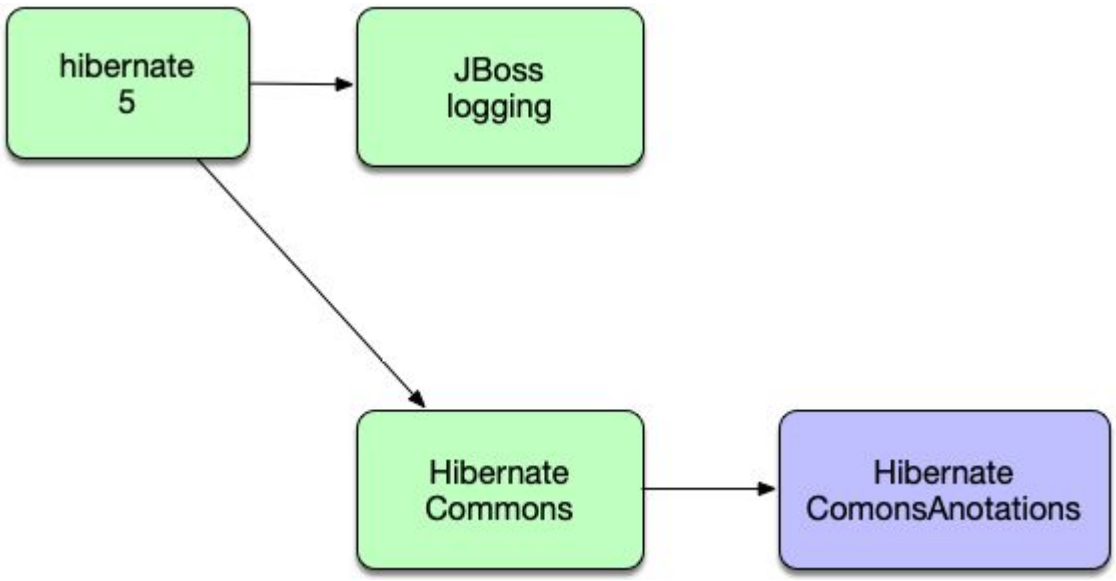
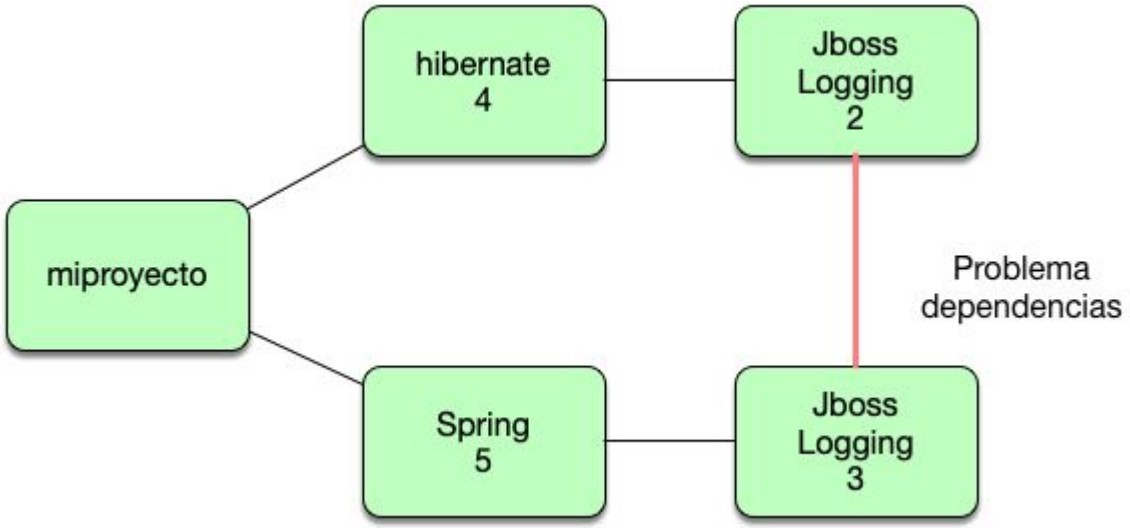
El concepto de Spring Boot Starter cada día se usa más pero a mucha gente le cuesta entender las ventajas que tiene . En muchos casos estamos acostumbrados a usar Maven como herramienta para gestionar las dependencias y el que aparezca algo nuevo en el horizonte siempre nos hace dudar, ya que estamos bastante contentos con cómo funciona Maven .

¿Ahora bien cual es el problema de Maven? .

Normalmente cuando nosotros definimos las dependencias de Maven tenemos la gran ventaja de que si por ejemplo usamos Hibernate 5.4 . Maven se encargará de instalar todas las dependencias que vengan asociadas al proyecto.



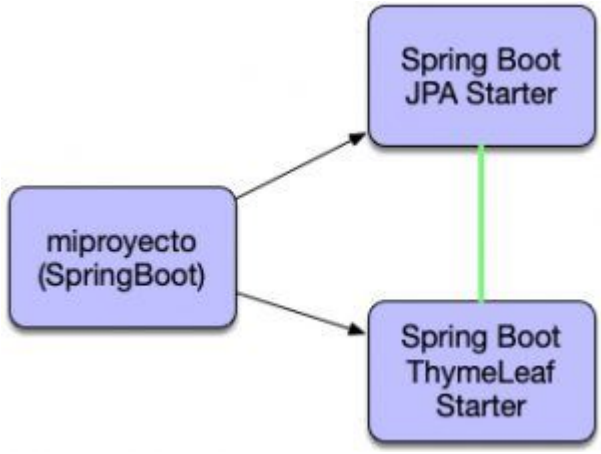
Esto muchas veces lo considerábamos suficiente . Sin embargo cuando más se complican los proyectos mas frameworks pueden llevar integrados entre ellos . Esto hace que las dependencias sean más y más complejas de gestionar y que algunas de ellas estén compartidas entre varios frameworks. Esto hace que el versionado se complique y que por ejemplo una versión de Hibernate no sea compatible con una versión de Spring ya que necesiten dependencias diferentes.



En muchos casos hace falta un experto para configurar una aplicación clásica de Spring algo que no es fácil de encontrar.

Spring Boot Starter y Simplificación

El concepto de Spring Boot Starter viene a solventar estos problemas . ¿Cuál es su enfoque?, pues generar dependencias que estén ligadas con Spring de forma directa. Es decir ya no tenemos que instalar JPA o no tenemos que instalar Thymeleaf como dependencias de Maven sino que lo que instalamos es un Starter de Spring . En estos casos Spring Boot JPA o Spring Boot ThymeLeaf. Al instalar estos Starters Spring Boot se encargará de hacer encajar las dependencias de tal forma que ambos proyectos puedan encajar de forma natural en nuestra solución con sus verisones correspondientes.



Spring Boot y Convenciones

Los starters de Spring boot no solo nos aportan una gestión correcta de las dependencias sino que ademas se encargan de definir convenciones automáticas que podemos utilizar directamente sin tener que complicarnos la vida . Por ejemplo supongamos que nos descargamos con Spring Initializer el Starter de ThymeLeaf , automaticamente contendrá una carpeta para ubicar sus plantillas sin que nosotros tengamos que complicarlos la vida

Maven y Boot

Una vez tenemos claro lo que es un starter abrir el fichero pom.xml muestra con claridad como de sencillo queda el proyecto:

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

```
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
thymeleaf</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>
```

Apenas hay dependencias . Aprendamos a utilizar los starters de Spring Boot para simplificar el desarrollo de aplicaciones y evitar errores en la gestión de dependencias.

Spring Boot Starter más comunes

Aquí te dejo una lista de los 10 starters más usados de Spring Boot,

1. spring-boot-starter-web: Es el que usas si vas a crear una aplicación web. Viene con todo lo necesario para manejar peticiones HTTP, tanto para crear APIs REST como para páginas web tradicionales. Incluye Tomcat como servidor web embebido.
2. spring-boot-starter-data-jpa: Si tu aplicación necesita conectarse a una base de datos, este starter es esencial. Te permite usar JPA para interactuar con bases de datos de manera sencilla. Además, trae Hibernate, que es el framework más usado para manejar las bases de datos relacionales.
3. spring-boot-starter-security: Te ayuda a añadir fácilmente autenticación y seguridad a tu aplicación. Ideal si necesitas gestionar logins o restringir accesos en ciertas áreas de tu app.
4. spring-boot-starter-thymeleaf: Este es para cuando necesitas generar vistas web con el motor de plantillas Thymeleaf. Muy útil para aplicaciones web donde tienes que mostrar

interfaces de usuario dinámicas.

5. spring-boot-starter-test: Si vas a hacer pruebas (y deberías), este starter incluye todo lo necesario para pruebas unitarias y de integración, con herramientas como JUnit y Mockito.
6. spring-boot-starter-actuator: Si quieres monitorizar tu aplicación y obtener métricas de rendimiento o salud del sistema, este es el starter que buscas. Te permite ver cómo está funcionando tu app en tiempo real.
7. spring-boot-starter-logging: Este starter te permite gestionar los logs de la aplicación con herramientas como Logback y SLF4J. Básicamente, te ayuda a ver qué está pasando en tu app mientras corre.
8. spring-boot-starter-validation: Ideal para validar datos de entrada en tus formularios o APIs. Usa Hibernate Validator y te facilita mucho el trabajo de verificar que los datos sean correctos.
9. spring-boot-starter-mail: Si necesitas que tu aplicación envíe correos electrónicos, este es el starter que te hace la vida fácil. Usa JavaMail y se configura de manera súper sencilla.
10. spring-boot-starter-cache: Si tu aplicación necesita cachear datos para mejorar el rendimiento, este es el que te va a ayudar a implementarlo sin mucho esfuerzo.

Otros artículos relacionados

1. Spring Boot Thymeleaf y su configuración
2. Spring REST CORS y su configuración
3. Curso Spring Boot y sus tecnologías
4. Spring boot Yaml y propiedades