openKonsequenz - How to run the module "Contact Base Data"

Table of Contents

Please be sure that you have first **Portal (Auth n Auth)** installed and configured!

Requirements

- Browser (Chrome or Firefox)
- `contact-base-data.jar` file after a successfully maven build located in `<backend project root>/target`. See `get_started.txt`.

Prerequisites

- **To see this application running you have to run Portal application too.** The reason is the authentication, which happened in the Portal login phase.

Install and configure Apache Tomcat

Tomcat is an open-source Java Servlet Container and provides a "pure Java" HTTP web server environment in which Java code can run.

- Download Tomcat version 8.x.xx from https://archive.apache.org/dist/tomcat/tomcat-8/ and extract it (apache-tomcat-8-x-xx.zip).
- Place the extracted folder on your C drive.

Install and deploy the Database

Use any software for databases which is compatible to postgreSQL, we suggest pgAdmin 3:

1. Download and install pgAdmin (version 3 is used during developing process) from: https://www.pgadmin.org/download/
2. Create a database example: `ContactBaseDataDbProd`
3. Create a Role with name `CBD_SERVICE` and a password. You can either use the script in `<backend project root>/deploy/db/01_createRole.sql` (edit the password beforehand) or do it manually with pgAdmin.
4. Execute the sql script `<backend project root>/deploy/db/02_create_tables_with_example_data.sql`

The Database is now ready to use and filled with some example data. If you want to clean the example data execute the sql script `<backend project root>/deploy/db/03_clean_example_data.sql`

How to run the Backend

Put the `contact-base-data.jar` (see Requirements) in a folder of your choice. Copy the file `application.yml` from `<backend project root>/src/main/resources/application.yml` next to your `contact-base-data.jar`. Configure your copied `application.yml` according to your desired and mandatory settings. See next paragraph for an explanation of each option.

Configuration of the Contact Base Data Backend

The backend service is configured with the `application.yml` file.

Profiles

This yml-file can be divided into different configuration profiles. When starting the backend-service one has the possibility to specify the active profile with the -D flag (more on that later). If no active profile is selected the "Default" profile will be used. The "Default" profile starts at the beginning of the file and ends at the first "---". Profiles are divided by "---" and can be recognised by the keyword "spring: profiles: <profilename>".

If a profile is missing a setting, it'll be taken from the "Default" profile.

```
[...]

---

spring:
profiles: test

[...]
```

Configuration Settings

All possible configuration values located in the in application.yml and their explanation.

- **spring**:
  - **datasource**: Section for the database connection
    - url: jdbc:postgresql://serverdomain:port (default: 5432)/NameOfDatabase (Example: ContactBaseDataDbProd)
    - username: cbd_service
    - password: <password of cbd_service> see "Install and deploy the Database" point 3
  - **flyway**:
    - **enabled**: (true or false) If enabled=true then the database migrations will be performed automatically when starting the application (this parameter should normally be set to "false")

- **ldap**:
  - **base**: The base LDAP path
  - **port**: The LDAP server port
  - **username**: Admin user of your LDAP
  - **password**: Admin password
  - **urls**: The URL of the LDAP server should be in the format ldap://myserver.example.com:10389. For SSL access, use the ldaps protocol and the appropriate port, e.g. ldaps://myserver.example.com:636

- **ldap-sync**:
  - **db-id-mapping**:
    - **telephone-number-id**: the primary ID for "telephone number" row in table REF_COMMUNICATION_TYPE. (Default: 1) If set to -1 ldapmapping is disable for "telephone number"
    - **mail-id**: the primary ID for "mail" row in table REF_COMMUNICATION_TYPE. (Default: 2) If set to -1 ldapmapping is disable for "mail"
  - **scheduling**:
    - **enabled**: (true or false) Switches LDAP synchronisation on/off
    - **cron-expression**: Cron Trigger Tutorial
      Spring Cron only takes 5 parameters not 6, the year is exlcuded!
      Examples: '*/10 * * * * *' = every 10 seconds, '0 0 */3 ? * *' = every 3 hours, '0 0 0 * * ?' = every day at midnight.
  - **attribute-mapping**:
    - [not changeable variable in contact base data modul]: [attribute field name in your LDAP to be mapped].All possible mappings are:

```
uid: uid
fullname: cn
lastname: sn
firstname: givenname
title: title
mail: mail
department: department
telephone-number: phone
```

- **authnauth-sync**:
  - **attribute-mapping**:
    - **lastname**: (true or false) Switches AuthNAuth synchronisation of field Lastname on/off
    - **firstname**: (true or false) Switches AuthNAuth synchronisation of field Firstname on/off

- **scheduling**:
  - **enabled**: (true or false) Switches AuthNAuth synchronisation on/off
  - **cron-expression**: Cron Trigger Tutorial
    Spring Cron only takes 5 parameters not 6, the year is exlcuded!
    Examples: '*/10 * * * * *' = every 10 seconds, '0 0 */3 ? * *' = every 3 hours, '0 0 0 * * ?' = every day at midnight.
- **technical-username**: A technical user from the AuthNAuth modul doesn't have to be an admin
- **technical-userpassword**: Technical user password

- **server**:
  - **port**: (Default: 9155) Port which is used for this backend server
  - **max-http-header-size**: Maximum size for the http-headers

- **jwt**:
  - **tokenHeader**: Name of the http-header which carries the authentication-token. (should be "Authorization")
  - **useStaticJwt**: If set to "true" then the backend will use **jwt.staticJwt** as Authorization-token. (This won't work for calls to other modules like the Auth'n'Auth-Modul, because the token will be out of date)

- **services**:
  - **authNAuth**:
    - **name**: authNAuthService

- **authNAuthService**:
  - **ribbon**:
    - **listOfServers**: Here one can configure the base URL to the Auth'n'Auth-service Example: http://entopkon:8880. The server where the Auth'n'Auth modul is installed.

- **feign**:
  - **client**:
    - **config**:
      - **default**:
        - **connectTimeout**: (Default: 60000) Connection timeout for the REST-Calls (in ms).
        - **readTimeout**: (Default: 60000) Read timeout for the REST-Calls (in ms).

- **cors**:
  - **corsEnabled**: (Default: false) (true or false) Cross-Origin Resource Sharing on/off

Starting the Backend

To execute the backend with the "Default" profile just run the command:

```
$ java -jar ./contact-base-data.jar qserver
```

To execute the backend with for example the profile "qserver" run the command:

```
$ java -Dspring.profiles.active=qserver -jar ./contact-base-data.jar qserver
```

Make sure you have set "corsEnabled" to true in your Production environment since Frontend (Tomcat) and Backend (Spring Application) will run on different ports!

It’s recommend to install the execution of the Backend as a System Service (Win/Linux).

How to run the Frontend

Configure your Webserver

Frontend (Tomcat) and Backend (Spring Application) will run on different ports you need to configure your webserver to proxy the request coming form the Frontend to the Backend. The following is an example configuration for an Apache HTTP Server (Webserver) for port 80.

```
<VirtualHost *:80>

        # Kontaktstammdaten
        ProxyPass /contactdatabase/api http://localhost:9155

</VirtualHost>
```

`http://localhost:9155` is the location of the Backend. The port has to be adjusted according to your settings made in see section "How to run the Backend" → "Configuration of the Contact Base Data Backend" → "Configuration Settings" → "Server:" "Port:"

Deploying the Frontend

1. Create a folder named `contactdatabase` in `<tomcat>/webapps` folder
2. After building the Frontend (see `howtoBuild` file) copy the content of the `dist` folder to `<tomcat>/webapps/contactdatabase`
3. Start your Tomcat

Now log in with the "Portal Application" and open the "Contact Base Data" module. If it shows the overview and the its example data the application was successfully deployed.