Un poco de Java y +

Otra forma de hablar de nuestro día a día…

BASE DATOS, JAVA, MAVEN, QUÉ ES, UN POCO DE

¿Qué es Flyway?

Date: 2 enero 2018 Author: Luis Miguel Gracia 0 Comentarios

Flywayes una herramienta open-source para migrar entre base de datos.

Es una herramienta muy sencilla de usar que puede usars desde línea de comandos, desde su API Java, o desde Maven y Graddle.

Además ofrece Plugins para herramientas como Spring Boot, Grails, Paly, SBT, Grunt,…

Soporta un gran número de bases de datos: Oracle, SQL Server (including Amazon RDS and Azure SQL Database, DB2, MySQL (including Amazon RDS, Azure Database & Google Cloud SQL), MariaDB, PostgreSQL (including Amazon RDS, Azure Database, Google Cloud SQL & Heroku),Redshift, CockroachDB, SAP HANA, Sybase ASE, H2, HSQLDB, Derby and SQLite.

Sus comandos son: Migrate, Clean, Info, Validate, Undo, Baseline and Repair.

Veamos cómo se usa desde **línea de comandos:**

Una vez descargada y descomprimida la herramienta (descargar) configuraré el fichero **%FLYWAY%/conf/flyway.conf** por ejemplo:

```
flyway.url=jdbc:h2:file:./foobardb
flyway.user=SA
flyway.password=
```

En el directorio **%FLYWAY%/sql/** crearé un SQL (V1__Create_person_table.sql) (prefijo V1__)

```
create table PERSON (
    ID int not null,
    NAME varchar(100) not null
);
```

Y desde **%FLYWAY%/** ejecutaré **>flyway migrate**

```
flyway-5.0.3> flyway migrate
```

Si todo ha ido bien veré:

```
Database: jdbc:h2:file:./foobardb (H2 1.4)
Successfully validated 1 migration (execution time 00:00.008s)
Creating Schema History table: "PUBLIC"."flyway_schema_history"
Current version of schema "PUBLIC": << Empty Schema >>
Migrating schema "PUBLIC" to version 1 - Create person table
Successfully applied 1 migration to schema "PUBLIC" (execution time 00:00.033s)
```

Flyway crea una tabla **flyway_schema_history** en la que va almacenando el historial de migraciones, por tanto sabe los scripts que he ejecutado (aquí se explica su funcionamiento: https://flywaydb.org/getstarted/how).

Tras esto podría hacer una segunda migración, en este caso haré los INSERTs:, al directorio **/sql** le añado el fichero `V2__Add_people.sql` : (fijaos en el prefjjo V2__)

```
insert into PERSON (ID, NAME) values (1, 'Axel');
insert into PERSON (ID, NAME) values (2, 'Mr. Foo');
insert into PERSON (ID, NAME) values (3, 'Ms. Bar');
```

Y ejecuto:

```
flyway-5.0.3> flyway migrate
```

Flyway detecta que el segundo script no se ha lanzado y lo ejecuta:

```
Database: jdbc:h2:file:./foobardb (H2 1.4)
Successfully validated 2 migrations (execution time 00:00.018s)
Current version of schema "PUBLIC": 1
Migrating schema "PUBLIC" to version 2 - Add people
Successfully applied 1 migration to schema "PUBLIC" (execution time 00:00.016s)
```

También puedo ejecutar **flyway info** que me da info sobre el estado de la migración:

```
flyway-5.0.3> flyway info
```

This should give you the following status:

```
Database: jdbc:h2:file:./foobardb (H2 1.4)

+-----------+---------+---------------------+------+---------------------+---------+----------+
| Category  | Version | Description         | Type | Installed On        | State   | Undoable |
+-----------+---------+---------------------+------+---------------------+---------+----------+
| Versioned | 1       | Create person table | SQL  | 2017-12-21 18:05:10 | Success | No       |
| Versioned | 2       | Add people          | SQL  | 2017-12-21 18:05:10 | Success | No       |
+-----------+---------+---------------------+------+---------------------+---------+----------+
```

Con Flyway también puedo hacer **UNDO de migraciones,** para esto crearé un fichero con el prefijo **U (+ el número de la versión a la que hacer UNDO)__**

En el ejemplo previo:

```
U2__Add_people.sql: DELETE FROM PERSON;
```

Y

```
U1__Create_person_table.sql: DROPTABLEPERSON;
```

Para lanzar el UNDO >**flyway undo**

que como resultado da:

```
Database: jdbc:h2:file:./foobardb (H2 1.4)
Current version of schema "PUBLIC": 2
Undoing migration of schema "PUBLIC" to version 2 - Add people
Successfully undid 1 migration to schema "PUBLIC" (execution time 00:00.030s)
```

Y con el info:

```
flyway-5.0.3> flyway info

Database: jdbc:h2:file:./foobardb (H2 1.4)

+-----------+---------+---------------------+----------+---------------------+---------+----------+
| Category  | Version | Description         | Type     | Installed On        | State   | Undoable |
+-----------+---------+---------------------+----------+---------------------+---------+----------+
| Versioned | 1       | Create person table | SQL      | 2017-12-17 19:57:28 | Success | Yes      |
| Versioned | 2       | Add people          | SQL      | 2017-12-17 20:01:13 | Undone  |          |
| Undo      | 2       | Add people          | UNDO_SQL | 2017-12-17 22:45:56 | Success |          |
| Versioned | 2       | Add people          | SQL      |                     | Pending | Yes      |
+-----------+---------+---------------------+----------+---------------------+---------+----------+
```

Flyway también puede usarse desde Maven:

Primero configuraré el plugin de Maven:

```
...
<build>
    <plugins>
        <plugin>
            <groupId>org.flywaydb</groupId>
            <artifactId>flyway-maven-plugin</artifactId>
            <version>5.0.3</version>
            <configuration>
                <url>jdbc:h2:file:./target/foobar</url>
                <user>sa</user>
            </configuration>
            <dependencies>
                <dependency>
                    <groupId>com.h2database</groupId>
                    <artifactId>h2</artifactId>
                    <version>1.4.191</version>
                </dependency>
            </dependencies>
        </plugin>
    </plugins>
</build>
```

En este caso crearé un directorio `src/main/resources/db/migration` donde iré poniendo mis scripts (src/main/resources/db/migration/V1__Create_person_table.sql)

Lo ejecutaré con:

```
bar> mvn flyway:migrate
```

Flyway tiene 3 versiones, una Community con licencia Apache2 y 2 comerciales:

	Community Edition	Pro Edition	Enterprise Edition
SQL-based migrations	✔	✔	✔
Java-based migrations	✔	✔	✔
Repeatable migrations	✔	✔	✔
Placeholder replacement	✔	✔	✔
Callbacks	✔	✔	✔
Custom migration resolvers/executors	✔	✔	✔
Safe for multiple nodes in parallel	✔	✔	✔
Native SQL dialect support (PL/SQL, SQLPL, T-SQL, ...)	✔	✔	✔
Latest database versions compatibility	✔	✔	✔
Java 8 / 9 compatibility	✔	✔	✔
Oracle SQL*Plus compatibility		✔	✔
Custom error handlers		✔	✔
Dry runs		✔	✔
Undo		✔	✔
Display query results		✔	✔
Older database versions compatibility			✔
Java 6 / 7 compatibility			✔
License	Apache v2	Commercial	Commercial
Maven Repository	✔	✔	✔
Source Code included	✔	✔	✔
Support	Community	Email (best effort) English	Email (2 business days) English, German, French, Dutch, Spanish