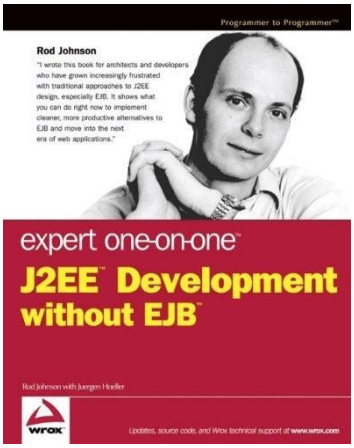
Spring Framework

Spring Framework es una plataforma de trabajado creada por Rod Johnson (https://en.wikipedia.org/wiki/Rod_Johnson_(programmer)) en el año 2003. Durante ese año Rod escribió un libro titulado “Expert One-on-One J2EE Development without EJB”, en el cual hablo por primera vez de spring y de las ventajas que suponía este framework versus las tecnologías oficiales java existentes en esa época.



Simplicidad, es la palabra que define el desarrollo a través de Spring. El imagino el desarrollo en Java de otra manera, más robusto y configurable, con más facilidades y posibilidades para los desarrolladores. Esto se ve reflejado en este framework. La búsqueda de ser simple.
Springframework nos da la ventaja de eliminar de nuestro código la responsabilidad de toda la creación de objetos y paso de dependencias, con lo que tendremos un código más limpio.

Springframework se basa en los estándares oficiales J2EE, la web oficial del framework es: https://spring.io

Spring Framework

¿Qué es Spring Framework?

Spring Framework es una plataforma que nos proporciona una infraestructura que actúa de soporte para desarrollar aplicaciones Java. Spring maneja toda la infraestructura y así te puedes centrar en tu aplicación. Diciéndolo más coloquialmente, Spring es el "Pegamento" que une todos los componentes de la aplicación, maneja su ciclo de vida y la interacción entre ellos.

> *"Que es un framework… Desde un punto de vista sencillo, es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación."*

Spring Framework es un contenedor ligero en contraposición a un servidor de aplicaciones J2EE. En el caso de una aplicación web, te basta con un contenedor de servlets como Tomcat o Jetty. Pero Spring no solo se puede usar para crear aplicaciones web, se podría usar para cualquier aplicación java, aunque su uso habitual sea en entornos web, nada te impide utilizarlo para cualquier tipo de aplicación.

> *"Un contenedor es un componente especial que permite contener en su interior a otros componentes, incluidos otros contenedores. Un contenedor posee, además de la habilidad de contener otros componentes, la de organizar dichos componentes en su interior, manejar su ciclo de vida, entre otras características."*

Nació en una época en la que las tecnologías empresariales oficiales de Java (J2EE) tenían todavía mucho por mejorar. Los servidores de aplicaciones eran monstruosos devoradores de recursos y los EJB eran pesados, inflexibles y era demasiado complejo trabajar con ellos. En ese contexto, Spring popularizó ideas como la inyección de dependencias o el uso de objetos convencionales (POJOs) como objetos de negocio, que suponían un poco de aire fresco. Estas ideas permitían un desarrollo más sencillo y rápido y unas aplicaciones más ligeras. Eso posibilitó que de ser un framework inicialmente diseñado para la capa de negocio pasara a ser un completo stack de tecnologías para todas las capas de la aplicación.

> *"Spring es un framework alternativo al stack de tecnologías estándar en aplicaciones JavaEE."*

Básicamente, la mayor diferencia que podemos encontrar entre desarrollar con Spring y con J2EE es la posibilidad de usar un servidor web al estilo Tomcat para desplegar la aplicación. Las tecnologías J2EE más sofisticadas requieren del uso de un servidor de aplicaciones, ya que las APIs son implementadas por el propio servidor, mientras que Spring no es más que un conjunto de librerías portables entre servidores. La idea es obstaculizar lo menos posible una posible portabilidad a J2EE, idea que es de agradecer en un mundo en que todos los fabricantes intentan de una forma u otra mantener sus herramientas.

Una vez que hemos leído las definiciones de lo que es Springframework... lo primero que tenemos que entender de Spring… es que es un conjunto de módulos, de los cuales podemos utilizar los que queramos. No estamos atados a utilizar todo el framework completo en los desarrollos que emprendamos. El corazón de Spring, el "core", realiza lo que es llamado inversión de control / inyección de dependencias. (Temas explicados más adelante) Básicamente lo que significa esto es que la creación de nuestros objetos las llevara a cabo un contenedor de recursos inyectándolos a otros objetos que dependientes, sin tener que logra sin que nuestro código tenga dependencia alguna con Spring, salvo la clase que cree el contenedor Spring.

Springframework nos da la ventaja de eliminar de nuestro código la responsabilidad de toda la creación de objetos y paso de dependencias, con lo que tendremos un código más limpio.

Spring Framework

¿Qué es Spring Framework?

El Universo de los Frameworks

Normalmente cuando trabajamos en cualquier plataforma solemos utilizar algún tipo de framework. Estos no son ni más ni menos que un conjunto de clases que nos facilitan el trabajo. Utilizamos el framework para crear un conjunto de objetos que nuestra aplicación necesita.

En la mayoría de las ocasiones para desarrollar aplicaciones no es suficiente usar un único framework sino que necesitamos utilizar varios. Cada uno de los cuales generará su propio conjunto de objetos.

Esta situación genera problemas ya que cada framework es totalmente independiente y gestiona su propio ciclo de vida de los objetos.

Spring ayuda a solventar este problema ya que cambia las responsabilidades y en vez de que el propio desarrollador sea el encargado de generar los objetos de cada uno de los frameworks es Spring quien se encarga de construir todos los objetos que la aplicación va a utilizar.

Springframework, es un contenedor, pero no es solo un framewrok más. Es un contenedor que gestiona el ciclo de vida de los objetos y como se relacionan entre ellos. Proporciona una gran infraestructura que permite que el desarrollador se dedique a la lógica de la aplicación.

Es Ligero, es muy rápido en tiempo de procesamiento y no es intrusivo a la hora de desarrollar. Esto último es uno de sus puntos más fuertes.

Está orientado a aspectos, soporta la programación orientada a aspectos, lo que permite facilitar una capa de servicios que son ideales para este tipo de programación como auditoría, o gestión de transacciones.

Spring Framework

Spring Modules

Spring Framework está organizado en módulos. Estos módulos son agrupados en:

1. Core Container.
2. Data Access/Integration.
3. Web.
4. AOP (Aspect Oriented Programming), Instrumentation.
5. Test.

Desde un punto de vista más genérico, Springframework puede verse como un soporte que nos proporciona tres elementos básicos:

1. Servicios empresariales: Podemos hacer de manera sencilla que los objetos sean transaccionales, o que su acceso esté restringido a ciertos roles, o que sea accesible de manera remota y transparente para el desarrollador, o acceder a otros muchos servicios más, sin tener que escribir el código de manera manual.

2. Estereotipos configurables: Se puede anotar clases indicando por ejemplo que pertenecen a la capa de negocio o de acceso a datos. Se pueden configurar nuestros propios estereotipos. Por ejemplo, podríamos definir un nuevo estereotipo que indicara un objeto de negocio que además sería cacheable automáticamente y con acceso restringido a usuarios con determinado rol.

3. Inyección de dependencias: La inyección de dependencias nos permite solucionar de forma sencilla y elegante cómo proporcionar a un objeto cliente acceso a un objeto que da un servicio que este necesita. Por ejemplo, que un objeto de la capa de presentación se pueda comunicar con uno de negocio. En Spring las dependencias se pueden definir con anotaciones o con XML.

Adicionalmente Spring tiene una serie de proyectos independientes basados en el core del Framework que nos permiten completarnos al momento de seleccionarlo como plataforma de desarrollo:

1. Spring Data.
2. Spring Integration.
3. Spring Batch.
4. Spring Security
5. Spring Web Flow
6. Spring Boot

Y muchos otros más …

Spring Framework

Inyección de Dependencias (DI)

¿Qué es?

Originalmente la inyección de dependencias se llamaba inversión de control (Muchos autores intercambian el mismo concepto al hablar de ellas), pero en el 2004 Martin Fowler llegó a la conclusión que realmente no se invertía el control, sino las dependencias.

La DI (dependency injection) permite a un objeto que conozca sus dependencias mediante una interfaz y no por su implementación. De esta forma, la implementación puede variar sin que el objeto dependiente se dé cuenta. La gran ventaja de la DI es el acoplamiento débil entre objetos.

El objetivo es lograr un bajo acoplamiento entre los objetos en nuestra aplicación. Con este patrón, los objetos no crean o buscan sus dependencias, sino que éstas son suministradas al objeto a través de un tercero (El contenedor). El contenedor (la entidad que coordina cada objeto en el sistema) es el encargado de realizar este trabajo al momento de instanciar el objeto. Se invierte la responsabilidad en cuanto a la manera en que un objeto obtiene la referencia a otro objeto.

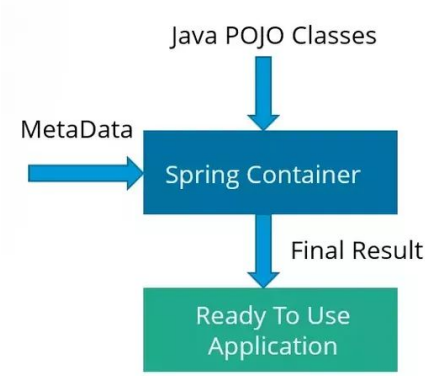
De esta manera, los objetos conocen sus dependencias por su interfaz. Así la dependencia puede ser intercambiada por distintas implementaciones a través del contenedor. En resumen, programaremos orientado a interfaces e inyectaremos las implementaciones a través del contenedor.

Inversión de Control (IoC)

¿Qué es?

En inglés, conocido como Inversion of Control (IoC), es un estilo de programación en el cual un framework o librería controla el flujo de un programa. Esto representa un cambio con respecto a paradigmas tradicionales donde el desarrollador especifica todo el flujo del programa.

Es el principio subyacente a la técnica de Inyección de Dependencias, siendo términos frecuentemente confundidos.



Spring Framework

Ventajas de DI/ IoC

1. Eliminación de instrucciones new: Al usar este tipo de patrón, la primera consecuencia más observable a primera vista es que el código permitirá que las instancias inyectadas no se generarán desde dentro a través del uso de la palabra "new" si no que se harán por reflexión o mediante algún framework.

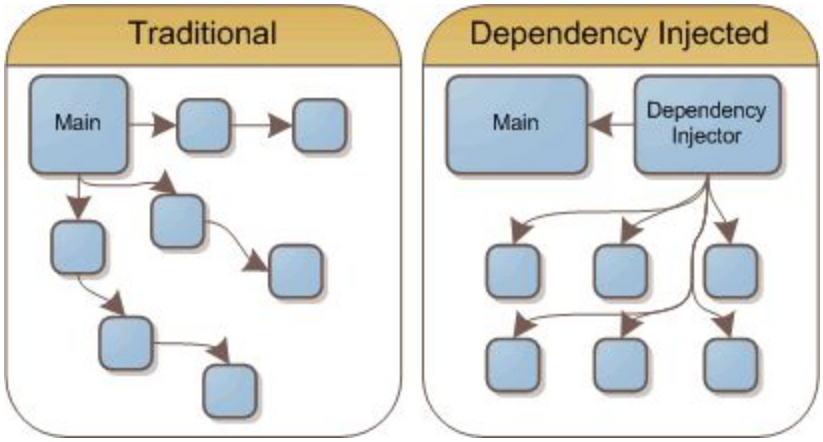
2. Sustitución de piezas software (Legos): Como se mencionó anteriormente, una ventaja es la posibilidad de modificar una pieza de software por otra sin necesidad de reprogramar las clases que lo van a utilizar. Esto puede ser interesante en desarrollos en los que es posible añadir o quitar elementos variables como plugins.

3. Mejoras en los test: Debido a la no utilización de la instrucción new, se tendrá de una gran ventaja para el testeo automático de aplicaciones que consiste en hacer mocking. Esta técnica consiste en simular piezas de software dependientes y reemplazarlas por otras más simples y más rápidas para ejecutar multitud de test.

4. Depuración de clases ya compiladas: Aunque se tiene grandes ventajas con el uso del mocking, muchas veces el problema en los test es proveer lo que no se conoce. Caracteres ocultos, datos incorrectos, valores extraños y mucha información corrupta puede hacer que nuestro sistema no funcione bien y no lo hayamos previsto. En ese caso los test automáticos no funcionarán y tendremos que depurar el código para encontrar ese misterioso

problema. Esto no es algo propio del patrón de inyección de dependencias, pero sí que el patrón conduce o induce a crear un diseño modular y con componentes independientes y como consecuencia a crear diferentes proyectos para cada módulo.

5. Configuraciones: Las configuraciones también son tareas a añadir en el caso de utilizar inyección de dependencias. Al hacer la inyección se traslada la responsabilidad de la instanciación de un código imperativo a un código declarativo. Estas declaraciones pueden ser de muchas formas como xml, anotaciones, archivos de texto, bases de datos, etc. Dependerá del framework o de nuestro propio algoritmo de inyección el que se use una forma u otra.

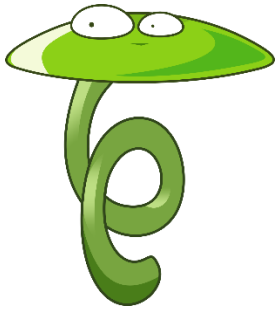


Spring Framework

¿Qué es un Bean?

Es un componente de software que tiene la particularidad de ser reutilizable. En el lenguaje de programación java cumplen varios criterios:

1. Tienen todos sus atributos privados (private).
2. Tienen métodos set() y get() públicos de los atributos privados que nos interese.
3. Tienen un constructor público por defecto.



Beans en Springframework

A diferencia de los bean tradicionales que representan una clase, los en Springframework son objetos creados y manejados por el contenedor Spring.

El contenedor se encuentra en el núcleo del marco de trabajo de este y utiliza inyección de dependencias para gestionar los componentes que forman la aplicación. Se encarga de varias tareas, como crear, conectar y alojar los objetos definidos por los beans. Además, hace de dispensador proporcionando beans por petición. El contenedor se encarga de cargar las definiciones de beans.

Tipos de contenedor de Spring:

1. Bean Factor: Contenedor sencillo con soporte básico de inyección de dependencias.

2. Aplication Context: Es una implementación de un bean factory que proporciona opciones avanzadas como:

    a. Medios para resolver mensajes de texto e internalizacion,
    b. Publicación de beans registrados como receptores o formas genéricas de abrir recursos de archivo.

Spring Framework

¿Qué es un Bean?

Ciclo de vida de un Bean en Springframework

Los beans de Springframwork tienen un ciclo de vida en el contexto de la aplicación. Podemos ordenar las fases de la vida de un bean de la siguiente forma:

1. Instanciación.

2. Inyección de las propiedades.

3. Nombre del vean.
4. Postprocesado (pre inicializacion)
5. Inicialización.
6. Postprocesado (post inicialización)
7. Bean listo para uso.
8. Destrucción.

Formas de crear un Bean en Springframework

1. Bean simples: No poseen atributos.
2. Bean con inyección por constructor: Pasando los atributos por constructor.
3. Bean con referencias de objeto de constructores: Cuando pasamos un bean como atributo del constructor de otro.
4. Bean con inyección de propiedades: Cuando en vez del método constructor utilizamos métodos setters de atributos.
5. Con valores simples: Enteros, reales, etc.
6. Con valores complejos:

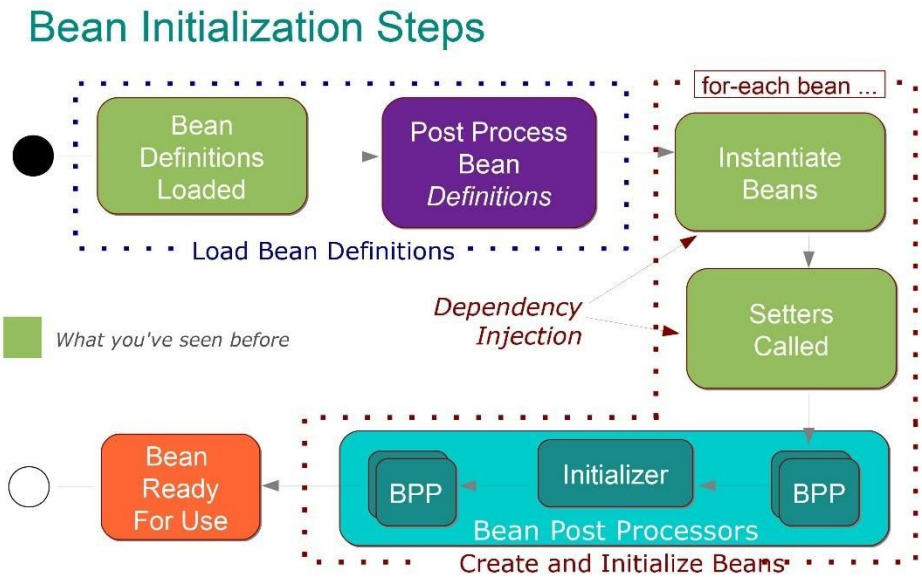
a. Por referencia de otro objeto: Pasando un bean id al método set.
b. Colecciones de datos: listas, arrays, maps.
c. Con valor nulo: cuando necesitamos pasar un valor nulo.

Ámbito de un Bean

Cuando trabajamos con un Bean en Springframework por defecto son singletons. ¿Que significa esto? Significa que el contenedor solo instancia un objeto de la clase, y cada vez que se pide una instancia del bean en realidad se obtiene una referencia al mismo objeto.

El ámbito singleton es el indicado en muchos casos, pero si necesitamos cambiar este comportamiento, podemos asignar otros ámbitos para el vean como:

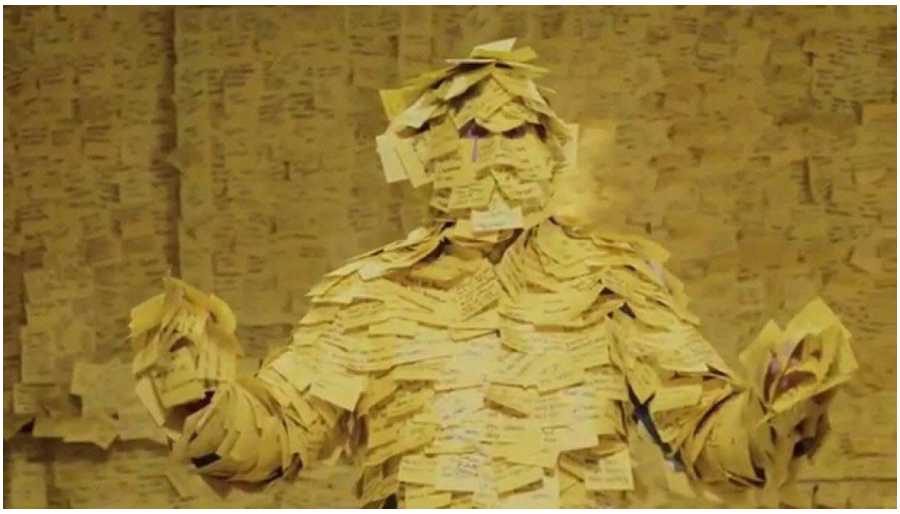
1. Para especificar que queremos una nueva instancia cada vez que se solicite el bean, se usa el valor prototype.
2. En aplicaciones web, se pueden usar los ámbitos de request y session.



Spring Framework

Anotaciones

Las anotaciones en Java son formas de añadir metadatos al código fuente para que esten disponibles para la aplicación en tiempo de ejecución. Las Anotaciones Java pueden añadirse a los elementos de programa tales como clases, métodos, campos, parámetros, variables locales, y paquetes.



Springramework posee una lista de aplicaciones, las cuales usaremos durante el tiempo que desarrollemos con el framework, a continuación, las más comunes:

1. @Bean, define un dentro del application context.
2. @Scope, indica el ámbito del Bean.
3. @Service, Indica que el Bean creado es un posible servicio.
4. @Component, Indica que el Bean creado es un posible componente.
5. @Repositoy, Indica que el Bean creado es un objeto de acceso a datos.
6. @Controller, Indica que el Bean creado es un componente Web.
7. @PostConstruct, Indica que el método en el Bean será disparado luego de ser llamado el constructor de la clase.
8. @PreDestroy, Indica que el método en el Bean sera disparado luego de ser eliminado el Bean del contexto.

9. @Autowired, Indica que el contexto debe de inicializar el argumento marcado en el Bean.