1.1 ¿Qué es Spring?

Spring es un *framework* de código abierto para el desarrollo de aplicaciones (empresariales) Java. Su origen está en el trabajo de Rod Johnson, que trabajaba como consultor externo, y que plasmó en el libro Expert One-to-One J2EE Design and Development (Octubre, 2002). Según Johnson, el modelo de los *Enterprise Java Beans* era excesivamente tedioso y pesado para muchas aplicaciones desarrolladas hasta el momento. Por ello, condensó su experiencia y sus buenas prácticas en un conjunto de clases que fueron el origen del framework.

Algunas de las críticas de Johnson al uso de EJBs son (ver *J2EE development without EJB (2004)*):

- Complejidad (descriptores, implementaciones de interfaces, …) y baja productividad del programador.
- Modelo remoto solamente basado en RMI.
- Muchas aplicaciones no necesitan componentes remotos.
- Difíciles de depurar (volver a hacer *deployment* y volver a arrancar).
- Mapeo O/R basado en *entity beans* limitado (por ejemplo, no existe la herencia)
- …

Spring popularizó desde su inicio ideas como la inyección de dependencias (que aprenderemos más adelante), el uso de *POJOs* (*Plain old Java Object*) como objetos de negocio, etc… que suponían un cambio radical con respecto al estándar; de esta forma, las aplicaciones podían ser más ligeras, y posibilitó que un framework que estaba inicialmente ideado para la capa de negocio se convirtiera en un *stack* de tecnologías para todas las capas de la aplicación.

Spring está basado en los siguientes principios:

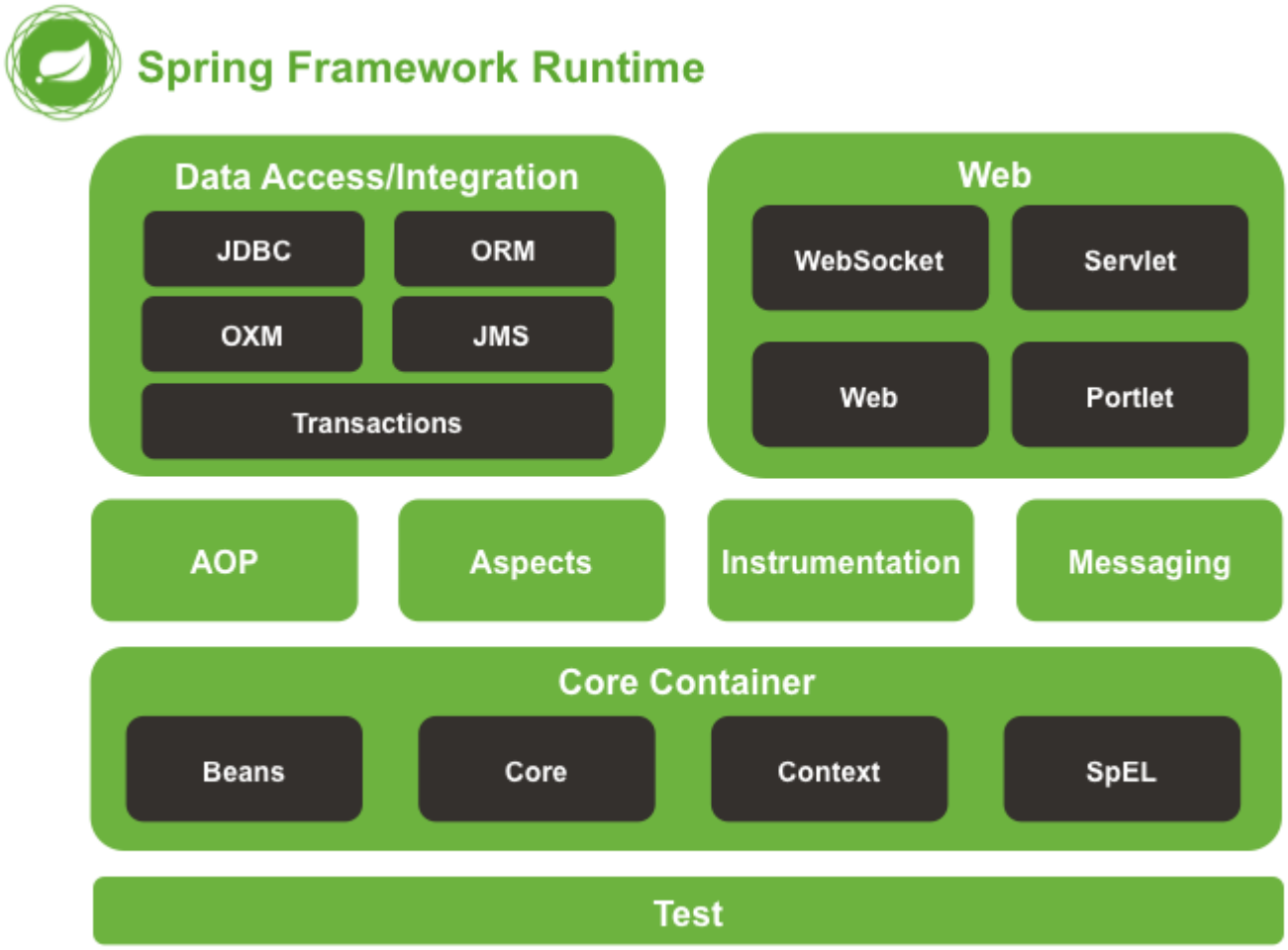
- El buen diseño es más importante que la tecnología subyacente.
- Los Java Beans ligados de una manera más libre entre interfaces es un buen modelo.
- El código debe ser fácil de probar.

Desde hace tiempo, estas ideas tan innovadoras popularizadas por Spring se han ido incorporando poco a poco al estándar, de forma que Spring y JavaEE han convergido mucho. Aun así, Spring ha conseguido crear una comunidad de desarrolladores en torno a sus diferentes tecnologías, siendo una alternativa necesaria de conocer, y muy utilizada en el entorno empresarial. En la actualidad, las aportaciones más novedosas de Spring se centran en campos de BigData, NoSQL, HTML5, Mobile, aplicaciones *sociales*, seguridad…

A día de hoy, una de las principales diferencias que podemos encontrar entre JavaEE y Spring es la posibilidad de usar un servidor web más convencional (estilo Tomcat) para desplegar la aplicación (con Spring Boot aprenderemos que nos podemos olvidar incluso de esta parte). JavaEE requiere el uso de un servidor de aplicaciones que, normalmente, requerirá de mayor conocimiento para su configuración y despliegue, y necesitará de unos recursos más potentes para *correr* cualquier aplicación.

1.2 Módulos de Spring

En la siguiente imagen, podemos ver las diferentes tecnologías (en terminología Spring, proyectos) que conforman a día de hoy Spring:



Algunos de los módulos más importantes de Spring

- **Core**: La parte fundamental de este *framework* es el módulo *Core*, y los adyacentes *Bean* y *Context*. Proveen toda la funcionalidad para la inyección de dependencias, permitiéndole administrar la funcionalidad del contenedor de Beans (trabajaremos sobre estos conceptos de forma más amplia en las próximas lecciones). Además, también proveen de los servicios Enteprise como JDNI, EJB, …
- **AOP**: Se trata de un módulo que nos permitirá utilizar el paradigma de Programación Orientada a Aspectos (*Aspect Oriented Programming*). Este paradigma permite mejorar la modularización y separar las responsabilidades. De esta forma, podemos separar las

funcionalidades comunes, que se utilizan transversalmente a lo largo de toda la aplicación, de aquellas que son propias de cada módulo.

- **Data**: Se trata de un gran módulo, formado por múltiples submódulos, y que nos permite simplificar el acceso y persistencia de datos. *Spring Data* nos proporciona soporte para usar base de datos relacionales (*JDBC*), ORMs (como por ejemplo *JPA*, *Hibernate*, …) e incluso modelos de persistencia NoSQL (como por ejemplo, *MongoDB*).
- **Web**: Este módulo nos permitirá implementar el patrón Modelo-Vista-Controlador (MVC) de una manera sencilla y limpia, haciendo uso de forma transparente también de otros patrones de diseño, como *FrontController*. De esta forma, podemos separar limpiamente la lógica de negocio de la presentación de los datos y el acceso a los mismos. Además de aplicaciones que implican el uso de vistas y formularios, también podremos crear servicios web (por ejemplo, al estilo REST) de una forma sencilla y rápida.

La modularidad de Spring nos permitirá la posibilidad de usar solo algunas de las partes del framework, y poder combinar esta con otros frameworks diferentes:

- Un proyecto que implemente MVC mediante el uso de Struts podría incorporar el contenedor de IoC mediante el uso de Spring (sin necesidad de utilizar Spring Web MVC).
- Una aplicación web desarrollada con Spring Web MVC podría implementar su capa de datos mediante el uso de Hibernate (sin hacer uso de Spring Data).

1.3 Versiones de Spring

La primera versión de Spring se publicó en marzo de 2004. Actualmente, la versión más estable en la actualidad de Spring es la 4.3.2, si bien ya podemos encontrar en fases preliminares la versión 5.0 (que nos permitirá trabajar con JDK 9).

Versión	Descripción
1.0	Primera versión de Spring
2.0	Espacios de nombres XML, soporte para AspectJ
2.5	Configuración a través de anotaciones
3.0	Actualización a Java5+, Configuración a través de Java (`@Configuration`)

A continuación, podemos ver algunos de los elementos que se han incorporado en las diferentes versiones 4.X

Versión	Cambios introducidos
4.0	• Mejora de la documentación. • Eliminación de código *deprecado*. • Adaptación al uso de Java 8 • Basado en Java EE 6 (JPA 2.0, Servlet 3.0), con soporte para Java EE 7 (JPA 2.1, JTA 1.2, …)

Versión	Cambios introducidos
	• Mejoras en el contenedor de inversión de control • Mejoras en el módulo web para la creación de servicios REST (`@RestController`, `AsyncRestTemplate`) • Mejoras en las prestaciones de *Java Message Services*. • Soporte para anotaciones JCache (JSR-107)
4.1	• Mejoras varias en el módulo web • Mejoras en el módulo de WebSocket. • Varias mejoras en el módulo *core* (sobre todo a nivel de configuración a través de Java). • Mejoras en el módulo de acceso a datos (como por ejemplo, soporte para Hibernate 5.0).
4.2	• Mejoras en JMS • Mejoras en el módulo web (integración de OkHTTP, soporte para CORS, anotaciones propias el mapeo, con `@RequestMapping` como metaanotación, …) • Mejoras en el módulo *core*: métodos *default* de Java 8, mayor información en las excepciones, soporte de inyección por constructor para `@Configuration`…
4.3	• Mejoras en el módulo web: anotaciones compuestas para `@RequestMapping`: `@GetMapping`, `@PostMapping`, `@PutMapping`, `@DeleteMapping`, y `@PatchMapping`; nuevas anotaciones compuestas para los ámbitos web: `@RequestScope`, `@SessionScope`, `@ApplicationScope`; otra serie de nuevas anotaciones… • Mejoras en los módulos de WebSocket y Testing