

Introducción a Spring

Antonio Espín Herranz

Introducción

- **Spring** es un framework (marco de trabajo) de **código abierto** creado por **Rob Jonhson (empresa Interface21 → SpringSource)**.
- Fue creado para tratar las dificultades del desarrollo de aplicaciones empresariales.
- Spring se compone de un **contenedor de beans**.
- A parte de una gran cantidad de módulos que nos proporcionan distintas funcionalidades.
- Es un framework ligero que trabaja con el patrón **IoC** (Inversión of Control).

Características

- **Contenedor:**
 - Contiene y gestiona el ciclo de vida de los objetos de nuestra aplicación. Los crea, los configura y los destruye.
- **Marco de Trabajo:**
 - En Spring los objetos de la aplicación se define de forma **declarativa**, normalmente en un **archivo XML / anotaciones**.
- **Ligero:**
 - Todo el framework se puede distribuir de un archivo jar que ocupa unas 5,5 Mb.
- **Inyección de dependencias:**
 - Fomenta el **acoplamiento débil** mediante una técnica conocida como inyección de dependencias (DI).
- **Orientado a aspectos:**
 - Spring tiene un amplio soporte para la **programación orientada a aspectos (AOP)**.

Características

- En el caso de **Spring** está pensado para **todas las capas de la aplicación** a diferencia de otros frameWorks como Struts2 / JSF que está pensando para la capa de presentación.
- Lo normal es **integrar Spring con otras tecnologías** / frameWorks como: Struts2 o JSF / Hibernate.
- En el caso de **Spring** se le deja la gestión de **todos los beans de la aplicación (lógica de negocio)**.

Posibles Arquitecturas

- **JSF + Spring + Hibernate:**
 - JSF: Capa de presentación.
 - Spring: Servicios, objetos de negocio.
 - Hibernate: Acceso a datos. ORM: Mapeo de tablas de la BD en objetos java.
- **Struts + Spring + Hibernate:**
 - Struts: En la capa de presentación, sustituye a JSF.
- **Spring MVC + Hibernate / Templates JDBC Spring:**
 - Spring MVC soporte para montar una aplicación en 3 capas y la parte de acceso a datos con Hibernate o las plantillas JDBC.

Características

- **Spring se puede utilizar en todo tipo de aplicaciones**, no necesariamente tiene que ser una aplicación Web, ni tampoco es necesario utilizar todos los módulos de Spring en una aplicación.
- Puede utilizarse dentro de una aplicación **Swing** con gestión de Base de datos.
- Incluso en **dispositivos** de pocos recursos como un **móvil**.

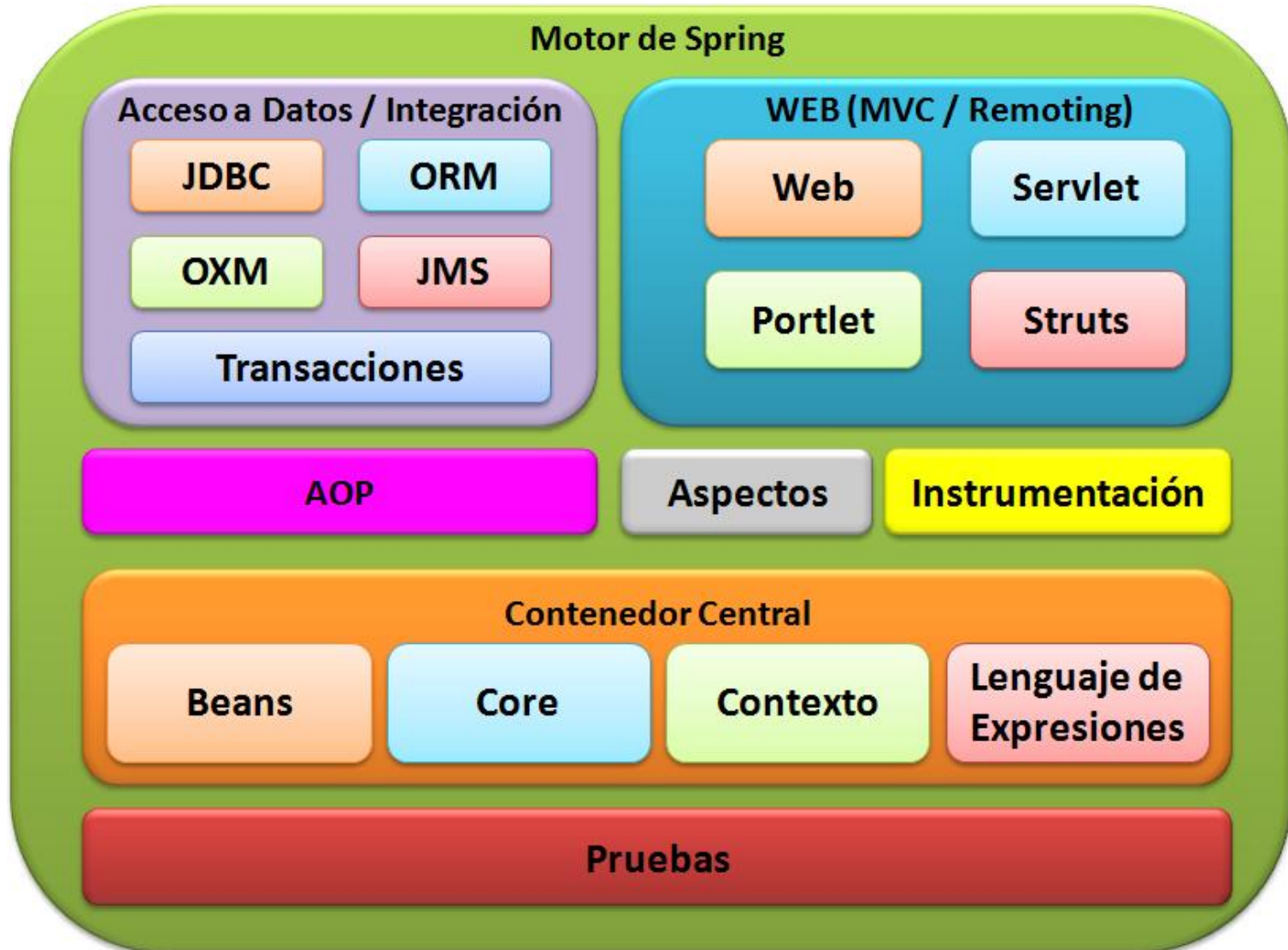
Estrategias

- Surge como **sustitución** de otras tecnologías más pesadas como **EJBs**.
 - **Desarrollo ligero y poco invasivo**, con objetos Java simples (**POJO**: Plain Old Java Object).
 - **Acoplamiento débil** mediante la inyección de dependencias y la orientación de interfaz.
 - **Programación declarativa** mediante aspectos y convenciones comunes.
 - **Reducción del código reutilizable** mediante aspectos y plantillas.

Módulos de Spring

- Spring está dividido en alrededor de **20 módulos** y colocados en los siguientes **grupos**:
 - **Contenedor Central** (Core Container)
 - **Acceso a Datos** / Integración
 - **WEB**
 - **AOP** (Programación Orientada a Aspectos)
 - **Pruebas**

Módulos de Spring



Contenedor del núcleo de Spring

- El elemento central de **Spring** es un **contenedor que gestiona** la forma en que los bean de una aplicación de Spring se crean, se configuran y administran.
- Hay **varias implementaciones del contexto** de aplicaciones de Spring.
- A parte de la fábrica y los contextos de aplicación **dispone de servicios empresariales** como correo electrónico, integración con EJB, JNDI, ...

Módulo AOP de Spring

- Este módulo es la base para el desarrollo de los aspectos básicos de una app. de Spring.
- **Proporciona acoplamiento débil.**
- **Los temas relacionados con seguridad, transacciones se desacoplan de los objetos donde van ubicados.**
- **Se aplican de una forma declarativa.**

Acceso de datos e integración

- Cuando trabajamos hay una serie de pasos que son repetitivos:
 - abrir conexión,
 - crear sentencia,
 - procesar el resultado,
 - cerrar conexión.
- Mediante este módulo **Spring permite extraer el código repetitivo**, trabaja con plantillas JDBC.
- También proporciona conexión a otros frameworks como Hibernate.

Web y acceso remoto

- Tiene soporte para el **patrón Modelo-Vista-Controlador** (la interfaz de usuario se separa de la lógica de la aplicación).
- Incluye **dos formatos**:
 - Uno basado en **Servlet** para aplicaciones Web convencionales.
 - Y otro se basa en **Portlet**.
- Además de soporte para **RMI** y **JAX-WS**.

Pruebas

- Incluye un módulo para hacer pruebas de la aplicaciones.
- Dentro de este módulo hay **una serie de implementaciones de objetos de prueba** para escribir unidades de prueba para el código.

El catálogo de Spring

- **Se dispone de una gran cantidad de proyectos asociados al marco de Spring, como ejemplo:**
 - **Spring Web Flow:** Guía al usuario en la creación de aplicaciones basadas en el MVC.
 - **Spring Web Services:** implementación de WS.
 - **Spring Security:** La seguridad se implementa mediante AOP mediante un mecanismo de programación declarativa.
 - **Spring Mobile:** permite el desarrollo de aplicaciones Web móviles.
 - **Spring.NET:** Sobre la plataforma .NET.

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**:
 - 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 (bueno por un lado, por otro descentralización).

Ventajas y desventajas

- 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.

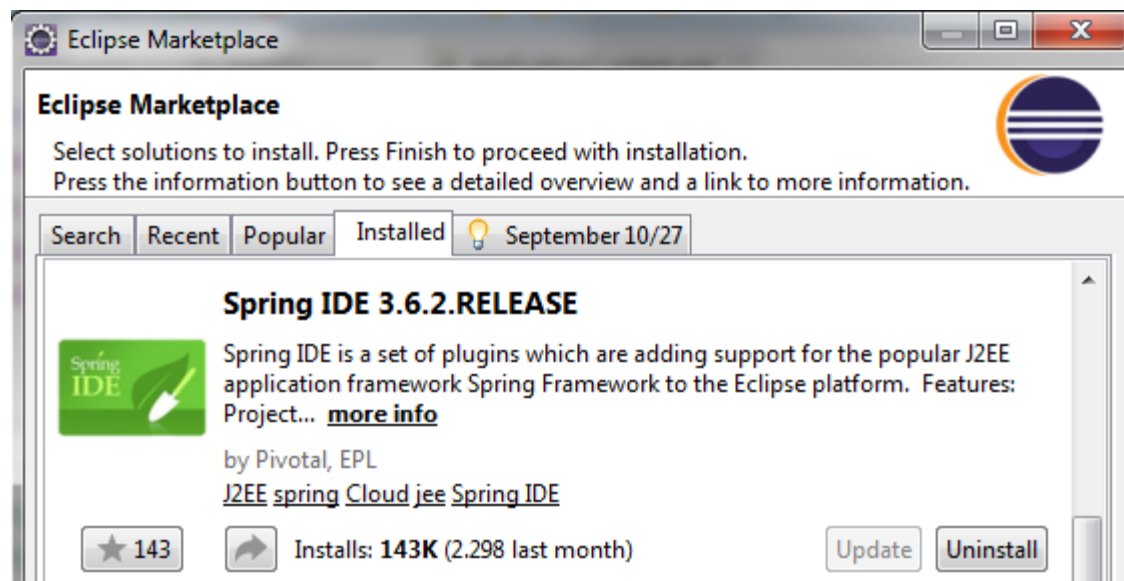
Herramientas

- **Instalar:**
 - Framework de **Spring 3.1.2**
 - Eclipse con el plugin de Spring.
 - Apache Commons (necesario para Spring)
- **Base de datos (MySQL / Oracle):**
 - MySQL
 - MySQLAdministrator.
 - MySQLQuery.

Instalar plugin Eclipse

- Desde eclipse Marketplace:
- Instalar Spring en Eclipse (**opción Marketplace**)

<http://www.mkyong.com/spring/how-to-install-spring-ide-in-eclipse/>



También se puede trabajar con **Spring Tool Suite. STS**

<https://spring.io/tools/sts/all>

Instalar: Spring

- **Descargar siempre las versiones GA:** (la versión GA - estable)

Página principal de Spring: <http://spring.io/>

- Descomprimir el fichero de Spring:
 - **spring-framework-3.1.2.RELEASE-with-docs.zip.**
 - En una carpeta en C:\
- En la carpeta **dist** tenemos los **jar** que tenemos que utilizar para agregar a nuestros proyectos.
- También se da **la opción de incluir la dependencia** del fichero de configuración de **Maven (pom.xml)**.

Instalar: **SpringSource Tool Suite**

- Dos posibilidades:
Existe una herramienta: **SpringSource Tool Suite**.
<http://www.springsource.org/spring-tool-suite-download>
- Integra **eclipse con los plugins de Spring** para el desarrollo de aplicaciones.
- Dispone de un instalador.
- **Mejor opción** (suele ir más rápido).
 - Eclipse con el plugin de STS. Se instala desde el eclipse Marketplace → **Menú Help**.

Instalar: **commons-logging-1.1.1**

- Descarga Apache Commons:
- Descomprimir en una carpeta en C:\
commons-logging-1.1.1-bin.zip
- A los proyectos de Spring es necesario **agregar**
el jar: **commons-loggin-api-1.1.1.jar**

Documentación

- La **documentación** de Spring la podemos encontrar en la siguiente URL:

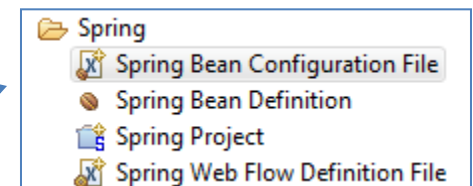
<http://docs.spring.io/spring/docs/3.1.2.RELEASE/spring-framework-reference/>

- **API:**

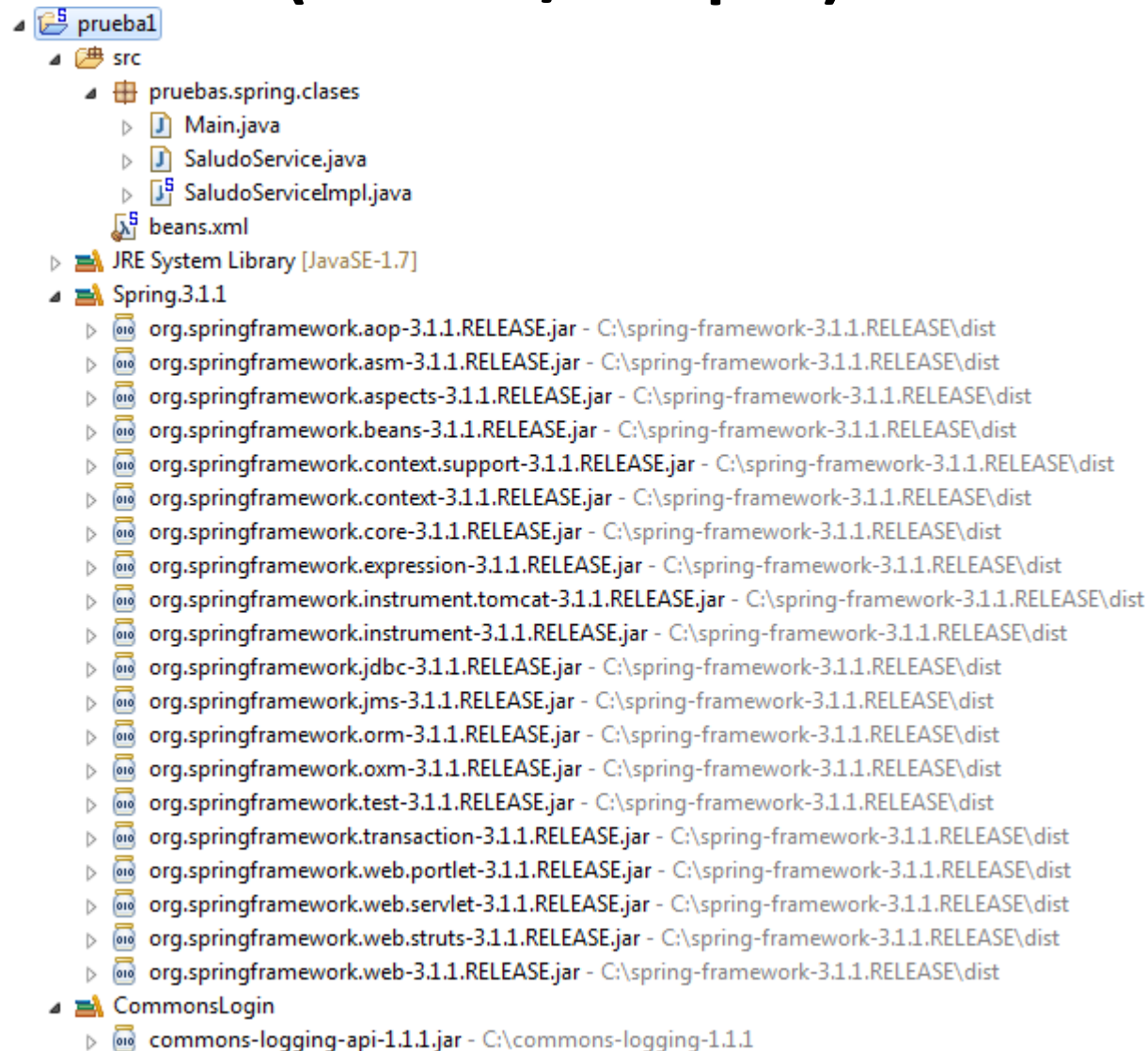
<http://docs.spring.io/spring/docs/3.1.2.RELEASE/javadoc-api/>

Ejemplo paso a paso

- Queremos crear un ejemplo en Spring que emita un saludo por pantalla.
- Creamos un proyecto java.
 - Necesitamos agregar los jar de Spring y commons-login-api-1.1.1.jar
 - **Se recomienda crear una librería de usuario** con los jar de Spring, para agregarla a los proyectos que vayamos haciendo.
- A nivel de java:
 - Creamos un interface que declara el método saludar().
 - Una clase que representará el saludo y que implementa el interface anterior.
 - Y una clase principal para crear el saludo.
- En el proyecto vamos a crear un fichero XML:
 - En el cual mediante programación declarativa vamos a declarar nuestro bean.



Ejemplo paso a paso: Estructura del proyecto (en STS / Eclipse)



Ejemplo paso a paso

- **El fichero: beans.xml**

```
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="http://www.springframework.org/schema/beans  
http://www.springframework.org/schema/beans/spring-beans.xsd">
```

```
<bean id="saludoService"  
class="pruebas.spring.clases.SaludoServiceImpl">  
<property name="saludo" value="Hola desde Spring"></property>  
</bean>  
</beans>
```

Inyecta la propiedad **saludo**, a través del método setSaludo()

- El bean se declara dentro de este fichero, se le da un identificador y la ruta completa a la clase java que lo representa.
- Desde la declaración se puede dar valor a las propiedades del bean.

Ejemplo paso a paso

- **El interface: SaludoService.java**

```
package pruebas.spring.clases;
```

```
public interface SaludoService {  
    public void saludar();  
}
```

Ejemplo paso a paso

- **Implementación del interface: SaludoServiceImpl.java:**

```
package pruebas.spring.clases;
```

```
public class SaludoServiceImpl implements SaludoService {  
    private String saludo;  
    public SaludoServiceImpl(){}  
    public SaludoServiceImpl(String saludo){ this.saludo = saludo; }  
    public void saludar() { System.out.println(saludo); }  
    public String getSaludo() { return saludo; }  
    public void setSaludo(String saludo) { this.saludo = saludo; }  
}
```

- La clase proporciona un constructor por defecto y otro con argumentos.
- Declara la propiedad saludo, y los métodos set / get.

Ejemplo paso a paso

- La clase principal: Main.java

```
package pruebas.spring.clases;  
  
import org.springframework.beans.factory.BeanFactory;  
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        BeanFactory factory = new ClassPathXmlApplicationContext("beans.xml");
```

```
        SaludoService saludo = (SaludoService) factory.getBean("saludoService");
```

```
        saludo.saludar();
```

```
    }
```

```
}
```

- La clase captura el bean de la factoría que se inicializa con el fichero XML declarado con anterioridad.

Maven

- Es una herramienta que cumple tres propósitos:
 - Es un gestor de proyectos
 - Es un organizador de dependencias
 - 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.**

Maven

- 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 éxito/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
- 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

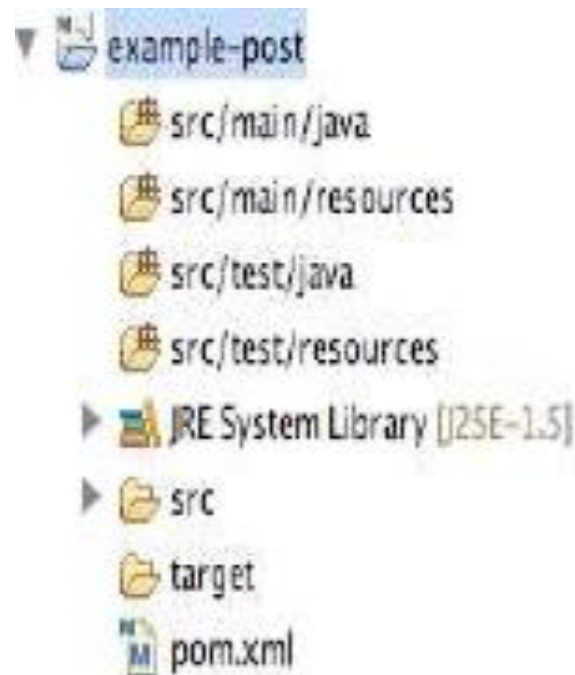
- ***Maven como organizador de dependencias***
- Es común que al participar en distintos proyectos se creen distintas configuraciones y con un montón de librerías de terceros con versiones distintas, lo que hace que se convierta el disco duro en un conjunto de librerías o carpetas desperdigadas. Esto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

Maven como organizador de dependencias

- Maven resuelve esto creando 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.

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:



Maven como creador de proyectos

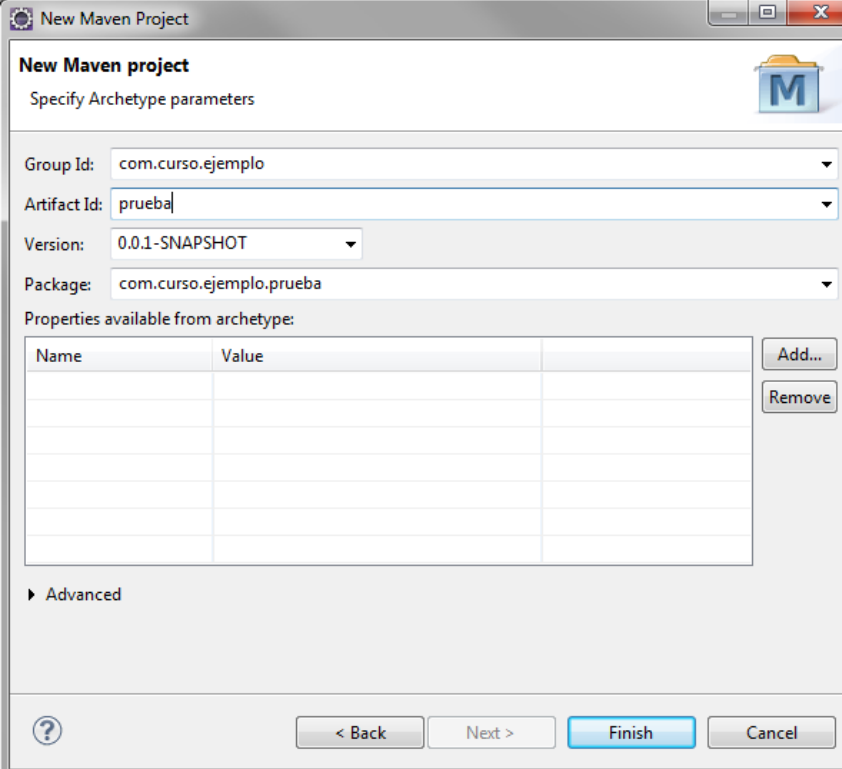
- **Maven tiene cuatro carpetas fuente por defecto:**
 - **src/main/java** : donde guardaremos nuestras clases java fuente. Debajo de esta carpeta situaremos nuestras clases en distintos paquetes.
 - **src/main/resources** : aquí almacenaremos los recursos (ficheros xml, ficheros de propiedades, imágenes, ...) que pueda necesitar las clases java de nuestro proyecto. Igualmente aquí tienen que ir los ficheros de configuración de Spring o Hibernate por ejemplo.
 - **src/test/java** : en dicha carpeta se guardan las clases de test que se encargarán de probar el correcto funcionamiento de nuestra aplicación. Aquí por ejemplo podemos guardar nuestros test unitarios de JUnit.
 - **src/test/resources** : en esta carpeta guardamos los recursos que usan los recursos.

Fichero configuración en Maven

- 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.

Ejemplo proyecto con Maven

- En eclipse, seleccionar Maven Project. Next.



The screenshot shows the 'New Maven Project' dialog box in Eclipse. The title bar says 'New Maven Project'. Inside, the subtitle is 'New Maven project' and 'Specify Archetype parameters'. There is a Maven logo icon in the top right. The form contains the following fields:

- Group Id:
- Artifact Id:
- Version:
- Package:

Below these is a section 'Properties available from archetype:' containing a table with columns 'Name' and 'Value'. To the right of the table are 'Add...' and 'Remove' buttons.

At the bottom, there is an 'Advanced' section with a right-pointing arrow. The footer contains a help icon, '< Back', 'Next >', 'Finish' (highlighted in blue), and 'Cancel' buttons.

Ejemplo proyecto con Maven

- Editar el fichero: pom.xml (seleccionar pestaña).
- **Dependencias:**

```
<dependencies>
```

```
    <dependency> <groupId>org.springframework</groupId>  
        <artifactId>spring-context</artifactId>  
        <version>3.1.2.RELEASE</version>  
    </dependency>
```

```
</dependencies>
```

Al copiar la dependencia grabar, y eclipse descarga la librerías necesarias.