DBSeeder Development and Operational Image

This image supports the use of a Docker container for the development and operation of DBSeeder in an Ubuntu environment.

Table of Contents

1. Installed core components

With the following command you can check in detail which software components in which versions are included in the Docker image:

```
apt list --installed
```

Version 2.9.1

Component	Version	Remark	Status
Docker Engine	20.10.7		
Eclipse	2021-03-R		
Git	2.31.1		
Gradle	7.0.2		
Java	16.0.1	openjdk	
Ubuntu	20.04.2 LTS	focal	
Vim	8.2.2949		

2. Creating a new DBSeeder container

2.1 Getting started

```
> REM Assumptions:
> REM   - you want to map the container port 8443 to the host port 443
> REM   - the name of the Docker container should be: my_db_seeder
> REM   - the path the host repository is: //C/projects/my_repro
> REM   - the directory name for this repository inside the container should be:
my_repro_dir
```

```
> REM   - you want to use the latest version of the DBSeeder image
> docker run -it -p 443:8443 \
             --name my_db_seeder \
             -v //C/projects/my_repro:/my_repro_dir \
             konnexionsgmbh/db_seeder:latest

> REM Stopping the container
> docker stop my_db_seeder

> REM Restarting the container
> docker start my_db_seeder

> REM Entering a running container
> docker exec -it my_db_seeder bash
```

2.2 Detailed Syntax

A new container can be created with the `docker run` command.

Syntax:

```
docker run -it
           [-p <port>:8443] \
           [--name <container_name>] \
           konnexionsgmbh/db_seeder[:<version>]
           [<cmd>]
```

Parameters:

- **port** - an optional listener port
- **container_name** - an optional container identification
- **directory_repository** - an optional host repository directory - the default value is expecting the repository inside the container
- **version** - an optional version number of the image or the constant `latest`
- **cmd** - an optional command to be executed in the container, default is `bash` for running the `bash` shell

Detailed documentation for the command `docker run` can be found here.

Examples:

1. Creating a new Docker container named `my_db_seeder` using a repository inside the Docker container:

   `docker run -it --name my_db_seeder konnexionsgmbh/db_seeder:latest`

2. Creating a new Docker container named `my_db_seeder` using the host repository of a Windows directory `D:\projects\my_repro`:

```
docker run -it --name db_seeder -v //D/projects/my_repro:/my_repro
konnexionsgmbh/db_seeder:latest
```

3. Creating a new Docker container named `my_db_seeder` using the host repository of a Linux directory `/my_repro` and mapping port `8443` to port `8000`:

```
docker run -it --name my_db_seeder -p 8000:8443 -v /my_repro:/my_repro
konnexionsgmbh/db_seeder:latest
```

---

3 Working with an existing DBSeeder container

3.1 Starting a stopped container

A previously stopped container can be started with the `docker start` command.

Syntax:

```
docker start <container_name>
```

Parameter:

- **container_name** - the mandatory container identification, that is an UUID long identifier, an UUID short identifier or a previously given name

Detailed documentation for the command `docker start` can be found here.

3.2 Entering a running container

A running container can be entered with the `docker exec` command.

Syntax:

```
docker exec -it <container_name> <cmd>
```

Parameter:

- **container_name** - the mandatory container identification, that is an UUID long identifier, an UUID short identifier or a previously given name
- **cmd** - the command to be executed in the container, e.g. `bash` for running the `bash` shell

Detailed documentation for the command `docker exec` can be found here.