# 数据库迁移神器——Flyway

　　不知道你有没有遇到过这种场景，一套代码部署在不同的环境中，随着时间的过去，各个环境代码有版本差异，代码层面可以通过不同的版本来控制，但是数据库层面经常容易忘记更新!

　　比如刚开始环境 A 和环境 B 的代码版本是一样的，但是随着版本的迭代，环境 A 的系统一直持续迭代，但是环境 B 的系统由于种种原因没有升级，一直保持在最初的版本。如果某个时候需要对环境 B 的系统进行升级的话，你会发现，中间已经过了好多个版本，各个版本的差距很大，数据库结构有调整，不能直接打包发布，需要把之前所有对环境 A 调整的 SQL 都在环境 B 中执行一遍才行。

　　这个时候如果 SQL 版本做的好的问题不大，依次执行就行了，但是如果中间有人员离职或者记录缺失，那只能通过对比数据库结构来进行解决了。

　　**数据迁移**

前面我们的提到的场景专业的名词叫数据迁移，那为什么会出现数据迁移的场景呢?我从官网截了一张图大家可以看下，虽然说可能跟我实际开发不是一样，但是也差不多类似会出现这种场景，存在多个环境。可以看到虽然我们的代码可以通过版本迭代来控制，但是我们的数据库却不行，很多时候连脚本是否执行过都会忘记，这种事情光靠人记是很难的。



　　**Flyway**

Flyway 就是用来解决像这样的数据库迁移的工具，接入了 Flyway 过后，在数据库中会生成一张默认名为flyway\_schema\_history 的数据表，用来追踪数据库的变化。程序启动的时候 Flyway 都会在文件系统或者 classpath 路径下面寻找迁移脚本。每个迁移脚步都有相应的命名规则，Flyway 会根据文件的版本号进行迁移，每次迁移过后都会在flyway\_schema\_history 表中插入一条类似如下的记录，记录版本已经对应的脚本文件和校验码等信息：



每次启动的时候只会执行最高版本的脚本，而且如果版本没变，脚本变了是启动不了的。



　　**Flyway 的迁移类型**

**版本迁移**

　　最常见的迁移就是就是版本化迁移，每次迁移都会对应的迁移版本，迁移的版本必须全局唯一，版本迁移最大的特点就是依次只被执行依次。

　　**撤销迁移**

　　每个撤销迁移都对应的一个版本迁移，也就是说撤销迁移是针对版本迁移所存在的，每一个撤销迁移与版本迁移都是一一对应的，而且对应的版本号必须一致。

　　**可重复迁移**

　　可重复迁移有描述和校验码，但是没有版本号，程序在每次启动的时候，如果发现脚本文件有变化就会执行。

　　**基于 SQL 的迁移**

上面提到的几种类型都是基于 SQL 文件来执行的，只不过每种类型的命名格式不一样，下图是从官网上截下来的，大家看下每种类型的文件应该按照如下的格式去命令，其中的 Separator 是两个下划线。



　　主要分为下面几个部分：

　　prefix：前缀，不同的类型采用不同的前缀，版本迁移使用 V，撤销迁移使用 U，可重复迁移使用 R，当然这些都是可配置的;

　　Version：版本号，可以使用点符号或者单下划线链接;

　　Separator：分隔符，两个下划线，也是可以配置的;

　　Description：版本描述可以用下划线和空格分隔;

　　Suffix：后缀，一般都是 .sql

　**SpringBoot 项目接入 Flyway**

　　SpringBoot 项目接入 Flyway 非常简单，主要分为如下几步即可，我们依次来看一下。

**加入依赖**



　　在项目 pom.xml 文件中加入上面的依赖即可。

**增加配置**



　　这里因为很多情况下我们并不是一个新项目就开始使用 Flyway，而是项目在迭代中才引入的，所以上面的配置spring.flyway.clean-disabled=true 一定要禁用。上面几个配置由于已经继承到 SpringBoot 中了所以配置起来十分简单。

　**迁移脚本**

文件脚本的命名规则按照上面说的，我们这边采用版本迁移。我们创建一个版本的 SQL 文件放到对应的类路径文件夹里面，文件名叫V1.2\_\_create\_test\_table.sql，文件内容如下，然后我们启动项目。



启动过程中我们看到如下日志，显示了当前的版本，以及迁移的版本。



我们再查看数据库，首先 test\_table 已经创建成功了



另外我们在查看flyway\_schema\_history 表，会发现已经多了一条版本数据，至此我们介入 Flyway 已经成功了。



　　**总结**

　　今天给大家介绍了一个数据库版本迁移的工具，这么好的工具大家赶紧用起来吧，这样在以后的版本迭代的过程中再也不会忘记执行SQL 了。