educación IT
CODE YOUR FUTURE

Bootcamp Java Developer

Fase 3 - Java Architect

Módulo 21

educación IT
CODE YOUR FUTURE

Spring Framework

Introducción

Spring Framework es una plataforma de trabajo creada por **Rod Johnson**, en el año 2003. Durante ese año, Johnson escribió un libro titulado “Expert One-on-One J2EE Development without EJB”, en el que habló por primera vez de Spring y de las ventajas que suponía este *framework* en comparación con las tecnologías oficiales Java existentes en esa época.

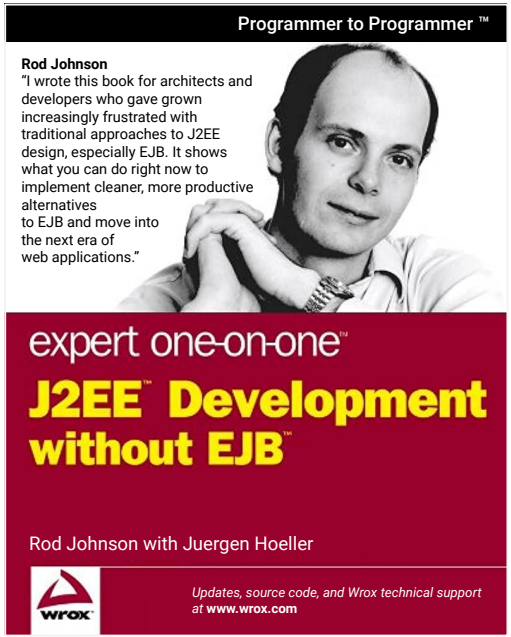


Imagen: Editorial Wrox; *New edition* (18 de junio de 2002).

***Simplicidad* es la palabra que define el desarrollo a través de Spring**. Johnson imaginó el desarrollo en Java de otra manera, más robusta y configurable, con más facilidades y posibilidades para los desarrolladores. En este *framework* se refleja la búsqueda de ser simple.

Spring Framework brinda la ventaja de eliminar de nuestro código la responsabilidad de toda la creación de objetos y paso de dependencias, lo que resulta en un código más limpio.

Spring Framework se basa en los **estándares oficiales J2EE**, la web oficial del framework es: Spring.io

¿Qué es Spring Framework?

Es una plataforma que proporciona una **infraestructura** que actúa de soporte para desarrollar **aplicaciones Java.** Spring maneja toda la infraestructura y así puedes centrarte en tu aplicación.

Dicho de manera más coloquial, Spring es el "pegamento" que une todos los componentes de la aplicación, maneja su ciclo de vida y la interacción entre ellos.

> "¿Qué 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".
>
> **Tom DeMarco**
>
> Libro *"Análisis Estructurado y Diseño"*, publicado por primera vez en 1978.

Spring Framework es un contenedor ligero en contraposición a un servidor de aplicaciones J2EE. En el caso de una aplicación web, 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 impide su utilización en 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.*

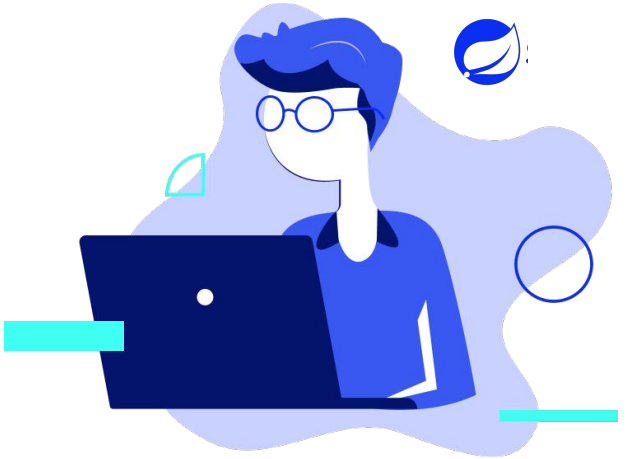
Spring Framework nació en una época en la que las tecnologías empresariales oficiales de Java (J2EE) tenían todavía mucho por mejorar. Los servidores de aplicaciones eran monstruosos devoradores de recursos y los EJB eran pesados, inflexibles y era demasiado complejo trabajar con ellos. En ese contexto, **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 se han leído las definiciones de Spring Framework, lo primero que se debe saber es que Spring es un **conjunto de módulos** entre los cuales se pueden **seleccionar aquellos que se deseen utilizar**. No estamos atados a utilizar el *framework* completo en los desarrollos que emprendamos. El corazón de Spring (“*core*”) realiza lo que se llama ***inversión de control / inyección de dependencias***. Básicamente, esto significa que la creación de nuestros objetos la llevará a cabo un contenedor de recursos, que los inyectará en otros objetos que dependan de ellos, sin que nuestro código tenga ninguna dependencia directa con Spring, excepto la clase que crea el contenedor de Spring.

Spring Framework brinda 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.



El universo de los frameworks

Normalmente, cuando se trabaja en cualquier plataforma, se suele utilizar algún tipo de *framework*. Estos no son ni más ni menos que un **conjunto de clases que facilita el trabajo.** El *framework* se utiliza para crear el conjunto de objetos que necesita la aplicación.

En la mayoría de las ocasiones, **para desarrollar aplicaciones no alcanza un único framework** sino que necesitamos utilizar varios. **Cada uno de ellos 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 lugar 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.**

Como ya mencionamos pero es válido recordar:

- Spring Framework es un contenedor, pero no es solo un *framework* más. Es un contenedor que **gestiona el ciclo de vida de los objetos** y cómo 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.

¡Sigamos trabajando!