Title: How can I configure Maven Liquibase plugin in Spring Boot?

Post Body:

I am learning Liquibase and Spring Boot so I've created a simple project with Spring Initializr.

In the POM.xml file I've added:

```
    <plugin>         <groupId>org.liquibase</groupId>         <artifactId>liquibase-maven-plugin</artifactId>         <version
```

I've specified as property file the application.properties so all the configuration of my application can happen in a single file.

When I run any liquibase-maven-plugin task from IntelliJ I get different errors, here's an example running the changeLogSync task:

```
[ERROR] Failed to execute goal org.liquibase:liquibase-maven-plugin:3.4.1:changelogSync (default-cli) on project simpleTest: T
```

If I add the right keys in the application.properties I am able to make it work.

For example I've found that liquibase-maven-plugin will not read the **spring.datasource.url** property but it will only read the **url** property.

For this reason my application.properties will have to be something similar:

```
environment                          = JUnit spring.datasource.url               = jdbc:h2:file:./target/test spring.datasource
```

If I follow this pattern I'll end up having several keys with slightly different names but with the same values in my application.properties and this solution is clearly very ugly and inefficient.

What is an efficient and maintainable way to configure and use Liquibase Maven Plugin in Spring Boot?

Edit after the answer received from Amith Kumar:

```
environment=JUnit spring.datasource.url=jdbc:h2:file:./target/glossary-test spring.datasource.driver-class-name=org.h2.Driver
```

Error after the edit:

```
[ERROR] Failed to execute goal org.liquibase:liquibase-maven-plugin:3.4.1:dropAll (default-cli) on project test: Error setting
```

Accepted Answer:

`application.properties` settings are very fast to have an up and running application but not the best solution in terms of flexibility

My advice is to configure a datasource using `@Configuration`, example here

And then configure liquibase passing datasource defined above as follows

```
@Configuration public class LiquibaseConfigurer {      @Autowired     @Qualifier('primaryDataSource')     private DataSource o
```

In this case you just need `liquibase-core` dependency as follows

```
    <dependency>         <groupId>org.liquibase</groupId>         <artifactId>liquibase-core</artifactId>     </dependency>
```

A simpler alternative is to configure liquibase outside the application with no maven plugin.

Download library, or install it with some package manager, and launch a command line with all settings

```
liquibase --driver=org.h2.Driver \      --classpath=/path/to/h2/driver.jar \      --changeLogFile=/db/changelog/db.changelog-m
```

Anyway the problem you have now is because you've written this:

```
url=${spring.datasource.url}
```

I don't know where did you find this syntax but try to replicate connections url and replace with the following

```
url=jdbc:h2:file:./target/test
```

do the same for other settings

Highest Rated Answer:

Liquibase maven plugin supports configuration injection through pom.xml.

So you can use properties-maven-plugin to include your properties from application.properties (or use yaml-properties-maven-plugin if you are using application.yml), and then inject them into the liquibase configuration:

Example:

```
<plugin>     <groupId>it.ozimov</groupId>     <artifactId>yaml-properties-maven-plugin</artifactId>     <version>1.1.3</versio
```

Now you can inject these properties in liquibase configuration:

```
<plugin>             <groupId>org.liquibase</groupId>             <artifactId>liquibase-maven-plugin</artifactId>
```

I also needed to set the logicalFilePath to ensure that the changelog path inferred by spring boot integration and the maven plugin where the same.