Laboratorio N° 4 Componentes y conectores

Santiago Cárdenas Amaya

Juan Pablo Fonseca

Escuela Colombiana de ingeniería Julio Garavito

Bogotá 2023

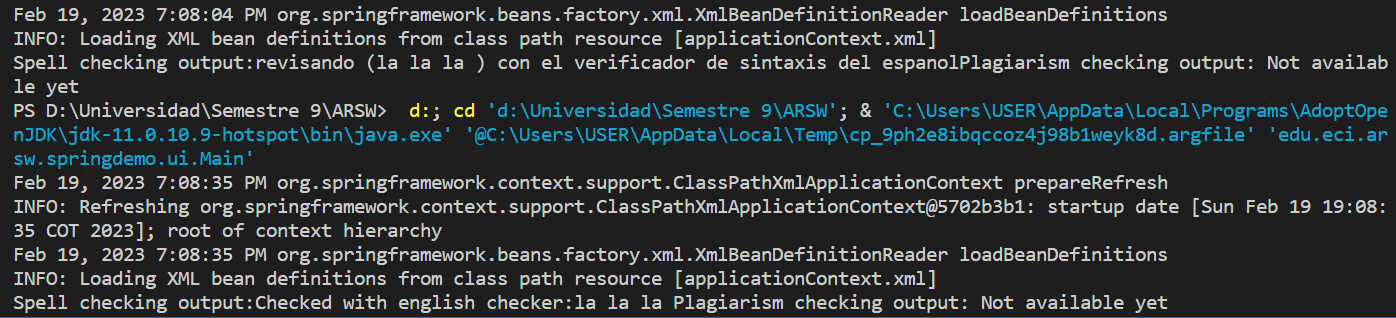
Introducción

“No hay que casarse con una tecnología”. Esta frase iba en torno al hecho de que muchas veces uno se familiariza demasiado con un lenguaje, al punto de que no se quieren hacer las cosas fuera de este lenguaje, pero esta frase tiene mas aplicaciones, desde el uso de distintos frameworks con un mismo fin, hasta diagramas que pueden presentar una mayor claridad de las cosas.

En este documento se podrá apreciar el uso de springboot como framework de inyección de dependencias, y aunque no es muy remarcado a lo largo del desarrollo, el conocimiento de un nuevo diagrama que puede proporcionar igual o mas información valiosa que el tradicional diagrama de clases, se está haciendo referencia al diagrama de componentes.

# Ejercicio introductorio spring

Es la siguiente imagen se evidencia el resultado de haber inyectado inicialmente SpanishSpellChecker, y debajo de lo que está en azul es lo que imprimió al inyectar EnglishSpellChecker.



# Ejercicio original

1. Configure la aplicación para que funcione bajo un esquema de inyección de dependencias, tal como se muestra en el diagrama anterior.

Lo anterior requiere:

* Agregar las dependencias de spring
* Agregar la configuración de spring
* Configurar la aplicación -mediante anotaciones- para que el esquema de persistencia sea inyectado al momento de ser creado el bean 'BlueprintServices'.

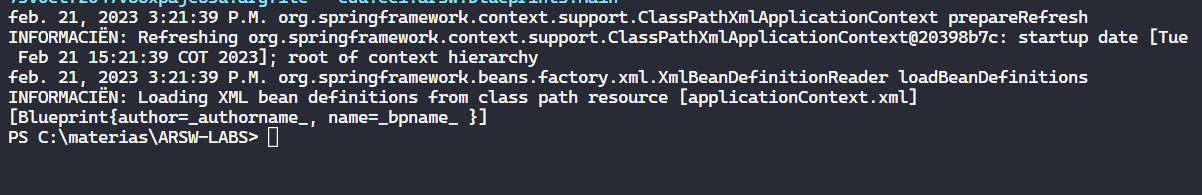
Se puede ver el desarrollo en el commit “p1” (77430a6acb)

1. Complete los operaciones getBluePrint() y getBlueprintsByAuthor(). Implemente todo lo requerido de las capas inferiores (por ahora, el esquema de persistencia disponible 'InMemoryBlueprintPersistence') agregando las pruebas correspondientes en 'InMemoryPersistenceTest'.

Se puede ver el desarrollo en el commit “p2” (addfc25281)

1. Haga un programa en el que cree (mediante Spring) una instancia de BlueprintServices, y rectifique la funcionalidad del mismo: registrar planos, consultar planos, registrar planos específicos, etc.

Se probó que la clase se pudiese instanciar generando una clase main y se instanció correctamente. La siguiente imagen muestra que la ejecución de Main.java no generó ningún error.



1. Se quiere que las operaciones de consulta de planos realicen un proceso de filtrado, antes de retornar los planos consultados. Dichos filtros lo que buscan es reducir el tamaño de los planos, removiendo datos redundantes o simplemente submuestrando, antes de retornarlos. Ajuste la aplicación (agregando las abstracciones e implementaciones que considere) para que a la clase BlueprintServices se le inyecte uno de dos posibles 'filtros' (o eventuales futuros filtros). No se contempla el uso de más de uno a la vez:

- (A) Filtrado de redundancias: suprime del plano los puntos consecutivos que sean repetidos.

- (B) Filtrado de submuestreo: suprime 1 de cada 2 puntos del plano, de manera intercalada.

Los 2 fueron realizados junto a sus debidas pruebas.

1. Agregue las pruebas correspondientes a cada uno de estos filtros, y pruebe su funcionamiento en el programa de prueba, comprobando que sólo cambiando la posición de las anotaciones -sin cambiar nada más-, el programa retorne los planos filtrados de la manera (A) o de la manera (B).

Usando FilterRedundancia.java





Usando FilterSubmuestreo.java





Diagrama de componentes.

Diagrama

Descripción generada automáticamente

# Conclusiones

Se puede concluir que el uso de frameworks distintos a los que se está ya acostumbrado pueden facilitar la vida, como lo es el caso de springboot. Adicional a esto se pudo elaborar correctamente un diagrama de componentes, el cual permite conocer de un vistazo el funcionamiento de distintos servicios aislados.