Database migrations with Flyway

Introduction

- Flyway updates a database from one version to a next using migrations. We can write migrations either in SQL with database-specific syntax or in Java for advanced database transformations.
- Migrations can either be versioned or repeatable. The former has a unique version and is applied exactly once. The latter does not have a version. Instead, they are (re-)applied every time their checksum changes.
- Within a single migration run, repeatable migrations are always applied last, after pending versioned migrations have been executed. Repeatable migrations are applied in order of their description. For a single migration, all statements are run within a single database transaction.

Flyway Maven Plugin

To install a Flyway Maven plugin, let's add the following plugin definition to our pom.xml:

```
<plugin>
    <groupId>org.flywaydb</groupId>
    <artifactId>flyway-maven-plugin</artifactId>
    <version>4.0.3</version>
...
</plugin>
```

Latest version of the plugin available at Maven Central.

Plugin Configuration

Plugin may be configured directly via the <configuration> tag in the plugin definition of our pom.xml:

```
<plugin>
 ...
    <configuration>
        <user>databaseUser</user>
        <password>databasePassword</password>
        <schemas>
            <schema>schemaName</schema>
        </schemas>
...
    </configuration>
</plugin>
```

Maven Properties

We may also configure the plugin by specifying configurable properties as Maven properties in pom.xml:

```
<project>
...
  <properties>
    <flyway.user>databaseUser</flyway.user>
    <flyway.password>databasePassword</flyway.password>
    <flyway.schemas>schemaName</flyway.schemas>
    ...
  </properties>
...
</project>
```

Maven Properties

We may also configure the plugin by specifying configurable properties as Maven properties in pom.xml:

```
<project>
...
  <properties>
    <flyway.user>databaseUser</flyway.user>
    <flyway.password>databasePassword</flyway.password>
    <flyway.schemas>schemaName</flyway.schemas>
    ...
  </properties>
...
</project>
```

External Configuration File

We may also provide plugin configuration in a separate .properties file:

```
flyway.user=databaseUser
flyway.password=databasePassword
flyway.schemas=schemaName
...
```

The default configuration file name is flyway.properties and it should reside in the same directory as the pom.xml file. Encoding is specified by flyway.encoding (Default is UTF-8).

Example Migration

First, let's add database as a dependency:

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <version>1.4.196</version>
</dependency>
```

Next, we create myFlywayConfig.properties in $PROJECT_ROOT with the following content:

```
flyway.user=databaseUser
flyway.password=databasePassword
flyway.schemas=app-dbflyway.url=jdbc:h2:mem:DATABASE
flyway.locations=filesystem:db/migrationc
```

Flyway adheres to the following naming convention for migration scripts: **<Prefix><Version>__<Description>.sql**
Where:
<Prefix> – Default prefix is V, which may be configured in the above configuration file using the flyway.sqlMigrationPrefix property.
<Version> – Migration version number. Major and minor versions may be separated by an underscore. The migration version should always start with 1.
<Description> – Textual description of the migration. The description needs to be separated from the version numbers with a double underscore.

Example: **V1_1_0__my_first_migration.sql**

Example Migration

Create a directory db/migration in $PROJECT_ROOT with a migration script named V1_0__create_employee_schema.sql containing SQL instructions to create the employee table:

```
CREATE TABLE IF NOT EXISTS `employee` (
    `id` int NOT NULL PRIMARY KEY,
    `name` varchar(20),
    `email` varchar(50),
    `date_of_birth` timestamp
);
```

Next, we invoke the following Maven command from $PROJECT_ROOT to execute database migrations:

```
$ mvn clean flyway:migrate -Dflyway.configFile=myFlywayConfig.properties
```