1. ¿Qué es SpringFramework?

Es un framework ligero para el desarrollo de aplicaciones en Java.

Contenedor de Inversión de control, es un integrador de aplicaciones y framework para desarrollar aplicaciones java.

***¿Qué es un framework?*** *Es un conjunto estandarizado de buenas prácticas, patrones de diseño, librerías, conceptos y criterios que ayudan a acelerar el desarrollo de aplicaciones.*

*No son una aplicación por sí misma, pero sirven como base para la construcción de otras aplicaciones, y al estar basadas en buenas prácticas favorecen la calidad de código y la reutilización de los componentes entre proyectos.*

Por tanto, Spring es un framework de propósito general, eso significa que a diferencia de otros frameworks como Struts, JSF o Tapestry que están orientados a la construcción de aplicaciones web, Spring puede crear aplicaciones de todo tipo:

* Aplicaciones de consola
* De escritorio
* Web
* Remotas
* Sociales
* Móviles
* Batch
* Integración de sistemas

# ¿Que es Spring Framework?

[Manual de Spring](http://www.programacionj2ee.com/category/manual-de-spring/)

Spring Framework nos provee de un conjunto de conceptos, técnicas y una metodología de programación para el desarrollo de aplicaciones informáticas de alta calidad.

Spring fue escrito originalmente para la plataforma J2EE de Java, plataforma orientada al desarrollo de aplicaciones web y ha ido evolucionando rápidamente hasta el día de hoy, donde podemos encontrar diferentes ramas de desarrollo de la mano de SpringSource y todo su equipo de desarrolladores.

## ****Principales características****

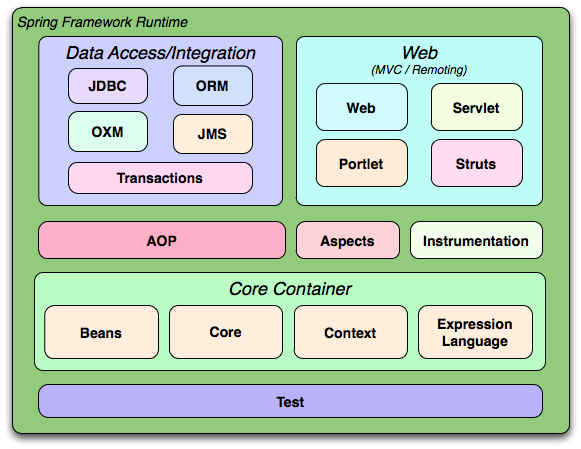
Spring Framework es en la actualidad la referencia en el mundo de los frameworks de programación para los desarrolladores web de todo el mundo. Su éxito se fundamenta en la constante labor de investigación e innovación que realiza su equipo de desarrollo.

Se trata de un framework que impulsa una metodología de trabajo ágil, eficiente y de buena praxis, lo que resulta en la creación de Software de elevada calidad y mantenibilidad.

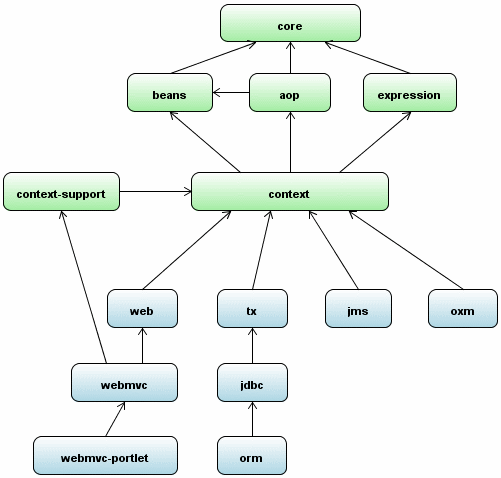
Tiene amplia compatibilidad para la integración con otros frameworks y librerias de uso común para la creación de aplicaciones web, desde Composite Views como Velocity o Tiles, APIs en capa de persistencia como Hibernate o JDO, y otra miscelanea de recursos como JavaMail, Quartz, etc…

## ****Modulos de Spring Framework****

Dentro del paquete estándar de Spring Framework podemos encontrar el siguiente conjunto de modulos, con el siguiente esquema general.

[caption id="attachment\_396" align="aligncenter" width="581"][](http://www.programacionj2ee.com/wp-content/uploads/2013/08/spring-modules.png)Esquema de Spring Framework[/caption]

En el siguiente esquema podemos ver las dependencias de cada modulo.

[caption id="" align="aligncenter" width="501"][](http://i.stack.imgur.com/x81Al.png) Esquema librerias Spring Framework[/caption]

Las funciones de cada una de los módulos principales son las siguientes.

### ****Core Container****

* **core** y**beans** – Implementación básica de Spring, IoC e Inyección de Dependencias.
* **context**- Contexto de aplicación y otras características (i18n, propagación de eventos, etc…).
* **expression language** - Lenguaje especifico para manipulación de objetivos en tiempo de ejecución.

### ****Web****

* **web** – Integración web básica, inicialización del contenedor IoC.
* **web-servlet**  - Caracteristicas MVC.
* **web-portlet**- Soporte para portlets.
* **web-struts** – Integración con Struts.

### ****Data Access****

* **tx(transaction)** – Soporte básico para transacciones.
* **jdbc** – Soporte JDBC.
* **orm** – Integración ORM con Hibernate, JDO, Ibatis, etc…
* **oxm** - Integración con manejadores XML como JAXB, Castor, XMLBeans, etc…
* **jms** - Consumir y producir mensajes.

### ****Otros módulos****

* **test** - Testeo de la aplicación.
* **aop** – Integración con AspectJ.

SpringSource tambien nos provee de otra serie de librerías para el desarrollo de aplicaciones web u otras plataformas que vayan a hacer uso de estas aplicaciones, entre todas estas, a lo largo de este manual haremos uso de algunas como las siguientes.

* **Spring Android** – Comunicación entre aplicaciones Android y WebServices.
* **Spring Mobile** – Identificación de dispositivos.

En este manual vamos a centrarnos en Spring Web, que nos provee de una plataforma inmejorable para el desarrollo de aplicaciones web y webservices bajo el paradigma de programación Modelo Vista Controlador.

Se puede encontrar una lista de todos los proyectos de Spring aquí (<http://www.springsource.org/projects>).

Spring facilita el desarrollo de aplicaciones con J2EE (JSP, XML, Servlet, EJB, RMI, JMS,…) en base a las capas de abstracción, así como la integración con otros frameworks como Hibernate, JSF, Struts, iBatis. También maneja aplicaciones no Web como SWING o de consola. Todo esto se gestionara por un fichero xml donde se le dice cómo debe comportarse e integrar los componentes.



### Ventajas y desventajas

Las ventajas son claras ayudan en la construcción de aplicaciones acelerando los tiempos de desarrollo y reutilizando capas o componentes entre aplicaciones. Además permiten la incorporación de personal nuevo al proyecto con menor curva de aprendizaje.

Las desventajas nos encontramos de varios tipos, si el framework es desarrollado por una empresa particular, este se convierte en un proyecto por mérito propio y es necesario dedicar tiempo y recursos al mantenimiento del propio framework, si el framework es de dominio público las actualizaciones las realiza la comunidad de usuarios y suele ser más rápido y menos propenso a fallos. Otra desventaja es que las aplicaciones acaban siendo dependientes del framework sobre el que se construyen y la tercera es que a menudo, el personal incorporado en un proyecto en marcha no tiene un concepto claro de cómo funciona la aplicación en su totalidad, puesto que la definición de la arquitectura y servicios ya han sido diseñados y solamente queda adaptar nuevos servicios o mejorarlos.

### ¿Qué convierte a Springframework en especial?

Springframework fue diseñado para asumir las deficiencias de otros frameworks más pesados sobre la arquitectura J2EE.

J2EE ha realizado una serie de buenas prácticas y patrones para la definición de arquitecturas de software empresarial dentro del mundo java, sin embargo algunos de sus componentes han sido más un quebradero de cabeza que una solución y algunas de sus buenas prácticas se han convertido en antipatrones.

Frameworks como EJB trataron de minimizar el impacto del desarrollo asumido por los programadores haciendo que las tareas de bajo nivel, (seguridad, concurrencia, gestión de memoria, etc.) quedaran en mano de estos componentes y del servidor de aplicaciones que los alojase, así los desarrolladores sólo tenían que centrarse en desarrollar las tareas funcionales, como por ejemplo alta, bajas y modificaciones.

Sin embargo, en la mayoría de las versiones de EJBs su desarrollo era tremendamente caro en tiempo y esfuerzo para tareas pequeñas en gran medida y con arquitecturas poco efectivas como el paso de conexiones remotas entre componentes aunque estuvieran dentro de la misma máquina. Lo mismo ocurría con otros componentes dentro de la infraestructura Java como JNDI o JDBC por poner algunos ejemplos, donde el simple propósito de crear componentes de este tipo hacía que siempre se creasen de la misma manera y por lo tanto se recurría al denostado "copy&paste" para nuevos desarrollos, lo que a la larga hace que se incumplan buenas prácticas y por tanto sura el mantenimiento de las aplicaciones.

## El propósito de Spring

Como se ha comentado anteriormente Spring nació para resolver estas cuestiones a la hora de poder trabajar con componentes J2EE, facilitando el desarrollo y mejorando las arquitecturas, haciendo que no sea indispensable elegir los componentes adecuados desde el primer momento, gracias a su modelo "plugin", permite elegir los componentes posteriormente y no anclarse a una solución concreta.

1. Características

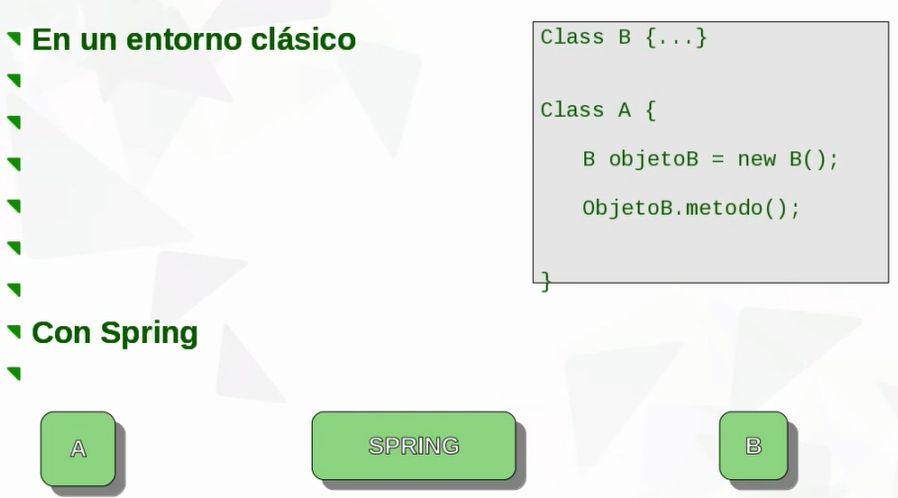
Características de SpringFramework

Podemos resumir las características en 3 bases:

### [Inversión de control](http://campus.dicotraining.com/mod/book/view.php?id=15)

Es la característica principal, el paradigma de [inversión de control](http://campus.dicotraining.com/mod/book/view.php?id=15) popularizado por [Martin Fowler](http://martinfowler.com/bliki/InversionOfControl.html), es también conocido como "Principio de Hollywood", por su famosa frase "no nos llame, nosotros le llamaremos", que resume claramente el propósito.

En un entorno orientado a objetos tradicional la llamada entre objetos se hace de manera directa, un objeto A invoca a un objeto B, sin embargo en este paradigma aparece un tercero (en este caso Springframework) que se encarga de realizar las llamadas entre objetos. Así cuando A quiere un objeto B, no lo llama directamente, sino que lo hace a través del intermediario que es Spring.



Las ventajas de este sistema es que tanto A como B no tiene porqué saber el uno del otro y se cargarían bajo demanda y por tanto no disponen de una **asociación fuerte**, lo que hace que exista un **bajo acoplamiento y mayor facilidad de reutilización y actualización.**

Ejemplo: **Proyectos**

**+ CosasDeBeans**

**+ HerenciaPropiedades**

**Herencia** (Proyecto HerenciaPropiedades)

Teniendo dos clases sin herencia se puede poner por configuración que exista una herencia omas bien herencia de propiedades,

Puedo crear un vean que no existe abstracto (AlternateBean.xml), heredara conjunto de propiedades. La factoría por debajo hace el setMessage2 y coge el valor aunque no exista la clase.

Si esta creado el bean como abstract no podrás crear objeto y por tanto nunca se podrá crear una instancia de el

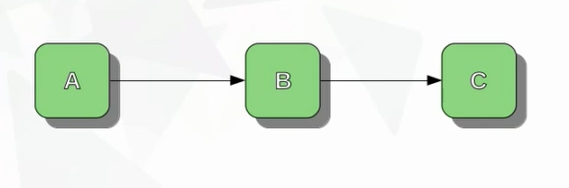
Ejemplo**: Proyecto HerenciaPropiedades**

**Ejercicio**: Generar un servicio que muestre los datos de una persona heredando el género de un bean abstracto que de la propiedad de masculino o femenino dependiendo del genero

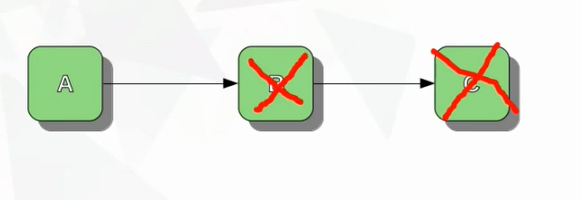
### [Inyección de dependencias](http://campus.dicotraining.com/mod/book/view.php?id=26)

Como extensión del principio anterior, disponemos de la [inyección de dependencias](http://campus.dicotraining.com/mod/book/view.php?id=26), así si el objeto A quiere al objeto B, pero con unos datos particulares y no como un objeto vacío, Spring puede "inyectar" valores en las propiedades públicas del objeto B, llenándolo de datos y así pasar el objeto B al objeto A tal y como este último lo necesita. Para realizar este trabajo se hace uso del [API de Reflexión](http://docs.oracle.com/javase/tutorial/reflect/index.html), y de una serie de buenas prácticas como es el desarrollo de clases bajo el modelo [POJO](http://es.wikipedia.org/wiki/Plain_Old_Java_Object) (Plain Old Java Object u objeto java de toda la vida).

Por ejemplo en la conexión a una base de datos



Si C esta caído, no tiene sentido que B exista para dar datos a A, por tanto B y C no se crearan.



Básicamente el modelo de desarrollo basado en POJOs debe cumplir una serie de reglas:

* Son clases Java simples, autodescriptivas
  + Esto implica no poner nombres extraños sólo conocidos por el programador
* Deben implementar el interfaz Serializable
* Su campos de clase debe ser privados
* Sus métodos de acceso debe ser públicos y hay de lectura y de escritura
  + los métodos de lectura comenzarán por get y los escritura por set
  + Deben utilizar nomenclatura [camelCase](http://es.wikipedia.org/wiki/CamelCase)
* Debe contener al menos un constructor vacío

A continuación se muestra un ejemplo:

import java.util.Serializable

public Class Usuario implements Serializable

private String nombre;

public String getNombre(){

this.nombre=nombre;

}

public void setNombre(String nuevoNombre){

this.nombre=nuevoNombre;

}

### Programación Orientada a Aspectos (AOP)

Basado en la intercepción de las cadenas de invocaciones entre objetos permitiendo manejar las aplicaciones desde otro punto de vista. La AOP es un nuevo paradigma de desarrollo que poco tiene que ver con el modelo de desarrollo orientado a objetos y que poco a poco empieza a tener más relevancia en las aplicaciones y sobretodo en Springframework.

Spring disponía hasta la versión 2.5 de su propio entorno de desarrollo AOP, pero a partir de la 3.0 ha asumido dentro de sus componentes al conocido [AspectJ](http://www.eclipse.org/aspectj/), aunque con algunas restricciones, pero hace uso de él de manera intensiva en la comunicación entre sus propios componentes del núcleo y también ofrece componentes basados en AOP para que podamos trabajar con este paradigma dentro de Spring haciendo más fácil y comprensible la manera de trabajar.

Dispondremos de un capítulo dedicado a **SpringAOP**

1. ¿Qué necesitamos para desarrollar para Spring?

### Un entorno de desarrollo

Necesitaremos un [IDE](http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado) para Java, cualquiera de los más populares servirá ([Eclipse](http://www.eclipse.org/), [IntelliJ](http://www.jetbrains.com/idea/), [NetBeans](http://www.netbeans.org/), [MyEclipseIDE](http://localhost/%7Edavid/moodle/http./www.myeclipseid.com)). Sin embargo, necesitaremos además plugins para Spring para cada uno de estos sistemas, además de otras herramientas que habrá que instalar y configurar, con el riesgo de equivocación de versiones que ello conlleva, por lo que la elección recomendada es usar la propia herramienta de la gente de Spring, llamada "[Spring Source Tool Suite](http://www.springsource.org/sts)". Se trata de un eclipse modificado con todos los plugins ya instalados y funcionando perfectamente.

### Java JDK instalado

Otra de las necesidades es disponer de un [JDK](http://www.oracle.com/technetwork/java/javase/downloads/index.html) instalado en el sistema, no vale con un JRE, es necesario un JDK versión 5 o superior, con recomendación en la 6, aunque la última (actualmente la 7) también es válida. Es requisito imprescindible incluso antes de poder instalar la herramienta STS, puesto que en un momento de la instalación nos pedirá la ruta del JDK.

**NOTA**: Puedes descargarlo desde aquí (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>) Asegúrate de elegir JDK y no JRE

### Maven

Maven es una herramienta bastante completa y desde la versión 3.x de Spring se ha convertido en imprescindible, ya que hasta la versión 2.5 de spring podíamos descargar un fichero comprimido donde venía la librería del core de spring, que se llamana spring.jar, y todas las depencias asociadas para que spring pudiera funcionar.

Pero desde la versión 3.x de spring esto ha cambiado, y es tan grande ahora spring que no es posible descargar sólo un fichero que contenga todo, así que han partido spring en módulos permitiendo así de ese modo que partes necesita mi aplicación.

Pero claro, eso tiene el inconveniene de que hay que conocer las dependencias asociadas a cada módulo y las versiones correctas a descargar para cada dependencia. Demasiado trabajo, aquí es donde entra en juego maven que resuelve correctamente las dependencias y sus versiones.

Maven en esencia es una herramienta de consola de comandos, pero la buena noticia es que hay un plugin para eclipse que funciona a las mil maravillas y permite trabajar en modo gráfico, pero hay que instalarlo en la versión adecuada. en las últimas versiones de eclipse es una tarea sencilla, pero aún mejor es si usas STS, porque ya está integrado y configurado.

### Una base de datos

En ciertas partes del curso necesitaremos mostrar qué capas de abstracción dispone Spring para facilitar el desarrollo contra bases de datos. En este caso por facilidad de administración e instalación haremos uso de [MySQL](http://www.mysql.com/downloads/mysql/) y su [conector](http://www.mysql.com/downloads/connector/j/).

1. ¿Qué es Maven?

Es una herramienta que cumple tres propósitos:

1. Es un gestor de proyectos
2. Es un organizador de dependencias
3. Es un creador de proyectos

### Maven como gestor de proyectos

Todos los proyectos sufren las típicas tareas:

**Descargar código de** CVS-> **Compilar** ->**Pasar pruebas**  -> **Documentar** ->**Integrar con otros módulos** ->**Publicar en entorno de testing/producción**

Algunas de estas tareas tienen subtareas, por ejemplo, para pasar adecuadamente las pruebas es posible que haya que:

* Integrarse con otros componentes desarrollados por terceros
* Arrancar bases de datos / servidores de aplicaciones
* Limpiar datos en la base de datos o crear datos maestros en cada prueba
* Reportar exito/error de las pruebas
* Notificar por email a los desarrolladores de los errores
* Publicar informes en una Intranet, wiki u otro sistema de comunicación del equipo de desarrollo

Si estamos documentando el proyecto es posible que nos interese crear distintos tipos deinformes, no sólo el clásico javadoc, si no pruebas de rendimiento, cobertura de test, chequeos de código bien formado, etc.

Lo mismo ocurriría si tenemos distintos entornos creados con software diferente para testing que para producción, habría que cambiar configuraciones.

Maven puede realizar todas estas tareas por sí mismo y se pueden automatizar para por ejemplo lanzarlo diariamente en modo batch por las noches y así obtener los resultados al día siguiente.

### Maven como organizador de dependencias

Es común que al participar en distintos proyectos se creen distintas configuraciones y con un montón de librerías de terceros con versiones distintas, lo que hace que se convierta el disco duro en un conjunto de liberías o carpetas desperdigadas. Esto tiene consecuencias ya que a veces nos encontramos con conflictos entre versiones de librerías iguales o similares.

Por otro lado, al no estandarizar los artefactos necesarios para desarrollar en el proyecto, otro error común suele ser que cada desarrollador puede elegir la versión que mejor le venga y en local todo funciona bien, pero cuando hay que integrarse con los módulos de los compañeros nos encontramos con incompatibilidades.

Maven resuelve esto creado un repositorio de librerías en el $HOME del usuario por defecto, concretamente crea una carpeta .m2 en el directorio principal del usuario y debajo una carpeta repository, donde almacenará todas las dependencias (librerías) de cada proyecto sin que interfieran entre ellas.

Si un proyecto necesita una dependencia y el programador no la tiene en su repositorio, maven automáticamente se la descargará de Internet y la introducirá en su repositorio.

También es posible crear un repositorio a nivel de empresa, o grupo de trabajo en una Intranet, para que los desarrolladores tiren de ese repositorio en lugar del central de Internet

### Maven como creador de proyectos

Otro de los inconvenientes posibles al trabajar en proyectos es que cada proyecto es distinto del anterior, por lo que es difícil a veces conocer la estructura del proyecto y la nomenclatura del mismo. Eso no ocurre con Maven ya que dispone de una estructura común para todos los proyectos creados con Maven:

 Esta estructura es la que se detalla a continuación:

/raíz del proyecto

 /----src/

          main/

                 java/

                 resources/

        test/

                 java/

                 resources/

Debajo de la raíz del proyecto se encuentra un directorio src donde se almacenará todo el código fuente, concretamente en la rama src/main/java, mientras que en src/main/resources se destina a aquellos activos que no son .java y no necesitan ser compilados en .class, pero que es necesario que estén dentro del CLASSPATH de la aplicación tales como archivos de propiedades, de configuración, etc.

Por otro lado está la rama src/test/java y src/test/resources con el mismo propósito que el anterior pero únicamente destinado al desarrollo de pruebas. Recordamos que una de las fases de maven es la de testing (pruebas) indicado en el primer punto. Si alguna fase falla no puedecontinuar con la siguiente fase. Pensemos que si fallan las pruebas con la base de datos, no tendría sentido lanzar la aplicación contra esa base de datos, por lo tanto se detendrá y no continuará hasta que se soluciones. Por supuesto, esto se puede personalizar.

### Un fichero para dominarlos a todos

Toda la configuración de estructura de proyecto, dependencias, y fases de trabajo se especifican en un fichero de configuración de Maven que se llama **pom.xml.**Se trata de un fichero en formato XML y cumple una estructura estándar. Este fichero dicta cómo se debe comportar maven y es compartido por todos los miembros del equipo, por lo que una configuración creada en un pom.xml se puede enviar a otro miembro del equipo e inmediatamente tendrá su proyecto correctamente configurado.

Teniendo esta estructura estándar de proyecto, nos encontramos con que todos los desarrolladores disponen de la misma forma de trabajar, si a ello unimos que no hay posibilidad de equivocarse en las dependencias, reducimos los posibles errores y damos una estructura común de trabajo a los miembros del equipo.