What is Flyway and how does it work?

Flyway is an open-source tool, licensed under Apache License 2.0, that helps you implement automated and version-based database migrations. It allows you to define the required update operations in an SQL script or as Java code.

You can run the migration from a command line client or automatically as part of your build process or integrated into your Java application. Flyway then detects the current version of your database and performs the necessary update operations to get the database to the latest version.

Define your first migration

You can use SQL scripts or Java classes to define your database migration. In most cases, you should define all migration steps in an SQL script like the following one. But if you need to implement very complex migrations, you can do also that in a Java class.

```
CREATE OR REPLACE FUNCTION calculate(IN x double precision, IN y double precision, OUT sum double precision) RETURNS double precision AS $BODY$ BEGIN sum = x + y; END; $BODY$ LANGUAGE plpgsql VOLATILE COST 100;

CREATE TABLE book (id bigint NOT NULL, publishingdate date, title character varying(255), price double precision, version integer, CONSTRAINT book_pkey PRIMARY KEY (id));

CREATE SEQUENCE hibernate_sequence INCREMENT 1 MINVALUE 1 MAXVALUE 9223372036854775807 START 1 CACHE 1;
```

I store these SQL statements in a file called *V1__create_database*. The file name follows Flyway's default naming convention: *V<VERSION>__DESCRIPTION.sql*.

So, this file contains the SQL statements for database version 1 and Flyway will store it with the description "create_database" in the *SCHEMA_VERSION* table.

Perform the database migration

You can integrate Flyway into your application, or you can run it automatically as part of your build process or manually from the command line.

Flyway command line client

The command line client is easy to use. You just need to download the latest version from https://flywaydb.org and extract the archive to your local file system.

In its default configuration, Flyway processes all SQL files located in the *sql* folder. So, you should copy you *V1__create_database.sql* file there.

And the conf folder contains the *flyway.conf* configuration files. The most important parameters are *flyway.url*, *flyway.user* and *flyway.password*.

```
flyway.url=jdbc:postgresql://localhost:5433/recipes
flyway.user=postgres
flyway.password=postgres
```

The *flyway.url* parameter defines the JDBC url which Flyway shall use to connect to the database. For most databases, Flyway will detect the required JDBC driver based on the *flyway.url*.

The parameters *flyway.user* and *flyway.password* are optional. The command line client will prompt you for the user information, if you don't provide them in the configuration file.

OK, so let's run Flyway and initialize the database to version 1. You can do that by calling the command line client with the *migrate* command.

```
C:\dev\wrk\Flyway\flyway-4.2.0>flyway migrate
Flyway 4.2.0 by Boxfuse

Database: jdbc:postgresql://localhost:5433/recipes (PostgreSQL 9.6)
Successfully validated 1 migration (execution time 00:00.038s)
Current version of schema "public": << Empty Schema >>
Migrating schema "public" to version 1 - create database
Successfully applied 1 migration to schema "public" (execution time 00:00.076s).
```

And you're done. As you can see in the log output, Flyway found an empty database and applied the migration script for version 1.

Integrating Flyway into your Java application

Flyway is implemented in Java and extremely easy to integrate. You just have to add the flyway-core jar file to your project. If you're using Maven, you can do that with the following dependency.

```
<dependency>
    <groupId>org.flywaydb</groupId>
    <artifactId>flyway-core</artifactId>
    <version>4.2.0</version>
</dependency>
```

And after you've done that, you can trigger the Flyway database migration from your Java code.

You can do that by creating a new *Flyway* instance, configuring the datasource and calling the *migrate()* method. You can either call the setDataSource method with a DataSource object or provide the JDBC connection, username and password as Strings.

```
Flyway flyway = new Flyway();

flyway.setDataSource("jdbc:postgresql://localhost:5433/recipes", "postgres", "postgres");

flyway.migrate();
```

OK, you're almost done. The only thing that's missing are the migration scripts. I'm reusing the SQL script from the previous example and copy it to the *src/main/resources/db/migration* folder. Similar to the command line client, Flyway's API integration checks all files in that folder and uses them to update the database if necessary.