|  |  |  |
| --- | --- | --- |
|  |  |  |
| spring.sql.init.mode | Використовується для визначення режиму виконання SQL-скриптів під час ініціалізації бази даних. Цей параметр дозволяє контролювати, коли і які SQL-скрипти будуть виконані під час створення або оновлення бази даних.  always: За цього режиму SQL-скрипти завжди виконуються при створенні або оновленні бази даних. Це означає, що кожного разу, коли додаток запускається або перезапускається, SQL-скрипти будуть виконані.  embedded: Цей режим використовується для вбудованих баз даних, таких як H2. SQL-скрипти виконуються лише у випадку, коли база даних створюється або оновлюється з нуля.  never: SQL-скрипти не будуть виконуватися автоматично ніколи. Ви повинні вручну виконати їх або використовувати інші механізми для виконання.  Наприклад, у режимі розробки ви, можливо, хочете використовувати `always`, щоб забезпечити, що ваша база даних завжди має правильну схему і дані. У режимі продакшн може бути більш доцільним використання `never`, оскільки ви хочете контролювати вручну процес виконання SQL-скриптів.  Цей параметр визначає, яким чином виконувати SQL-скрипти із файлів під час ініціалізації бази даних. Він має такі можливі значення:  always: Виконувати SQL-скрипти завжди, незалежно від того, який режим (`none`, `embed`, `script`) використовується для бази даних.  embedded: Виконувати SQL-скрипти, якщо база даних запускається в режимі вбудованої бази даних (наприклад, H2).  script: Виконувати SQL-скрипти, незалежно від типу бази даних, але тільки якщо вказано режим `always` або `embedded`.  none: Не виконувати SQL-скрипти незалежно від типу бази даних.  Параметр `spring.sql.init.mode` впливає на виконання SQL-скриптів, включаючи `data.sql`, при ініціалізації бази даних. Цей параметр приймає такі значення:  1. `never`: Скрипти не будуть виконані, незалежно від їх наявності.  2. `always`: Скрипти завжди будуть виконані під час ініціалізації бази даних.  3. `embedded`: Скрипти будуть виконані тільки для вбудованих баз даних.  Якщо ви використовуєте `spring.sql.init.mode=always` і маєте `data.sql` файл у відповідному місці (зазвичай в `src/main/resources`), то SQL-запити з цього файлу будуть виконані під час ініціалізації бази даних, незалежно від того, яка база даних використовується.  Важливо розуміти, як цей параметр впливає на ваш проект і вибирати підходящий режим в залежності від сценарію використання. |  |
|  | `spring.sql.init.mode` та `spring.datasource.initialization-mode` - це два різних параметри, які використовуються для ініціалізації бази даних при запуску Spring додатка. Ось їх різниця:  1. `spring.sql.init.mode`:  2. `spring.datasource.initialization-mode`:  Цей параметр вказує, коли і як виконувати скрипти із файлів для ініціалізації бази даних. Він має такі можливі значення:    - `always`: Виконувати скрипти при кожному запуску додатка.  - `embedded`: Виконувати скрипти тільки в тому випадку, якщо використовується вбудована база даних (наприклад, H2).  - `never`: Ніколи не виконувати скрипти.  Отже, різниця полягає у тому, що `spring.sql.init.mode` контролює, коли виконувати SQL-скрипти, незалежно від типу бази даних, а `spring.datasource.initialization-mode` визначає, коли і як виконувати скрипти в залежності від типу бази даних.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Так, параметр `spring.datasource.initialization-mode` пов'язаний з ініціалізацією бази даних, включаючи зовнішні файли типу `data.sql`. Однак, цей параметр впливає не тільки на файли `data.sql`, а також на інші внутрішні та зовнішні скрипти для ініціалізації бази даних, які можуть бути визначені в `schema.sql`, `data-h2.sql`, `data-mysql.sql` тощо.  Залежно від значення `spring.datasource.initialization-mode`, Spring Boot буде виконувати відповідні скрипти при ініціалізації бази даних. Ось як це працює:  1. `always`: Spring буде виконувати вказані скрипти завжди при кожному запуску додатка. Це може включати і файл `schema.sql`, і файл `data.sql`.  2. `embedded`: Це подібно до `always`, але скрипти будуть виконані тільки тоді, коли використовується вбудована база даних, така як H2.  3. `never`: Скрипти для ініціалізації бази даних не будуть виконані. Це корисно, коли ви хочете контролювати ініціалізацію бази даних самостійно.  Цей параметр впливає на всі типи скриптів, які можуть використовуватися для ініціалізації бази даних, і не обмежується лише зовнішніми файлами типу `data.sql`. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Ці параметри, мали б дозволити покласти файли в папку resources/mySubFolder  spring.datasource.schema=classpath:mySubFolder/schema.sql  spring.datasource.data=classpath:mySubFolder/data.sql  але чомусь не працюють, хоча в відео  Java Spring Boot - JPA - Hibernate - H2 - Database Initialization using data.sql and schema.sql  https://www.youtube.com/watch?v=9Yj2TCvrvaE&list=PLG5RS5k7TfUqDHBklOB35W3MamGLBLJQW&index=8  працювали.  Ці параметри не є аналогом:  spring.jpa.properties.hibernate.default\_schema=ИмяІснуючоїСхемиВDataBase  spring.jpa.properties.hibernate.hbm2ddl.import\_files=  що вони роблять ще не перевіряв. |  |
|  |  |  |
|  |  |  |
|  | `spring.jpa.defer-datasource-initialization` - це параметр у Spring Boot, який визначає, чи повинен додаток відкладати ініціалізацію джерела даних (datasource) до того часу, коли вся інша конфігурація біна `EntityManagerFactory` буде завершена.  Значення для `spring.jpa.defer-datasource-initialization` може бути:  1. `true` (за замовчуванням): Це означає, що ініціалізація джерела даних буде відкладена до того часу, коли всі біни будуть створені і всі налаштування будуть завершені. Це може бути корисним у випадках, коли ви працюєте з багатьма джерелами даних та хочете контролювати точний момент ініціалізації.  2. `false`: Це означає, що ініціалізація джерела даних буде відбуватися під час створення біна `EntityManagerFactory`. Це може бути корисним, якщо ви хочете перевірити з'єднання з базою даних в самому початку створення додатка.  Зазвичай, значення за замовчуванням (`true`) є прийнятним для більшості випадків. Однак, у деяких складних сценаріях ви можете встановити `spring.jpa.defer-datasource-initialization` в `false`, якщо ви хочете, щоб ініціалізація джерела даних відбувалася на пізньому етапі конфігурації.  Наприклад, для встановлення параметра `spring.jpa.defer-datasource-initialization` в `false`, ви можете додати наступний рядок до файлу `application.properties`:  ```properties  spring.jpa.defer-datasource-initialization=false |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |