Migration to Flyway

Agenda

- Why are we here?
- Pain and loathing in db migrations
- Typical functionality
- Flyway capabilities
- Live demo
- How flyway can make us happy?
- Q&A

Purpose (Why?)

- Understand how to make db upgrade logic more maintainable. + simple & similar approach in all components.

Flyway vs Liquibase

Plain SQL vs Hibernate (analogy)

- SQL & Java
- Rollback not supported

- Liquibase XML
- Rollback supported

http://techieindescribable.blogspot.co.uk/2013/08/comparison-between-flyway-and-liquibase.html

What can Flyway offer

- Controls migrations with simple sql or Java executing sql
- Supports command line, maven plugin with configuration, gradle plugin with configuration, Configuration/Java API (+Spring configuration)
- Commands: migrate, clean, info, validate, baseline, repair

Metadata table

- By default SCHEMA_VERSION.
- Stores metadata about database migrations

Migration States

- Success
- Failed (only if db does not support DDL transactions) – use repair to repair.

*Some others

Migrations

- Versioned – executed only once and have a version
- Repeatable – executed each time their checksum changes

Usages:

- (Re-)creating views/functions/etc.

-Bulk reference data reinserts

Migrations

- Written in SQL with placeholder injections (Checksum validation is done)
- Written in Java implementing JDBCMigration or SpringJDBCMigration (Handle Checksum validation by yourself) – use only SQL is not enough

Can we mix?

Answer: Yes. The Order is determined by version for versioned migrations for repeatable no guaranties

Callbacks

- SQL Callbacks with placeholder injections
- Written in Java, implementing FlywayCallback interface. Use only if SQL is not enough

Can we mix?

Answer: Yes, but note: Order is not guaranteed for same lifecycle callback.

Lifecycle Callbacks

beforeMigrate	Before Migrate runs
beforeEachMigrate	Before every single migration during Migrate
afterEachMigrate	After every single migration during Migrate
afterMigrate	After Migrate runs
beforeClean	Before Clean runs
afterClean	After Clean runs
beforeInfo	Before Info runs
afterInfo	After Info runs
beforeValidate	Before Validate runs
afterValidate	After Validate runs
beforeBaseline	Before Baseline runs
afterBaseline	After Baseline runs
beforeRepair	Before Repair runs
afterRepair	After Repair runs

Resolvers

- You can write a custom resolver for resolving specific migrations. In the tutorial it was mentioned for example you want to handle not only Java and SQL but for example CSV files also ☺

Logging

- Maven plugin uses maven logging configuration
- Java code uses apache commons that can integrate Log4j, Simple logger, JDKLogger, Slf4J, etc.

Testing

- Spring test extensions with support for flyway. With support for DBUnit.

Rollback

Flyway does not support rollback. There are reasons for that:

1) As soon as you have destructive changes delete/drop/truncate – no way flyway know how to restore data.

2) Downgrade scripts assume the whole migration failed – but in practice some statements may fail, some not.

Solution from Flyway team: Test your solution and use snapshot technology of underlying storage

Live demo

How can flyway make us happy and cover all needed functionality?

Multiple databases

- Separate into different maven projects
- Use maven plugin configuration to have one project

Support for patches

- Flyway do migration automatically.
- One possible variant use custom locations and add new location when needed

Script execution order

- Out of the box just use different versions in file names to ensure the order is correct.
- In teams development parameters can consider to use "outOfOrder=true" in order to execute previous versions that appear and use "locations" to enable scripts to be executed when needed. Use gaps in the versions one team uses 1.20.x and second 1.21.x

Support for different envs: UAT/DEV,etc

- Use Java Migration
- SQL placeholders and use different properties for different envs
- Use custom "locations" parameter in flyway which will depend on parameter $environment = UAT|DEV|PRE-PROD|PROD
- Different Maven profiles for different configs

But in general, it is not good to have different db schemas for envs…

Post script checks that everything is ok (compare to golden source)

- SQL Callbacks
- Java Callbacks if SQL callback is not enough

Loading entities

- Java executors move to be java callbacks in Flyway migration

Groovy migration scripts

- Flyway uses binary classes at runtime. So it does not matter if it is Java or Groovy.

Shell scripting, other scripts

- In flyway no support. There is a fork of flyway that introduces custom resolver to enable shell scripting as migration scripts…

Do we really need it?

How to start migrations…

General approach:

- Snapshot of current db as a baseline and start using flyway from baseline.

Or to modify current scripts to be in a flyway model one by one : for SQL – filenames should correspond to flyway model, directories structure changes, Java/Groovy files migrate to Java/Groovy migrations supported by Flyway.

Flyway references

- https://github.com/flyway/flyway/blob/master/flyway-sample/src/main/java/org/flywaydb/sample/Main.java
- http://openwritings.net/content/public/excerpt/how-use-flywaydb-programmatically-use-flywaycallback
- https://flywaydb.org/documentation/maven/migrate
- https://flywaydb.org/documentation/callbacks

Flyway references

- https://cyntech.wordpress.com/2009/01/09/how-to-use-commons-logging/ https://maven.apache.org/maven-logging.html
- https://github.com/flyway/flyway/blob/master/flyway-maven-plugin/src/main/java/org/flywaydb/maven/MavenLog.java
- https://github.com/flyway/flyway-test-extensions/wiki/Usage-flyway-spring-test

Q&A