Modularization of Spring Batch

What?

- Spring Boot 4 increased the modularization of its Spring Boot starter libraries

- One of the places where this is most noticeable is Spring Batch, as Spring Batch now has two starters instead of one.

- There's now a **spring-boot-starter-batch** and **spring-boot-starter-batch-jdbc** library

What's the difference?

- **spring-boot-starter-batch** relies on `ResourcelessJobRepository` and no longer requires a database to store metadata.

- Ideal for batch processes that do not require restartability or keeping metadata.

- **spring-boot-starter-batch-jdbc** contains the previous default behavior of Spring Batch, which stores batch metadata inside your database.

Watch out!

- Both starters have their own testing library

- `ResourcelessJobRepository` does not allow removing job instances or job executions

- This means you can no longer rely on `JobRepositoryTestUtils` to remove job executions and start with a clean slate

- So far, the only alternative I found is by restarting the application context by annotating your tests with `@DirtiesContext`

Watch out!

```java
@BeforeEach
void setUp() {
    // This does no longer work with
    // spring-boot-starter-batch!
    jobRepositoryTestUtils.removeJobExecutions();
}

@Test
// This does, but might impact performance
@DirtiesContext
void testBatch() throws Exception {
    // ...
}
```

Read more at
dimitri.codes/advent-of-spring