JDBC

JDBC is used to connect your Postgres database with the application.

[Requirements 2](#_Toc89077709)

[Guide 2](#_Toc89077710)

[Add the Postgres dependency to *pom.xml* 2](#_Toc89077711)

[Set the correct DB URL & Schema 3](#_Toc89077712)

[Implement / refactor Java classes to connect to the database 3](#_Toc89077713)

[Connection to database 3](#_Toc89077714)

[SSL mode: Prefer 3](#_Toc89077715)

[Create local user 4](#_Toc89077716)

[Local user, hakkaton user 4](#_Toc89077717)

[Create schema 5](#_Toc89077718)

[Create tables 5](#_Toc89077719)

[Grant privileges 5](#_Toc89077720)

[Add Some Data 5](#_Toc89077721)

[Acceptance criteria 6](#_Toc89077722)

[Remarks on connection pooling 6](#_Toc89077723)

[DDL script (used to create schema and table) 7](#_Toc89077724)

## Requirements

* Table *animal* is created on databanken.ucll.be
* User *hakkaton* has correct rights on this table
* Use the hakkaton database

## Guide

### Add the Postgres dependency to *pom.xml*

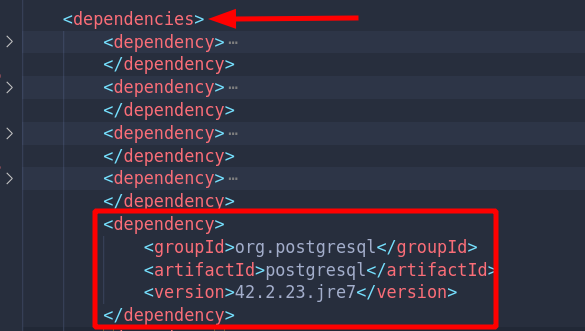
<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

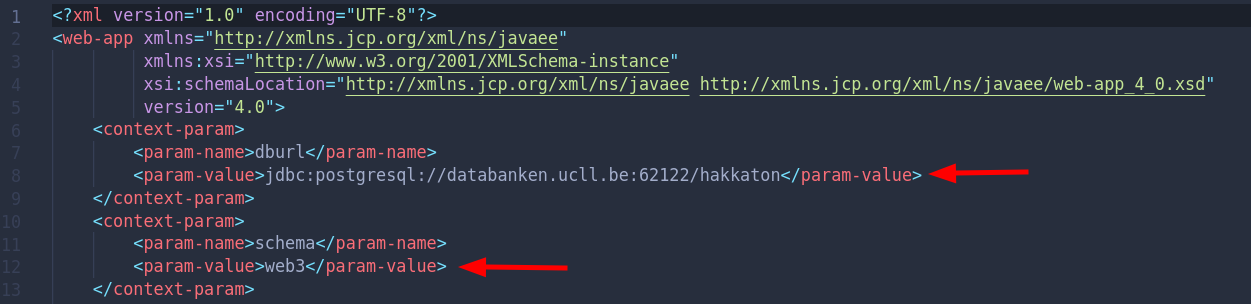
<version>42.2.23.jre7</version>

</dependency>



### Set the correct DB URL & Schema

Navigate to the *web.xml* file under *src/main/java/webapp/WEB-INF*. And set the context params *dburl* & *schema* correctly.



### Implement / refactor Java classes to connect to the database

For more information see: <https://github.com/UCLLWeb3-2122-students/database-web/tree/master/src/main/java/util>

## Connection to database

If you didn’t connect to the database yet this year, some action is needed.

Ask E. Steegmans or G. Jongen and send them an email with your first name, last name and r-number.

Create with pgadmin4 a connection with the database on port 52223.

The procedure is the same as in Web 3.

If you didn’t before, use following parameters:

**Object > Create > Server…**

**General**

Name = *<insert your servers name>.*

**Connection**

Hostname = databanken.ucll.be

Port = 52223

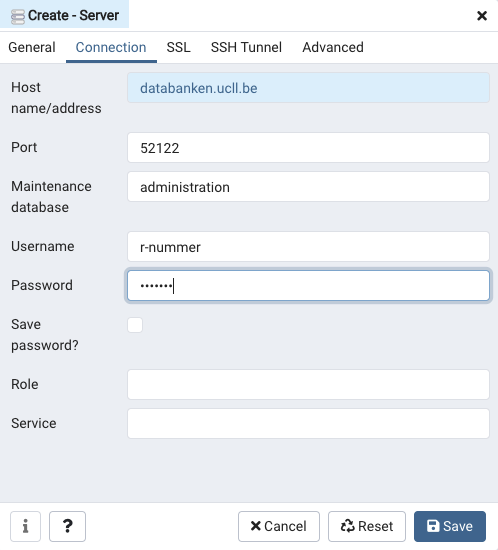
Maintenance database = administration

Username = (your r-number)

Password = (your password)

**SSL**

## SSL mode: Prefer



## 

## Create local user

More information about local user: <https://projektwerk.ucll.be/projects/4testing/wiki/Databankgebruik#local_users>

In this part we are creating a local user for your application. If you did it for web3, you don’t have to do it again.

Execute following command:

*SELECT user\_administration.set\_session\_svg('webontwerp','2TX31')*

The following query will provide you a local user and a password.

*SELECT user\_administration.new\_local\_user('webontwerp');*

If you need a new password, you just have to repeat the last step.

## Local user, hakkaton user

You should know it is not a good idea to publish user id and password on git. However, for the moment we don’t have another possibility.

For this reason, local user is introduced in web 3 (local\_r0123456). However, local user (and its password) is student related. Therefore, we use for this project week a “hakkaton” user. Ask your scrum master for your “hakkaton” user and use it for all your code you put in git.

## Create schema

During this week, we use the DB “hakkaton”.

Create a schema on the DB “hakkaton”. The schema name should start with “presence-” or “cactus-” (including “-” and depending on the chosen project), followed by your team.

Example: “presence-team1”.

## Create tables

Don’t forget to use the right [search\_path](https://www.postgresql.org/docs/9.1/static/runtime-config-client.html). (Checkout the link if you don’t understand the search\_path command.)

**set search\_path to** *<schemaNaam>***;**

Next, you can create the table 'animal’.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

All options on create table: <https://www.postgresql.org/docs/9.0/sql-createtable.html>

## Grant privileges

Give your team members (local\_user members, your hakkaton user) grant to your schema and tables:

**grant all privileges on schema** *<schemaName>* **to** *<user>***;**

**grant all privileges on ALL TABLES IN SCHEMA** *<schemaName>* **to** *<user>***;**

**grant all privileges on ALL SEQUENCES IN SCHEMA** *<schemaName>* **to** *<user>***;**

All options on grant: <https://www.postgresql.org/docs/9.0/static/sql-grant.html>

## Add Some Data

Add manually some data to your animal table.

## Acceptance criteria

· A schema for your team is setup with the right credentials in the DB “hakkaton”. The schema name should start with “presence-” or “cactus-” (including “-” and depending on the chosen project), followed by your team.

Example: “presence-team1”.

* In that schema a table animal is set up (columns id, name, type, food).
* Each team member has access to that scheme with his personal login and can edit the tables when needed.
* The hakkaton-user has access to the schema, table and sequences.
* At least 3 animals are added to your table.
* You can visit your website in the browser. The overview page shows animals in the database.

## Remarks on connection pooling

[GJ] Nota bij dit punt: in 22-23 voorziet de code van web 3 geen re-connectie meer. Als er lang geen activiteit is op de applicatie, gaat de connectie met de db altijd verloren.

In samenwerking met team “server” moet je hiervoor een oplossing zoeken.

The application is connected to the database when initiated. The connection is created in util.AppContextListener.

Port 62223 establishes a pooling connection. After some time, the connection’s place in the pool is cut down. It will be re-established automatically but has lost its history. The connection won’t be able to find its search path. Therefore, in the URL, you must mention the search path with all queries.

When the connection with the database has been idle for longer time, the database itself disconnects the application.

Therefore, you find some extra code in Servlet that checks the connection and re-opens it when necessary.

Some other classes are involved:

* DBConnectionManager.getInstance(): returns always new DBConnectionManager
* DB, DBSQL and Service: methods to get connection and to reconnect

## DDL script (used to create schema and table)

*-- Change groep102 to your group name*  
*CREATE SCHEMA* groep102;  
  
CREATE SEQUENCE groep102.animal\_id\_seq;  
  
CREATE TABLE groep102.animal  
( id integer NOT NULL DEFAULT nextval('groep102.animal\_id\_seq'::regclass),  
 name character varying COLLATE pg\_catalog."default" NOT NULL,  
 type character varying COLLATE pg\_catalog."default",  
 food integer,  
 CONSTRAINT animal\_pkey PRIMARY KEY (id, name)  
);  
  
  
INSERT INTO groep102.animal ("name","type","food") values ('Blub','vis',5);  
INSERT INTO groep102.animal ("name","type","food") values ('Mia','kat',7);