# HSE Setup and Usage Guide

Robert Jans

December 20, 2020

# Contents

# 1 (Temporary) Source Download and Running Application

The source codes and related documentation files are available on *GitHub* under

https://github.com/robix82/bsc_project/.

The repository can be cloned by issuing the following command

```
git clone https://github.com/robix82/bsc_project.git
```

For testing the application and its interactions with Qualtrics, I temporarily deployed it on

`http://www.robix-projects.org/hse`.

You can log in using the following credentials;

- username: admin
- password: admin

# 2 Employed Technologies

The project is constructed as a web application using the *SpringBoot* framework. It consists in a back-end written in java, HTMl pages (served via the *Thymeleaf* templating engine), some *(*JavaScript) to be run on client side, and *css* files for the user interface styling. Data is stored in part in a *MySql* database and in part as text files on the server's file system. The *JQuery* and *Bootstrap* frameworks are used for keeping the front-end code as simple as possible.
Dependency management and build configurations are handled by the *Maven* project management tool. In order to allow a simple deployment procedure, the application is packaged into a *Docker* image at build time. The application can be started and stopped using *Docker Compose* which automatically downloads, initializes, and links the required *MySql* database. For a detailed deployment description, see section 3. For inspecting and/or modifying the source code, I suggest using the *Eclipse* IDE.

# 3 Configuration, Build and Deployment

## 3.1 Dependencies

The system on which you build the application must have the following software installed:

- Java JDK 11
- Apache Maven 3.6.3
- Docker version 20.10.1

If you need to run the application locally without using docker, also *MySql* is required.
The server on which the application is to be deployed, only needs *Docker* and *Docker Compose*.

## 3.2 Configuration Files

### 3.2.1 Maven configuration: pom.xml

The build settings used by *Maven* are defined in `/hse/pom.xm`. This file contains some general information about the project, such as name and version, as well as a list of java packages and *Maven* plugins which are downloaded and set up at build time. The File also declares two profiles ("dev" and "prod"). The "dev" profile is intended for creating a local build to be used during development, while the "prod" profile is to be used for building the deployment version. The profiles are linked to specific configuration files

which contain various settings such as server ports and global constants: when the profile "dev" is selected, the application uses the file `/hse/src/main/resources/application-dev.properties`; when "prod" is selected, `/hse/src/main/resources/application-prod.properties`.
By default the "dev" profile is selected; for using the "prod" profile, the flag `-Pprod` is to be included in the *Maven* command (e.g. `mvn -Pprod clean install`).

### 3.2.2 Specific configurations in .properties files

The directory `/hse/src/main/resources/` contains three files with extension `.properties`: `application.properties`, `application-dev.properties` and `application-prod.properties`. The first one contains settings that are applied independently of the selected profile, while the other two contain profile-specific settings. The crucial settings to be considered at build time are the `spring.datasource.xxx` properties, which indicates the database which the application is going to connect to, and the `baseUrl` parameter, which indicates the prefix used at server level. For instance in the `application-prod.properties` as I have set it up, the data source is pointing to a *MySql* Docker container, and the base url is set to `/hse/`, since I deploy it on `http://www.robix-projects.org/hse/`. In the `application-dev.properties` file the data source points to a local *MySql* instance and the baseUrl is `/`.
The other properties indicate directory paths and should not need to be modified.

### 3.2.3 docker-compose.yml

The simplest way to run the application on a server is by using *Docker Compose*. The way in which the containers are created from the images and the internal ports used are defined in `/hse/docker-compose.yml`.

## 3.3 Creating a local build

### 3.3.1 Preparing the database

In order to run the application locally , a *MySql* database service running on port 3306 is required. The service must contain a database named `hse_db` and should be accessible via username `root` and password `root`. The tables are created automatically at application startup. If you need to use other login credentials, or the service is running on another port, these parameters can be set in `application-dev.properties`.

### 3.3.2 Issuing the build command

The command

```
mvn clean install
```

initiates the build process. The process involves executing several test suites, which should work without failure. In case the tests fail (e.g. due to path incompatibilities or missing files) the tests can be skipped using the `-DskipTests` flag:

```
mvn -DskipTests clean install
```

### 3.3.3 Running the application

Once the application is built, it can be run in several ways. During development it is convenient to run it from the IDE (in *Eclipse* package explorer, right-click on project → Run As → Spring Boot App). Alternatives are to run it from command line using *Maven*:

```
cd hse/
mvn spring-boot:run
```

or using *Java*:

```
cd hse/target/
java -jar hse-0.1.jar
```

## 3.4 Example deployment on *Ubuntu Server* with *Apache2*

### 3.4.1 Create and transfer the *Docker* image

The first step consists in creating the application's image by issuing

```
cd hse/
mvn -Pprod clean install
```

At this point, the output of `docker image ls` should contain a line similar to:

```
REPOSITORY           TAG       IMAGE ID       CREATED              SIZE
robix82/usi.ch-hse   0.1       c2137e7cc110   37 seconds ago       758MB
```

Once the image is created it can be exported as a `.tar` file by issuing

```
docker save robix82/usi.ch-hse:0.1 > hse.tar
```

Finally the `.tar` file and `/hse/docker-compose.yml` must be copied to the server, e.g. using `scp`.

### 3.4.2 Load the image and start the application

On the server, the image from the `.tar` file can be loaded with

```
docker load < hse.tar
```

With the image loaded, the application can be started by issuing

```
docker-compose up &
```

from the directory containing the `docker-compose.yml` file. The required *MySql* image will be downloaded and initialized automatically.
At this point the application is reachable on port 8081 (the port can be configured in `docker-compose.yml`).

### 3.4.3 Apache2 configuration

In order to make the application reachable on the server's external address with a custom prefix, it is necessary to configure a virtual host using Apache's `mod_proxy` module. For details on `mod_proxy`, please refer to https://www.digitalocean.com/community/tutorials/...

The virtual host configuration is done by placing a file (in this example `hse.conf`) in `/etc/apache2/sites-available/` and creating a symlink to it in `/etc/apache2/sites-enabled/`:

```
ln -s /etc/apache2/sites-available/hse.conf /etc/apache2/sites-enabled/
```

The content of the `.conf` file should look similar to

```
<VirtualHost *:80>

  ServerName www.robix-projects.org
  ProxyPreserveHost On

  ProxyPass /hse/ http://127.0.0.1:8081/
  ProxyPassReverse /hse/ http://127.0.0.1:8081/

</VirtualHost>
```

This configuration makes the application available under `http://www.robix-projects.org/hse`. Notice that the prefix `/hse/` must match the `baseUrl` property in `application-prod.properties` and the port (8081 in this example) must correspond to the port defined in `docker-compose.yml`.

# 4 Usage

## 4.1 Users, Roles and their Definition

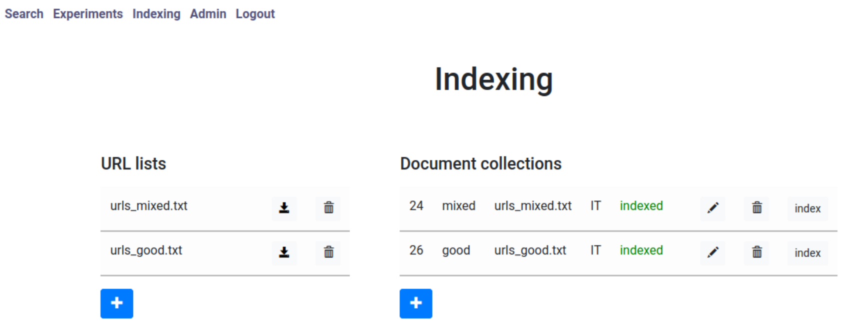
The application distinguishes three types (roles) of users: *Administrators*, *experimenters*, and *participants*. Depending on a user's role, different UI elements are visible or accessible: participants can access only the search interface; experimenters have access to the indexing and the experiment setup interfaces; administrators have access to all interfaces, including a page for creating, updating and removing administrators and experimenters. Participants can be defined by administrators or experimenters when an experiment is set up, or automatically if the experiment is run within a *Qualtrics* survey.
If the application is newly installed, a default user with *Administrator* role created. This user can log in using the user name "*admin*" and password "*admin*".

## 4.2 URL lists and Document collections

Before an experiment can set up, at least one document collection must be available. These can be created on the *indexing* UI page available to experimenters and administrators (see **Figure 1**). Creating a document collection involves the following steps:

- Upload a URL list, i.e. a text file (.txt) containing one url per line.
- Define a doc collection by setting its name, the URL list to be used, and the language (*IT* or *EN*) of the web pages.
- Start the indexing process. This may take a long time since it involves downloading all the pages over the network; while testing I observed times in the order of one second per URL.

Document collections will later be assigned to test groups, so the search engine returns different results depending on which test group a participant belongs to. Optionaly a document collection can be set as fixed result for the first query; in this case the first query submitted by a participant returns this entire document collection, in the order in which the URLs are set in the list used for generating the collection.



URL lists

| | | |
|---|---|---|
| urls_mixed.txt | ⬇ | 🗑 |
| urls_good.txt | ⬇ | 🗑 |

Document collections

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 24 | mixed | urls_mixed.txt | IT | indexed | ✎ | 🗑 | index |
| 26 | good | urls_good.txt | IT | indexed | ✎ | 🗑 | index |

User interface for creating document collections