



# Introdução a Web Services REST com Spring Framework

*Parte 6 → Versionamento do Banco de  
Dados com Migrations (Flyway)*

Prof. Me. Jorge Luís Gregório

[www.jlgregorio.com.br](http://www.jlgregorio.com.br)



# Agenda

- O que é versionamento do banco de dados?
- Conhecendo a biblioteca Flyway;
- **Prática 01**
  - Criando um projeto Spring REST;
  - Configurando as dependências no arquivo *pom.xml*;
  - Criando o banco de dados;
  - Conectando a aplicação ao banco de dados;
  - Configurando o Flyway;
  - Criando as *migrations*;
  - Gerenciando *migrations*;

- **Prática 02**

- Alterando a tabela *author*;
- Alterando a tabela *book*;
- Atualizando tabela *author*;
- Atualizando tabela *book*;

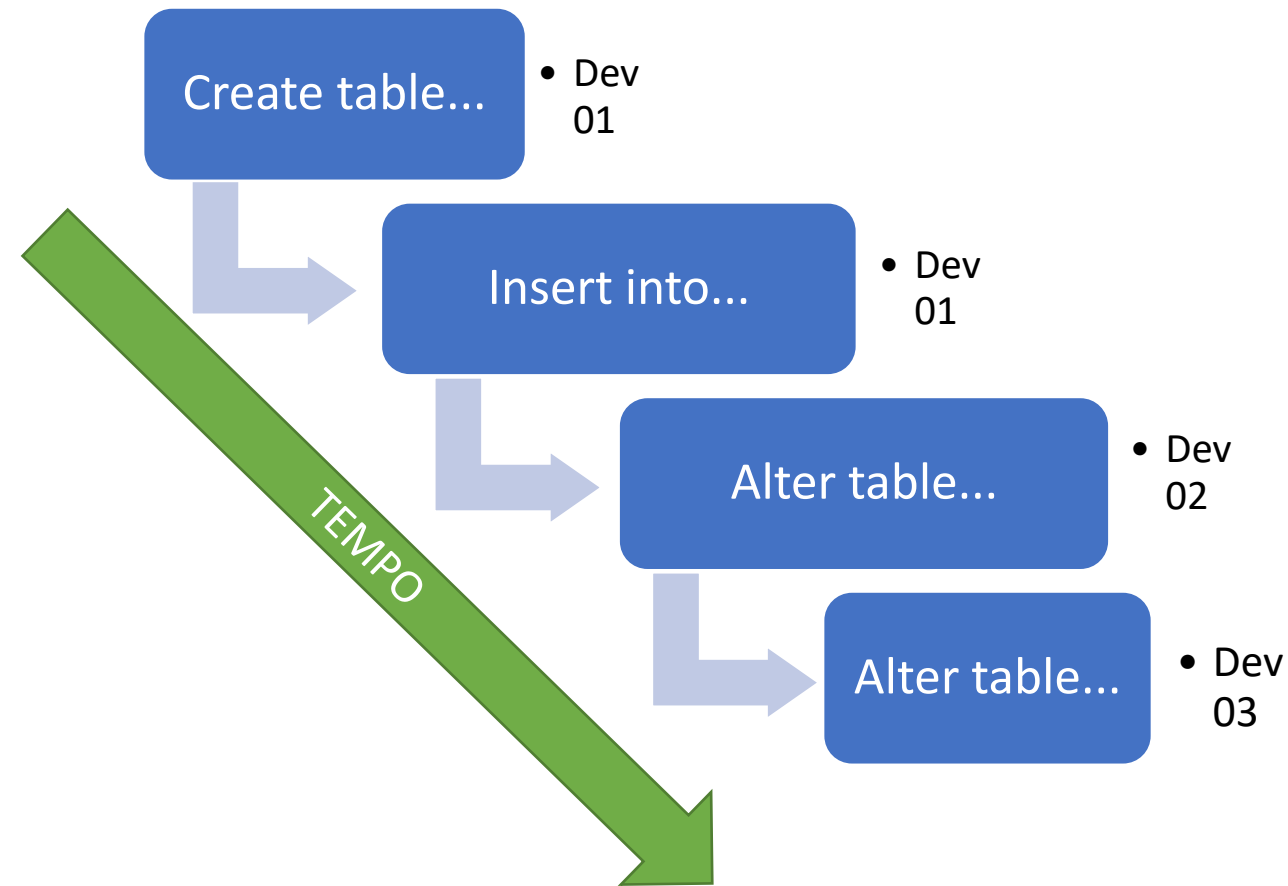
A large, stylized green leaf graphic is positioned on the left side of the slide, partially overlapping the text box. It has a light green fill and a darker green outline, with several curved segments radiating from a central point on the left.

**O que é versionamento  
de banco de dados?**

# Versionamento de banco de dados

- Em um projeto de sistemas, o banco de dados apresenta uma evolução incremental;
- Isso significa que, a medida que novos requisitos funcionais são adicionados, ou os requisitos existentes passam por modificações, é muito comum o modelo de dados sofrer modificações;
- Tabelas, campos, chaves primárias, índices, *views*, *stored procedures* e outros elementos do Banco de Dados são constantemente alterados;
- Gerenciar o *script* DDL (*Data Definition Language*) de maneira incremental manualmente não é tarefa fácil, principalmente em um cenário em que há muitos profissionais trabalhando no mesmo projeto;
- Para ajudar os desenvolvedores, os *frameworks* oferecem o recurso de **migrations**, que é uma maneira de versionar de maneira incremental o Banco de Dados;

# Cenário de múltiplos desenvolvedores



- Cada desenvolvedor possui o seu script;
- O que acontece na hora de fazer o *push* para o repositório?
  - Múltiplos scripts DDL;
- Faz sentido fazer *merge* em scripts DDL?
  - Não, pois o desenvolvimento do banco é incremental e envolve vários profissionais;
  - O ideal é ter um controle incremental de versões;

A large, stylized green leaf graphic is positioned on the left side of the slide, partially overlapping the text box. It has a smooth, curved shape with a pointed tip and a small circular detail at the base.

# **Conhecendo a biblioteca Flyway**



# Conhecendo o Flyway

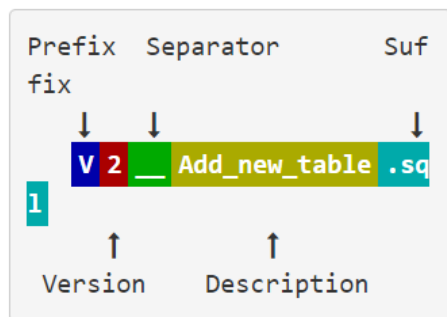
- Site oficial: <https://flyway.db.org>
- Possui suporte a diversos SGBDs: MySQL, PostgreSQL, MariaDB, DB2, Oracle, SQL Server, Firebird, etc.

## As migrations no Flyway são de 3 tipos:

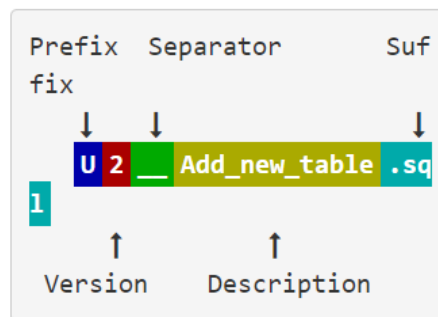
- **Versioned migrations:**
  - possuem uma versão, uma descrição e um *checksum*.
  - A versão deve ser única;
  - A descrição é informativa, para que o desenvolvedor lembre o que ela faz;
  - O *checksum* é usado para detectar mudanças acidentais;
  - Esse é o tipo mais comum de *migration*, pois é usada para criar a estrutura inicial do banco de dados;
  - Deve ser executada na ordem correta apenas uma vez.
- **Undo Migrations:**
  - Usada para desfazer alterações realizadas em uma *versioned migration*;
  - Deve possuir a mesma versão da *versioned migration* ao qual ela se refere;
  - É executada uma única vez na ordem correta;
  - Seu uso comum é associado a comandos como *alter table*, *drop table*, *delete*, etc...
- **Repeatable Migrations:**
  - Possuem uma descrição e um *checksum*, mas não possuem versão;
  - Podem ser executadas diversas vezes, sempre após as *versioned* e *undo migrations*;
  - Usadas para recriar views, procedures e reinserir grandes quantidades de dados;

# Nomeando migrations

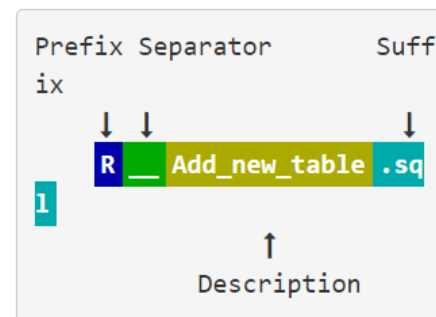
Versioned Migrations



Undo Migrations



Repeatable Migrations

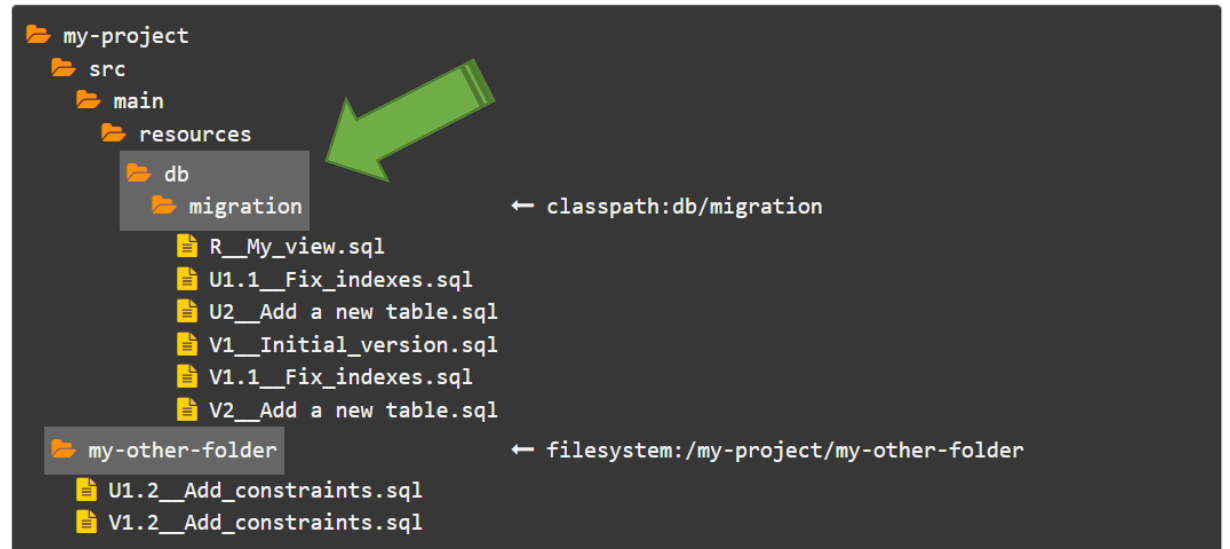


- **Prefix:** V – versioned; U – Undo; R – Repeatable;
- **Version:** valor incremental que define a versão e a ordem de execução das *migrations*. Cada evolução incremental do banco de dados deve possuir uma versão.
- **Separator:** dois underscores (\_\_)
- **Description:** a descrição da migration, separada por underscores;
- **Sufix:** .sql → o tipo de arquivo



# Onde ficam os arquivos de migration?

- Normalmente é criado dentro da pasta ***src/main/resources*** uma pasta ***db/migration***, como a figura ao lado;
- Dentro dessa pasta deve ficar os arquivos de *migrations*, seguindo as especificações da documentação da biblioteca Flyway.

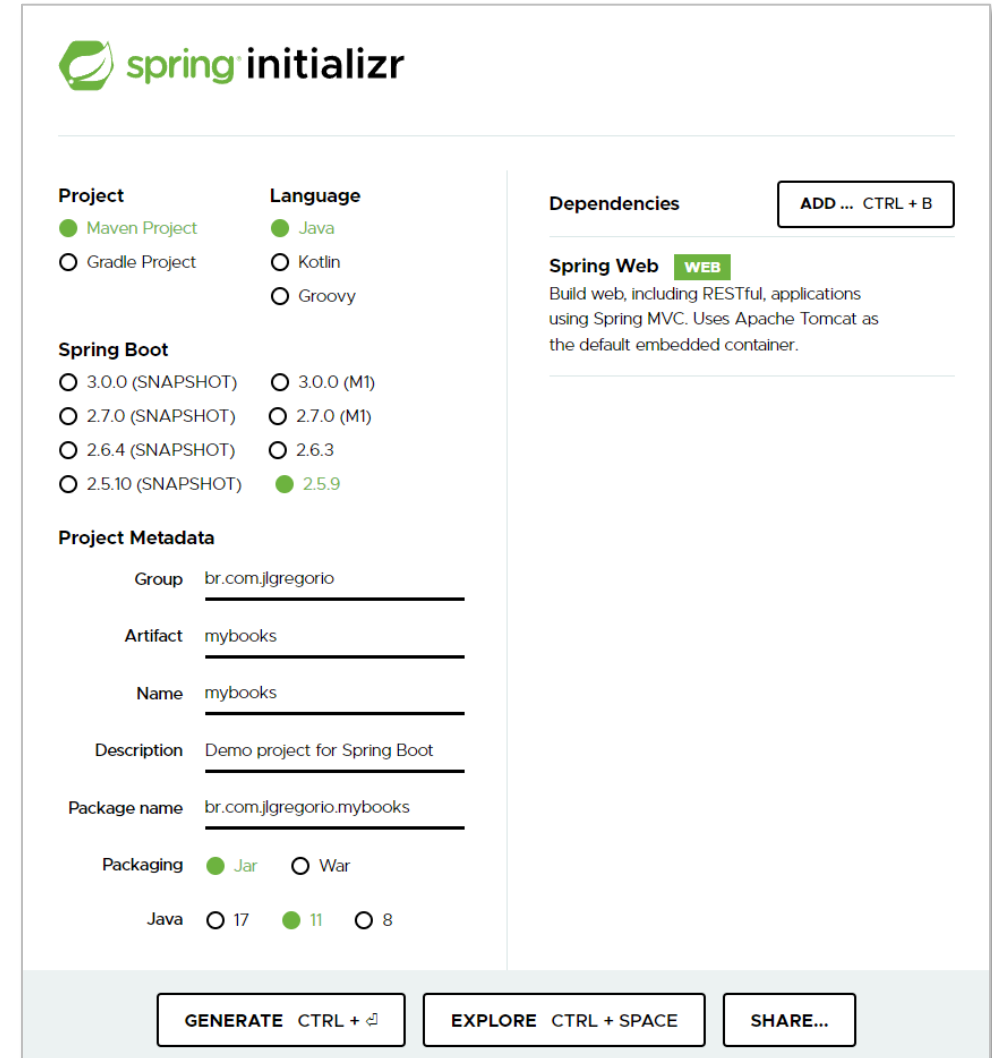


A large, stylized green leaf graphic is positioned on the left side of the slide, partially overlapping the title box. It has a light green fill and a darker green outline, with several curved segments.

# Prática 01

# Criando o projeto Spring REST

- Usando o Spring Initializr com as configurações ao lado, crie um projeto chamado MyBooks



The screenshot shows the Spring Initializr web interface. The 'Project' section has 'Maven Project' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '2.5.9' selected. The 'Project Metadata' section includes fields for Group (br.com.jlgregorio), Artifact (mybooks), Name (mybooks), Description (Demo project for Spring Boot), and Package name (br.com.jlgregorio.mybooks). The 'Packaging' section has 'Jar' selected. The 'Dependencies' section has 'Spring Web' selected. At the bottom, there are buttons for 'GENERATE', 'EXPLORE', and 'SHARE...'.

**spring initializr**

**Project**

☒ Maven Project  
☐ Gradle Project

**Language**

☒ Java  
☐ Kotlin  
☐ Groovy

**Spring Boot**

☐ 3.0.0 (SNAPSHOT) ☐ 3.0.0 (M1)  
☐ 2.7.0 (SNAPSHOT) ☐ 2.7.0 (M1)  
☐ 2.6.4 (SNAPSHOT) ☐ 2.6.3  
☐ 2.5.10 (SNAPSHOT) ☒ 2.5.9

**Project Metadata**

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 17 ☒ 11 ☐ 8

**Dependencies** ADD ... CTRL + B

**Spring Web** WEB  
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

**GENERATE** CTRL + G **EXPLORE** CTRL + SPACE **SHARE...**

# Configurando as dependências no arquivo *pom.xml*

- O driver de conexão com o banco de dados MySQL;
- A biblioteca de persistência JAVAX;
- A biblioteca Spring Data
- A biblioteca Flyway

```
<dependency>  
  <groupId>mysql</groupId>  
  <artifactId>mysql-connector-java</artifactId>  
  <version>8.0.26</version>  
</dependency>
```

```
<dependency>  
  <groupId>org.apache.aries.jpa.javax.persistence</groupId>  
  <artifactId>javax.persistence_2.1</artifactId>  
  <version>2.7.3</version>  
</dependency>
```

```
<dependency>  
  <groupId>org.springframework.data</groupId>  
  <artifactId>spring-data-jpa</artifactId>  
</dependency>  
<dependency>
```

```
  <groupId>org.flywaydb</groupId>  
  <artifactId>flyway-maven-plugin</artifactId>  
  <version>8.4.3</version>  
</dependency>
```

# Criando o banco de dados MySQL

- Inicialize o serviço do MySQL;
- Abra o terminal e digite:
  - `mysql -u root -p` → após pressionar ENTER, será pedida a senha;
  - Agora você está conectado ao MySQL;
- Crie o banco de dados **mybooks** digitando o comando:
  - `create database mybooks;`
- Use o seguinte comando para listar os *schemas* criados:
  - `show databases;`

# Conectando a aplicação ao banco de dados

- Abra o arquivo *src/main/resources/application.properties* e adicione as configurações do banco de dados:

```
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/mybooks?useTimeZone=true&serverTimeZone=UTC
spring.datasource.username=root
spring.datasource.password=mysqldba
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=false
```

# Configurando o Flyway

- No arquivo pom.xml, adicione as seguintes linhas na *tag **properties***:

```
<!--flyway -->  
<flyway.url>jdbc:mysql://localhost:3306/mybooks</flyway.url>  
<flyway.user>root</flyway.user>  
<flyway.password>mysqldba</flyway.password>  
<flyway.schemas>mybooks</flyway.schemas>
```

# Criando as migrations

- Abra a pasta **src/main/resources** e crie a pasta/subpasta **db/migration**
- Vamos imaginar um simples banco de dados com duas tabelas:
  - author: autores de livros;
  - book: títulos de livros
- Isso é um típico **relacionamento 1:N** em que um autor possui muitos livros;
- Dentro da pasta db/migration crie os arquivos
  - **V1.0\_\_create\_author\_table.sql**
  - **V1.1\_\_create\_book\_table.sql**
- Os códigos desses arquivos são típicos *scripts* DDL para a criação de tabelas. Veja ao lado:

```
/* V1.0__create_author_table.sql */
```

```
create table if not exists author (  
    id integer not null auto_increment,  
    name varchar(50) not null,  
    gender char(1) not null,  
    primary key (id)  
);
```

```
/* V1.1__create_book_table.sql */
```

```
create table if not exists book (  
    id integer not null auto_increment,  
    title varchar(50) not null,  
    author_id integer not null,  
    primary key (id),  
    foreign key (author_id) references  
    author(id)  
);
```



# Migrations para inserção de dados

- Agora vamos adicionar mais duas *migrations* para inserção de alguns dados nas tabelas criadas:
  - `V1.2__populate_author_table.sql`
  - `V1.3__populate_book_table.sql`
- O código SQL para inserção dos dados está ao lado:

```
/* V1.2__populate_author_table.sql */
```

```
insert into author (name, gender)
values
  ('George Orwell', 'M'),
  ('Arthur C. Clarke', 'M'),
  ('Andy Weir', 'M'),
  ('Ernest Cline', 'M'),
  ('Ann Leckie', 'F'),
  ('J. K. Rowling', 'F')
```

```
/* V1.3__populate_book_table.sql */
```

```
insert into book (title, author_id)
values
  ('A revolução dos bichos', 1),
  ('1984', 1),
  ('O fim da infância', 2),
  ('Encontro com Rama', 2),
  ('2001: uma odisseia no espaço', 2),
  ('Devoradores de Estrelas', 3),
  ('Perdido em Marte', 3),
  ('Jogador Número 1', 4),
  ('Armada', 4),
  ('Justiça Ancilar', 5)
```

# Executando migrations

- Uma vez que as migrations estejam definidas é necessário executar comandos a fim de criar as tabelas e inserir os dados;
- Para isso, abra o terminal na pasta raiz do projeto e digite o seguinte comando Maven:
  - *mvn flyway:migrate*



```
[INFO] Current version of schema 'mybooks': << Empty Schema >>
[INFO] Migrating schema 'mybooks' to version "1.0 - create author table"
[INFO] 0 rows affected
[INFO] Migrating schema 'mybooks' to version "1.1 - populate author table"
[INFO] 6 rows affected
[INFO] Migrating schema 'mybooks' to version "1.2 - create book table"
[INFO] 0 rows affected
[INFO] Migrating schema 'mybooks' to version "1.3 - populate book table"
[INFO] 10 rows affected
```

# Verificando o banco de dados

- Vamos voltar o gerenciador do MySQL e digitar o seguinte comando para visualizar as tabelas que foram criadas:
  - `show tables;`
- Observe que foi criada uma tabela a mais, identificada como ***flyway\_schema\_history***;
- Essa tabela é usada pelo Flyway para controlar a execução das *migrations*;
- Para verificar se os dados foram inseridos, faça um `select * from ...` nas tabelas.

```
MariaDB [mybooks]> show tables;
+-----+
| Tables_in_mybooks |
+-----+
| author             |
| book               |
| flyway_schema_history |
+-----+
3 rows in set (0.001 sec)
```

```
MariaDB [mybooks]> select * from author;
+----+-----+-----+
| id | name          | gender |
+----+-----+-----+
| 1  | George Orwell | M      |
| 2  | Arthur C. Clarke | M      |
| 3  | Andy Weir     | M      |
| 4  | Ernest Cline  | M      |
| 5  | Ann Leckie    | F      |
| 6  | J. K. Rowling  | F      |
+----+-----+-----+
6 rows in set (0.001 sec)
```

```
MariaDB [mybooks]> select * from book;
+----+-----+-----+
| id | title                                | author_id |
+----+-----+-----+
| 1  | A revolução dos bichos              | 1         |
| 2  | 1984                                | 1         |
| 3  | O fim da infância                  | 2         |
| 4  | Encontro com Rama                   | 2         |
| 5  | 2001: uma odisseia no espaço       | 2         |
| 6  | Devoradores de Estrelas            | 3         |
| 7  | Perdido em Marte                    | 3         |
| 8  | Jogador Número 1                   | 4         |
| 9  | Armada                              | 4         |
| 10 | Justiça Ancilar                     | 5         |
+----+-----+-----+
10 rows in set (0.000 sec)
```

# Verificando a tabela *flyway\_schema\_history*

installed_rank	version	description	type	script	checksum	installed_by	installed_on	execution_time	success
1	1.0	create author table	SQL	V1.0__create_author_table.sql	694460888	root	2022-02-01 20:09:32	9	1
2	1.1	populate author table	SQL	V1.1__populate_author_table.sql	785440364	root	2022-02-01 20:09:32	5	1
3	1.2	create book table	SQL	V1.2__create_book_table.sql	-1104538355	root	2022-02-01 20:09:32	9	1
4	1.3	populate book table	SQL	V1.3__populate_book_table.sql	767835704	root	2022-02-01 20:09:32	7	1

- A tabela *flyway\_schema\_history* mostra quais migrations foram executadas, a origem, o tempo de execução, a versão e outras informações úteis.

# Outros comandos Flyway

- `mvn flyway:migrate` → executa as migrations;
- `mvn flyway:clean` → zera o banco de dados, ou seja, destrói todas as tabelas, inclusive a tabela `flyway_schema_history`;
- `mvn flyway:info` → mostra informações da tabela `flyway_schema_history`;
- `mvn flyway:undo` → desfaz a última migration executada;
  - ATENÇÃO! O comando `mvn flyway:undo` não é suportado pela versão gratuita do Flyway. Para mais informações, consulte a documentação: <https://flywaydb.org/documentation/command/undo>
- `mvn flyway:validate` → valida as migrations (mais informações: <https://flywaydb.org/documentation/usage/commandline/validate>);
- `mvn flyway:repair` → repara a tabela `flyway_schema_history`, removendo as migrations com falha (mais informações: <https://flywaydb.org/documentation/usage/commandline/repair>)

A large, stylized green leaf graphic is positioned on the left side of the slide, partially overlapping the title box. It has a light green fill and a darker green outline, with a central vein and several smaller veins branching out.

# Prática 02

# Adicionando as dependências

# Alterando a tabela *author*

- Agora vamos criar a versão 2 do banco de dados;
- Isso significa que o nome das migrations deve começar com V2;
- Crie dois arquivos na pasta ***src/main/resources/db/migration:***
  - ***V2.0\_\_alter\_author\_table\_add\_country.sql;***
  - ***V2.1\_\_update\_author\_table\_with\_country.sql;***
- Os códigos dos arquivos estão a seguir:

```
/* V2.0__alter_author_table_add_country.sql */  
  
alter table author  
    add column country varchar(40) not null;
```

```
/* V2.1__update_author_table_with_country.sql */  
  
update author set country = 'Índia' where id = 1;  
update author set country = 'Reino Unido' where id in (2,6);  
update author set country = 'Estados Unidos' where id in(3,4,5);
```



# Criando e populando a tabela *category*

- Vamos criar a tabela *category* para associar à tabela *book*;
- Crie dois arquivos na pasta ***src/main/resources/db/migration:***
  - ***V2.2\_\_create\_category\_table.sql;***
  - ***V2.3\_\_populate\_category\_table.sql;***

```
/* V2.2__create_category_table.sql */  
  
create table if not exists category (  
    id integer not null auto_increment,  
    name varchar(50) not null,  
    description varchar(100),  
    primary key(id)  
);
```

```
/* V2.3__populate_category_table.sql */  
  
insert into category (name, description)  
values  
    ('Ficção', 'Obras de ficção em geral'),  
    ('Ficção Científica', 'Obras de ficção que seguem um certo rigor científico'),  
    ('Ficção/Fantasia', 'Obras que misturam ficção e fantasia'),  
    ('Ficção/Espacial', 'Temática espacial/space opera')
```

# Alterando e atualizando a tabela *book*

- Agora que temos a tabela *category*, vamos adicionar um novo campo a tabela *book*, definí-lo como chave estrangeira e relacioná-lo à tabela *category*.
- Vamos também atualizar os dados da tabela *book*. Assim vamos criar duas *migrations*:
  - `V2.4__alter_book_table_add_category.sql`
  - `V2.5__update_book_table_with_category.sql`

```
/* V2.4__alter_book_table_add_category.sql */
```

```
alter table book  
  add column category_id integer,  
  add foreign key (category_id) references category(id);
```

```
/* V2.5__update_book_table_with_category */
```

```
update book set category_id = 1 where id in (1,2);  
update book set category_id = 4 where id in (3,4,6,9);  
update book set category_id = 2 where id in (5,7,8,10);
```

# Alterando novamente a tabela *book*

- Para preservar a integridade referencial da tabela *book*, vamos modificar o campo *category\_id* para *not null*;
- Crie a *migration*:
  - `V2.6__alter_book_table_modify_category_not_null.sql`;

```
/* V2.6__alter_book_table_modify_category_not_null */  
  
alter table book  
  modify category_id integer not null;
```

- Execute os comandos `mvn flyway:clean` e, logo em seguida, `mvn flyway:migrate`
- *Verifique o banco de dados!*

# Atividade

- Crie uma *migration* com prefixo **V3.0\_\_** que executa um *drop column* no campo *description* da tabela *category*.
- O projeto completo desse material está no GitHub:
  - <https://github.com/prof-jlgregorio/mybooks>

# Sobre mim

## JORGE LUÍS GREGÓRIO

- Professor da Faculdade de Tecnologia “Prof. José Camargo” – Fatec Jales, e da Escola Técnica Estadual Dr. José Luiz Viana Coutinho – Etec Jales;
  - Articulista do Jornal de Jales – Coluna “Fatecnologia”;
  - Apresentador do Tech Trends, podcast oficial da Fatec Jales;
  - Bacharel em Sistemas de Informação; Especialista em Desenvolvimento de Software para Web e Mestre em Ciência da Computação.
  - Trabalha com tecnologia desde 1998, tendo atuado como analista de suporte; administrador de redes de computadores; desenvolvedor de software, *webdesigner* e professor.
- 
- Site oficial: [www.jlgregorio.com.br](http://www.jlgregorio.com.br)
  - Perfil do LinkedIn: [www.linkedin.com/in/jlgregorio81](http://www.linkedin.com/in/jlgregorio81)
  - Currículo Lattes: <http://lattes.cnpq.br/3776799279256689>

