camunda.bpm.data.example.kotlin.CamundaBpmDataKotlinExampleApplication`

Alternatively, you can run them from the command line:

```
./mvn spring-boot:run -f example/example-java
./mvn spring-boot:run -f example/example-kotlin
```

Continuous Integration

Github Actions is building all branches on commit hook. In addition, a special action is building releases from master branch.

Release Management

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

What modules get deployed to repository

Currently, the following modules are released to OSS Maven Central:

- camunda-bpm-data-parent
- camunda-bpm-data
- camunda-bpm-data-test

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.

Create feature for development

You can create a feature branch for development using:

```
./mvnw gitflow:feature-start
```

WARNING: This operation requires special permissions.

After the feature is complete, create a PR. To merge the PR into develop use the command:

```
./mvnw gitflow:feature-finish
```

Trigger a deploy

WARNING: This operation requires special permissions.

Currently, CI allows for deployment of artifacts to Maven Central and is executed via 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. The repository gets automatically closed and released on successful upload.

If you still want to execute the deployment from your local machine, you need to execute the following command on the `master` branch:

```
./mvnw clean deploy -B -DskipTests -DskipExamples -Prelease
```