"Presentación de Spring Framework"

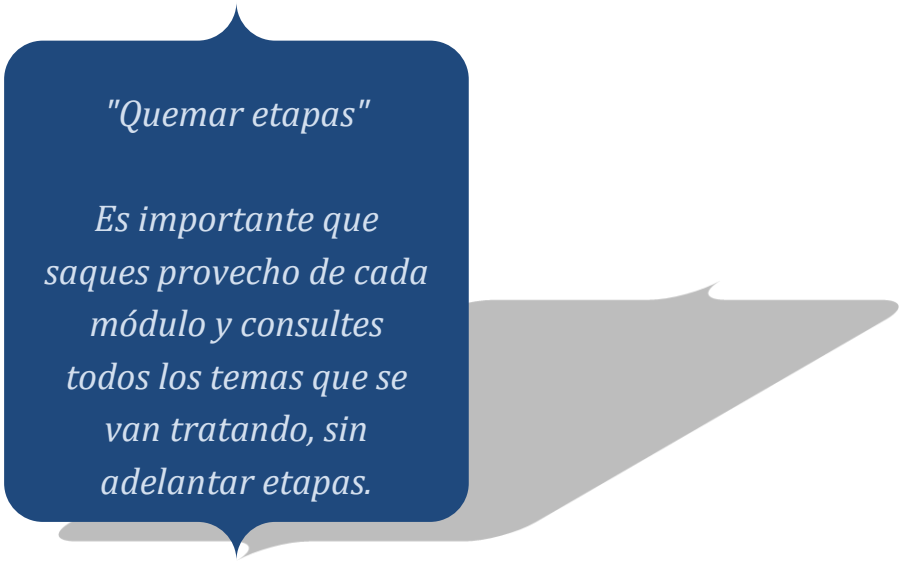
Módulo 1 / antes de empezar

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

Objetivos

El objetivo de este módulo semanal es hacer **la primera presentación formal** de Spring como una de las principales tecnologías y Framework de Java EE, multiplataforma y de arquitectura multicapa para el desarrollo de aplicaciones empresariales de la plataforma Java Oracle

La intención es comprender y discutir todo el alcance y las posibilidades de la herramienta y culminar está primera etapa con **la instalación de una aplicación base** y realizar la infalible y nunca bien valorada prueba clásica de *"hola mundo"* ☺



Introducción

El Spring Framework (también conocido simplemente como Spring) es un framework

de **código abierto utilizado por excelencia** para el **desarrollo de aplicaciones empresariales** para la plataforma **Java**. La primera versión fue escrita por Rod Jonhson, quien lo lanzó primero con la publicación de su libro Expert One-on-One Java EE Design and Development (Wrox Press, octubre 2002). También hay una versión para la plataforma .NET, Spring.net.

El framework fue lanzado inicialmente bajo Apache 2.0 License en junio de 2003. El primer gran lanzamiento hito fue la versión 1.0, que apareció en marzo de 2004 y fue seguida por otros hitos en septiembre de 2004 y marzo de 2005.

A pesar de que Spring Framework no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores en Java al considerársele una alternativa y sustituto del modelo de Enterprise JavaBean. Por su diseño el framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria.

Mientras que las características fundamentales de este framework pueden emplearse en cualquier aplicación hecha en Java, existen muchas extensiones y mejoras para construir aplicaciones basadas en web por encima de la plataforma empresarial de Java (Java Enterprise Platform).

Dos de los objetivos más importantes de Spring es permitir que el desarrollo se concentre en la lógica del negocio y que se haga empleando buenos principios de diseño orientado a objetos. Para lograrlo se utiliza un concepto muy interesante llamado Inversión del Control, también conocido como el principio Hollywood: "No nos llames, nosotros te llamaremos." Esto permite que el código escrito por los desarrolladores para la lógica principal del sistema no tenga dependencias sobre las clases del framework; lo cual redunda en un código mucho más limpio y con la posibilidad de utilizar todas las ventajas de la programación orientada a objetos (específicamente la herencia).

Es uno de los frameworks más utilizados al día de hoy y no es casualidad. El objetivo de Spring es simplificar el desarrollo de aplicaciones empresariales Java.

A grandes rasgos, un framework es un **conjunto de componentes**, **APIs** y de clases que nos permiten **resolver un problema en específico**, que ya tiene resuelto el tema del diseño y de la arquitectura. En el caso particular de Spring, nos permite resolver muchos de los problemas que se presentan al desarrollar aplicaciones con tecnología JEE (Persistencia, Mvc, Inyección de Dependencia, AOP etc., con **bajo acoplamiento** y **alta cohesión** etc).

Una de las mayores ventajas de Spring, **es la forma modular** en el que fue creado, permitiendo **habilitar/deshabilitar** las características a utilizar según se requiera.

La página oficial de Spring es http://www.springsource.org donde se pueden encontrar todos los proyectos relacionados con dicha tecnología. Spring es utilizado en proyectos muy diversos, como puede ser en Instituciones Bancarias, Aseguradoras, Instituciones Educativas y de Gobierno, entre muchos otros tipos de proyectos y empresas.

Un poco de Historia

Los primeros componentes de lo que se ha convertido en Spring Framework fueron escritos por Rod Johnson en el año 2000, mientras trabajaba como consultor independiente para sus clientes en la industria financiera en Londres. Mientras escribía el libro Expert One-on-one J2EE Design And Development (Programmer to programmer), Rod amplió su código para sintetizar su visión acerca de cómo las aplicaciones que trabajan con varias partes de la plataforma J2EE podían llegar a ser más simples y más consistentes que aquellas que los desarrolladores y compañías estaban usando por aquel entonces.

En el año 2001 los modelos dominantes de programación para aplicaciones basadas en web eran ofrecidas por el API Java Servlet y los Enterprise JavaBeans, ambas especificaciones creadas por Sun Microsystems en colaboración con otros distribuidores y partes interesadas que disfrutaban de gran popularidad en la comunidad Java. Las aplicaciones que no eran basadas en web, como las aplicaciones basadas en cliente o aplicaciones en batch, podían ser escritas con base en herramientas y proyectos de códigos abiertos o comerciales que proveyeran las características requeridas para aquellos desarrollos.

Rod Johnson es reconocido por crear un framework que está basado en las mejores prácticas aceptadas y amplio uso de patrones de diseños GOF, y ello las hizo disponibles para todo tipo de aplicaciones, no sólo aquellas basadas en web. Estas ideas también estaban plasmadas en su libro y, tras la publicación, sus lectores le solicitaron que el código que acompañaba al libro fuera liberado bajo una licencia open source.

Se formó un pequeño equipo de desarrolladores que esperaba trabajar en extender el framework y un proyecto fue creado en Sourceforge en febrero de 2003. Después de trabajar en su desarrollo durante más de un año lanzaron una primera versión (1.0) en marzo de 2004. Después de este lanzamiento Spring ganó mucha popularidad en la comunidad Java, debido en parte al uso de Javadoc y de una documentación de referencia por encima del promedio de un proyecto de código abierto.

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

Sin embargo, Spring Framework también fue duramente criticado en 2004 y sigue siendo el tema de acalorados debates. Al tiempo en que se daba su primer gran lanzamiento muchos desarrolladores y líderes de opinión vieron a Spring como un gran paso con respecto al modelo de programación tradicional; esto era especialmente cierto con respecto a Enterprise JavaBeans. Una de las metas de diseño de Spring Framework es su facilidad de integración con los estándares J2EE y herramientas comerciales existentes. Esto quita en parte la necesidad de definir sus características en un documento de especificación elaborado por un comité oficial y que podría ser criticado.

Spring Framework hizo que aquellas técnicas que resultaban desconocidas para la mayoría de programadores se volvieran populares en un periodo muy corto de tiempo. El ejemplo más notable es la inversión de control. En el año 2004, Spring disfrutó de unas altísimas tasas de adopción y al ofrecer su propio framework de programación orientada a aspectos (aspect-oriented programming, AOP) consiguió hacer más popular su paradigma de programación en la comunidad Java.

En 2005 Spring superó las tasas de adopción del año anterior como resultado de nuevos lanzamientos y más características fueron añadidas. El foro de la comunidad formada alrededor de Spring Framework (The Spring Forum) que arrancó a finales de 2004 también ayudó a incrementar la popularidad del framework y desde entonces ha crecido hasta llegar a ser la más importante fuente de información y ayuda para sus usuarios.

En el mismo año los desarrolladores del proyecto abrieron su propia compañía para ofrecer soporte comercial y establecieron una alianza con BEA. En diciembre de 2005 la primera conferencia de Spring fue realizada en Miami y reunió a 300 desarrolladores en el transcurso de tres días, seguida por una conferencia en Amberes en junio de 2006, donde se concentraron más de 400 personas.

¿Por qué Spring?

Seamos claros: Si eres o quieres ser un profesional del desarrollo de software sabrás que **los que sean capaces de diseñar y crear aplicaciones empresariales usando Spring tendrán trabajo por mucho tiempo**. Y es que la penetración y crecimiento de Spring crece exponencialmente todos los años.

Spring permite desarrollar aplicaciones flexibles, con alta cohesión y con un bajo acoplamiento.

Spring permitió simplificar el desarrollo JEE al utilizar clases Java Simples (POJO – Plain Old Java Object) para la configuración de servicios. Debido a que muchos proyectos muestran las mismas tareas a realizar una y otra vez, tales como Localización de Servicios, Manejo de Transacciones, Manejo de Excepciones, Parametrización de la aplicación, entre muchos más.

Spring permite resolver muchos de estos problemas de manera muy simple. Para lograr lo anterior el framework se base en dos conceptos fundamental:

- **DI (Dependency Inyection)**: El objetivo es lograr un **bajo acoplamiento** entre los objetos de nuestra aplicación. Con este patrón de diseño, los objetos no crean o buscan sus dependencias (objetos con los cuales colabora) sino que éstas son

proporcionadas o inyectadas al objeto, por ejemplo inyectar objetos a una clase (POJO) que tiene dependencias, en lugar de ser ella misma sea quien las instancie. 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.

- **AOP (Aspect Oriented Programming)**: AOP es un paradigma de programación que permite **modularizar las aplicaciones y mejorar la separación de responsabilidades** entre módulos y/o clases. Se trata de un paradigma de programación que intenta **separar las funcionalidades secundarias** de la lógica de negocios. En inglés denominan a estas funcionalidades "cross-cutting concerns" algo que se traduciría como "preocupaciones transversales". Por ejemplo los **loggers**, la **seguridad**, el manejo de **transacciones**, etc., son funcionalidades que atraviesan nuestro programa en varias abstracciones de éste. Por lo tanto corremos el riesgo de caer en la repetición de código y el acoplamiento entre nuestra lógica de negocios y la implementación de los cross-cutting concerns.

Las características anteriores son la base para la creación de contenedores ligeros (lightweight containers). Spring es uno de los contenedores ligeros más completos y populares al día de hoy.

Las cuatro claves de beneficios de Spring

Modularidad

Plain Old Java Objects mantienen su código limpio, simple y modular, bajo acoplamiento y alta cohesión.

Productividad

Más del 70% de los desarrolladores reportan ganancias de productividad y una reducción en el tiempo de desarrollo e implementación con Spring.

Portabilidad

Las aplicaciones se ejecutan en Tomcat, y en todos los servidores de aplicaciones Java EE, tales como JBoss, Glassfish etc., así como en plataformas en la nube.

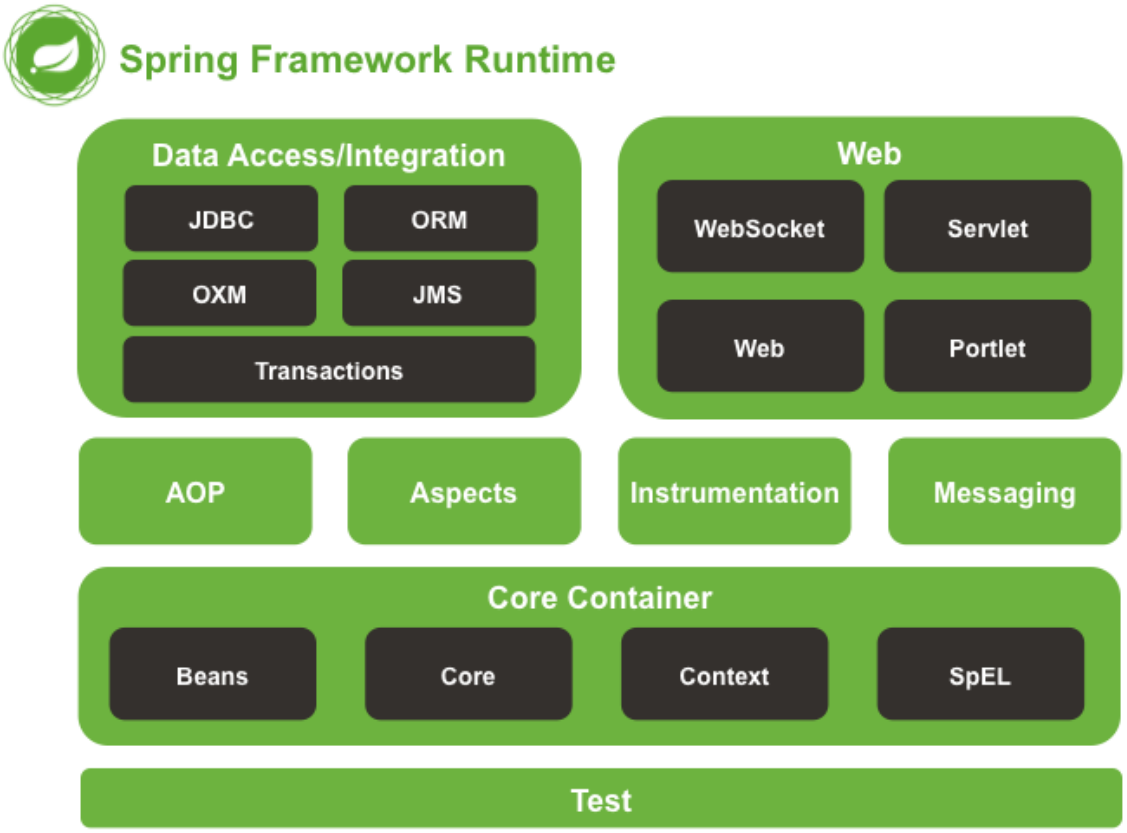
Capacidad de pruebas unitarias

Dependencias limpias, actualizadas y los justo y necesaria, aseguran que la integración con unit testing sea muy simple.

Lo último en tecnología y funcionalidad

Spring se compone de distintos módulos, permitiendo **seleccionar solo algunos de ellos o todos**, dependiendo de la naturaleza de la aplicación. En la figura podemos observar algunos de los módulos principales del Framework de Spring. A continuación listaremos varios de ellos:

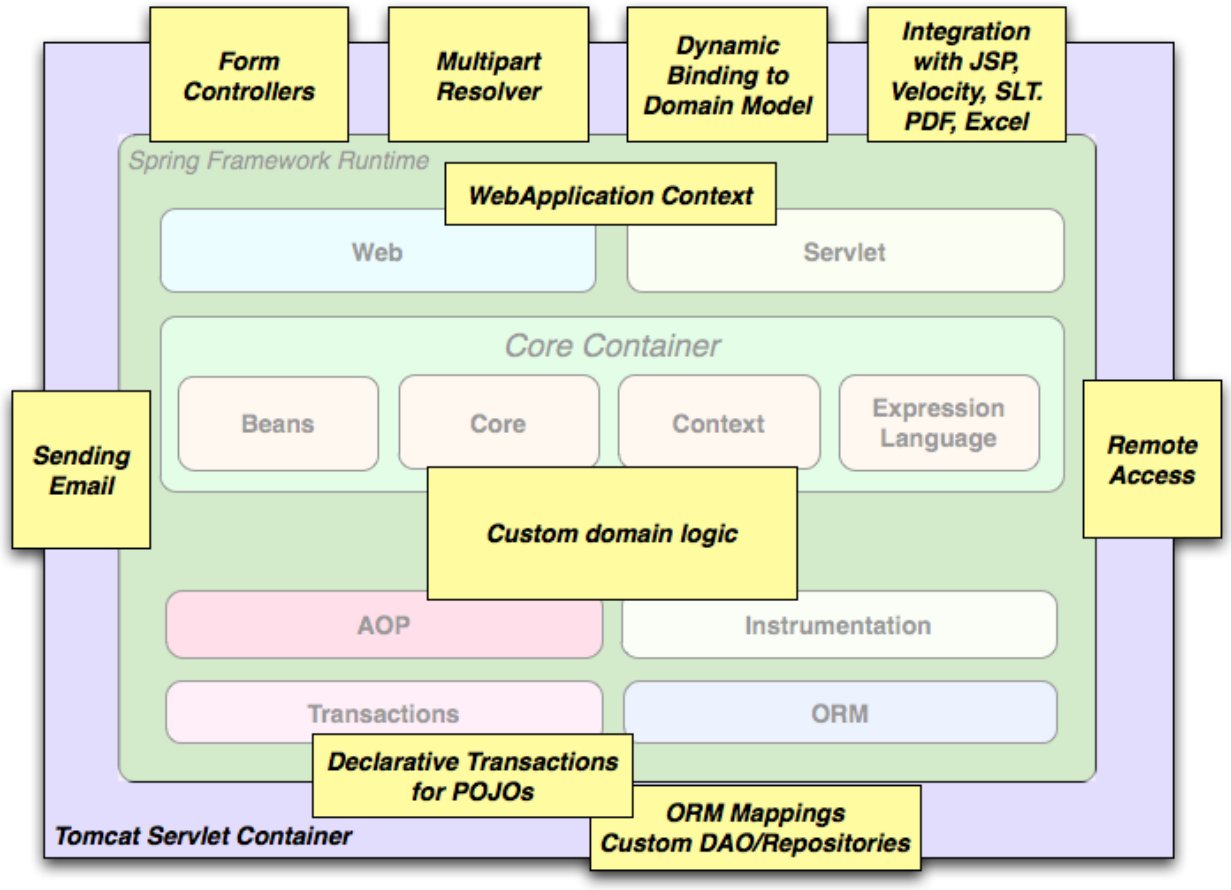
- **Spring Core**: Este módulo provee la funcionalidad básica de la fábrica de Spring. El componente principal es BeanFactory, el cual aplica el concepto de Inversion of Control (IoC) o también conocido como Dependency Injection (DI).
- **Spring Context**: Aquí es donde se realiza la configuración del framework, generalmente en un archivo de configuración XML. Incluye la configuración de beans de la capa Web MVC, de servicios empresariales tales como DataSource, JDBC, Hibernate, Repository, Internacionalización, validación, entre varios más.
- **Spring AOP**: Permite aplicar los conceptos de Programación Orientada a Aspectos (AOP), además incluye clases de soporte para el manejo transaccional, logger, la seguridad, entre varias clases más, permitiendo desacoplar estas características de nuestra aplicación. Además nos permite desarrollar interceptores de método y puntos de corte para desacoplar el código de las funcionalidades transversales.
- **Spring DAO Support**: Permite aplicar conceptos de la capa de datos Data Access Object (DAO) a través de POJOs (Plain Old Java Object), abstrayendo la complejidad, permitiendo crear un código JDBC, Hibernate u JPA más limpio y simple. Además cuenta con una capa de excepciones sobre los mensajes de error provistos por cada servidor específico de base de datos y con manejo de transacciones a través de AOP.
- **Spring ORM**: Permite integrarse con tecnologías tales como JPA, Hibernate e iBatis, entre otras integradas con DAO Support.
- **Spring Web**: Permite el desarrollo y la integración con tecnologías como Struts, JSF, Tapestry, entre otros.
- **Spring MVC**: Este módulo implementa el patrón MVC para ser utilizado en la capa de presentación.



El diagrama muestra los módulos con los que cuenta Spring (hasta la versión 2.5). En su núcleo (**Core**) se encuentra el **BeanFactory** – el contenedor fundamental de Spring y quien se encarga de la inyección de dependencias. El contenedor **ApplicationContext** se basa en BeanFactory y extiende su funcionalidad con soporte para i18n, eventos de ciclo de vida, validación y mejor integración con AOP.

Arquitectura

La arquitectura de una aplicación empresarial en Java se compone de distintas capas, cada capa tiene una función muy específica. Dividir una aplicación en capas tiene varias ventajas, como son separación de responsabilidades, un mejor mantenimiento a la aplicación, especialización de los programadores en cada capa, entre muchas más. Spring es un framework que resuelve varios problemas de distintas capas, desde la capa de presentación, la capa de negocio y la capa de datos. Aunque a veces también se combina con otras tecnologías y Spring quede como el orquestador de la capa de Servicio.



A continuación mencionaremos cada una de las capas de una aplicación multicapas.

- ✓ **Capa Cliente o Web**: la capa web que puede residir en un servidor web, las tecnologías más básicas que podemos encontrar en este servidor web son los JSP's, los Servlets y Spring MVC o bien la capa de una aplicación cliente con el método main.
- ✓ **Capa de Negocio**: en esta capa podemos encontrar tecnología como son los Enterprise Java Beans (EJBs) o frameworks como Spring.
- ✓ **Capa de Datos**: aquí vamos a encontrar tecnologías como JDBC, Hibernate, entre otras. Este código nos va a permitir comunicarnos con nuestra base de datos para leer y almacenar información en ella.

Plenamente probado, seguro y confiable

Spring se prueba constantemente mediante técnicas de test unitario de Java, desde el principio, con estrictos requisitos en materia de calidad de código para asegurarse de que todo el código contribuido no sólo ha sido objeto testeado, además de ser estable y fácil de extender y de mantener. Con todo esto **buscan garantizar que podemos crear nuestras propias aplicaciones** a partir del framework de Spring.

Además Spring cuenta con **Spring Security** una potente herramienta de autenticación, altamente configurable, un completo framework de control de acceso. Es uno de los proyectos de Spring más maduros y ampliamente utilizados, desarrollado en 2003 y mantenido activamente por **SpringSource**

Spring ha sido **probado** y **utilizado** en diversos proyectos alrededor del mundo, como en Instituciones Bancarias, Aseguradoras, Instituciones Educativas y de Gobierno, entre muchos otros tipos de proyectos y empresas.

Spring vs otros Frameworks

Spring vs Struts2

Hay un punto bien importante que los diferencia enormemente, y es que Struts2 es sólo un Framework Web MVC mientras que Spring además de tener un componente Web MVC tiene varios componentes más por ejemplo para la persistencia que integra diferentes Framework de persistencia y ORM como Hibernate JPA, IbTIS JDO etc.. Además de los componentes IoC para trabajar con inyección de dependencia, diferente resoluciones de vista hasta integra componentes como Jasper, EJB, WS, AOP etc, es decir es un mundo mucho más amplio que struts, por lo tanto lo hace mucho más grande, completo y robusto.

Además ambos se puede integrar, por ejemplo usar el MVC de struts y todo lo que es persistencia e inyección se hace con spring.

Spring vs EJB

Comparando Spring Framework y la plataforma EJB3, podríamos decir que no son tan comparables en muchos aspectos, por ejemplo spring es un Framework Java EE (como tal con todas sus letras) que tiene componentes web con mvc, persistencia, ioc, aop, forms, layout, pdf, rest, validaciones etc, no sólo está relacionado a la lógica de negocio y acceso a datos sino también a la capa web e incluso aplicaciones standard alone, mientras que EJB es sólo persistencia, transacciones, seguridad, aop y lógica de negocio (no web), solo podríamos hacer un comparativo sobre el acceso a datos de spring con el uso de ejb y persistencia básicamente.

Por otro lado los componentes de spring, los beans no se despliegan de forma remota, sólo local dentro de un proyecto, mientras que los EJB3 están basado en CORBA y los beans se pueden desplegar en un servidor de aplicaciones y acceder de forma local y remota.

Por lo tanto podemos decir que son tecnologías complementarias, ya que podríamos tener un spring web mvc que trabaja la lógica de negocio y persistencia a través de los

ejb y no con su propio componente Hibernate dao support u otro, pero pueden ser sustitutivas en el lado de la persistencia trabajar la persistencia con spring data access o ejb persistencia, dos caminos y alternativas, incluso en un proyecto podría ser ambas con ejb que accede a datos desde otro server y locamente accedemos a los datos con spring, las variaciones son infinitas.

Sin duda Spring es un Framework complejo, pero como todo en la vida es tire y floja, practica y práctica.

En Resumen

Spring proporciona cada uno de los componentes para muchos otros requisitos comunes en el desarrollo de aplicaciones empresariales, flexibles, escalables, extensibles, con alta cohesión y con un bajo acoplamiento, permitiendo seleccionar solo algunos de ellos o todos, dependiendo de las características del proyecto a desarrollar: Spring **Core**, Spring **Context**, Spring **AOP**, Spring **DAO** Support, Spring **ORM**, Spring **Web**, Spring **MVC**.

Todos estos paquetes de Software están disponibles de modo gratuito para su descarga, cuenta con una excelente documentación y foros de desarrolladores muy activos y amplios. Desde luego ya sabemos que para desarrollar en Spring, podemos usar distintos sistemas operativos y distintas configuraciones de Software.

> ***"Si he llegado más lejos ha sido apoyado en los hombros de gigantes"***
>
> Isaac Newton (1642-1727)

Fin.

Envía tus consultas a los foros!

Aquí es cuando debes sacarte todas las dudas haciendo consultas en los foros correspondientes

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com