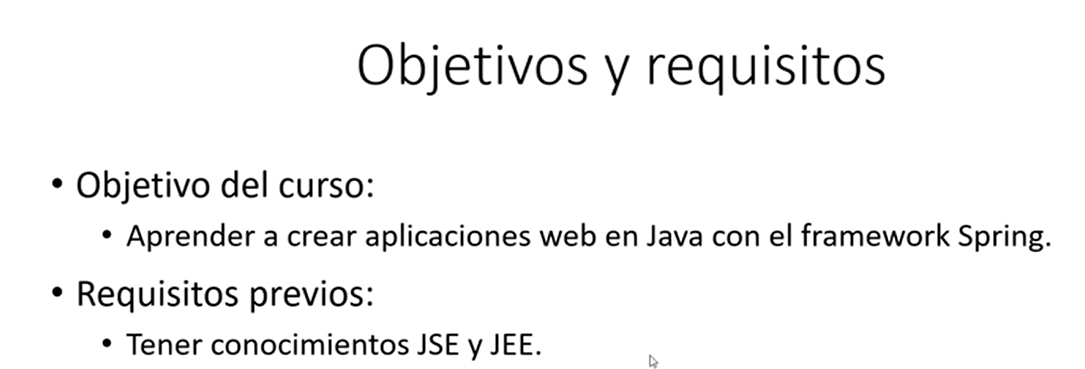
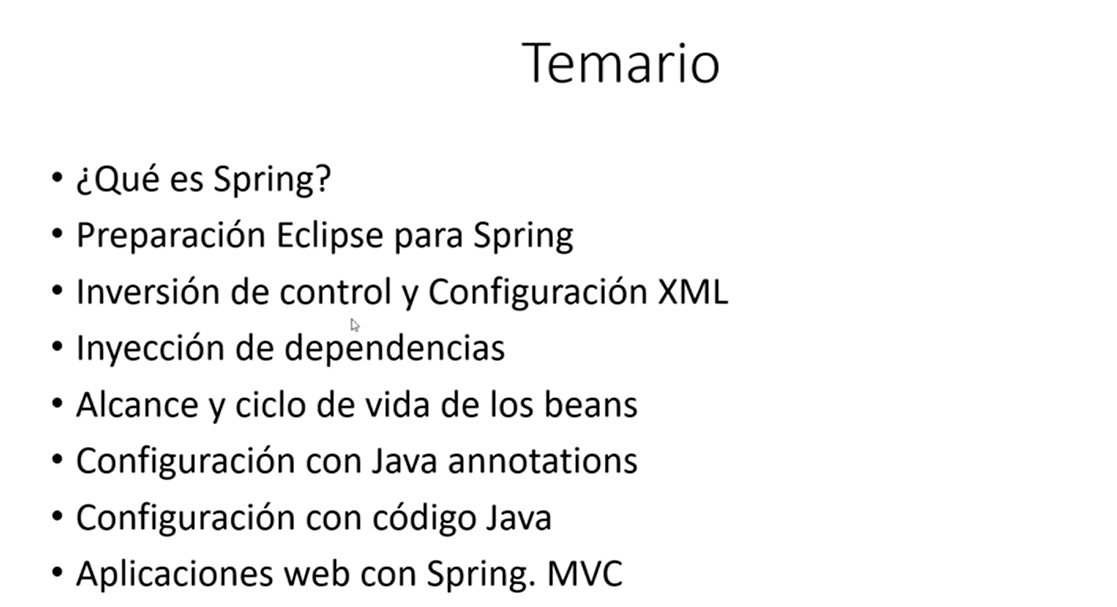
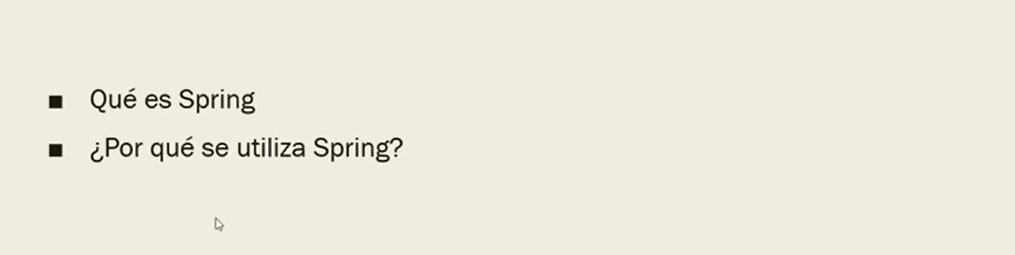
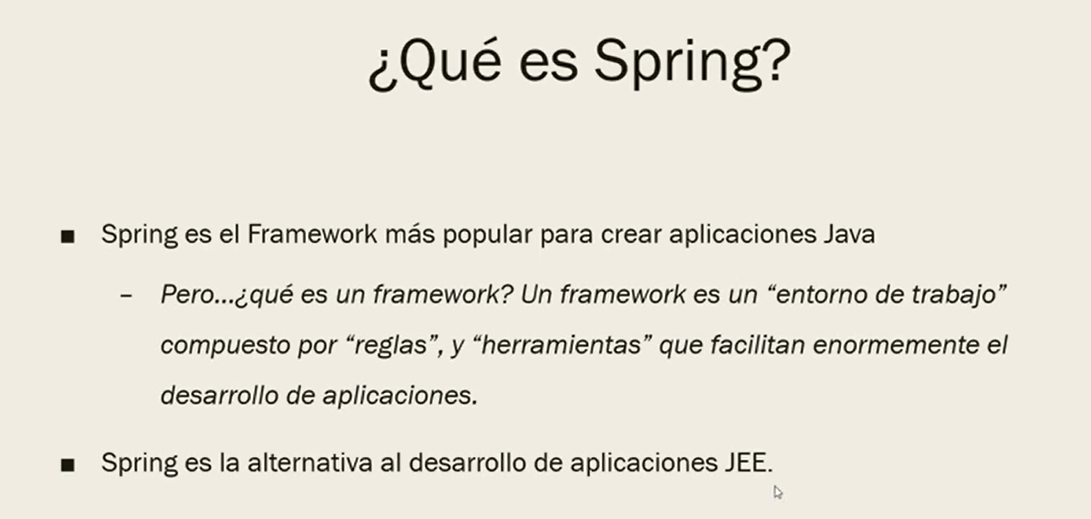
Curso Spring Framework

Presentación Introductoria

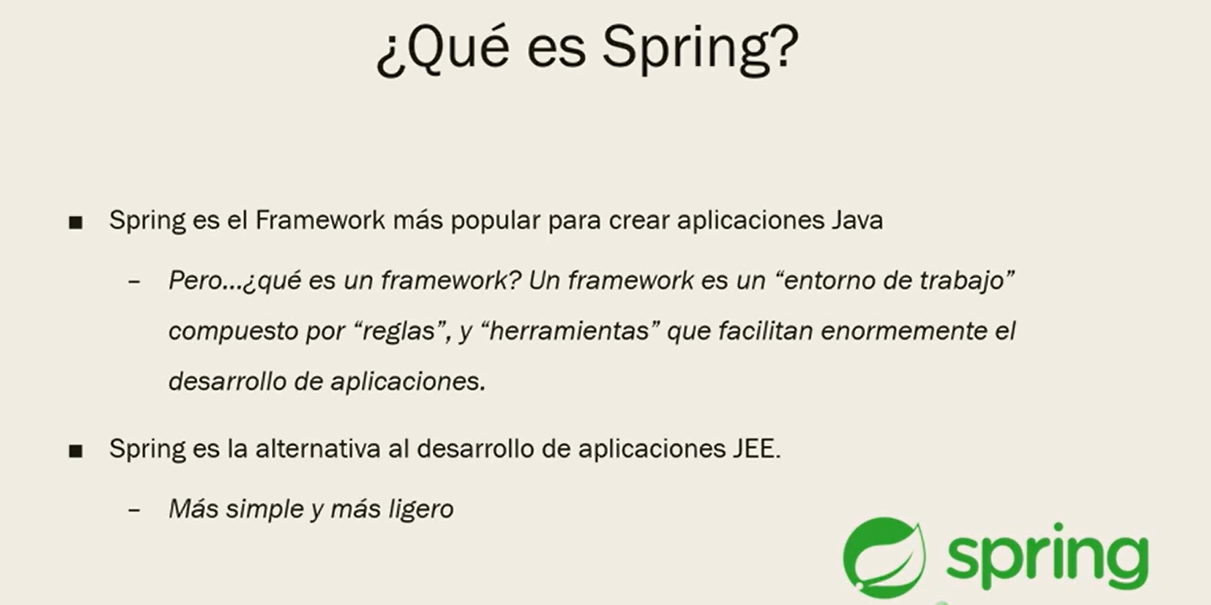


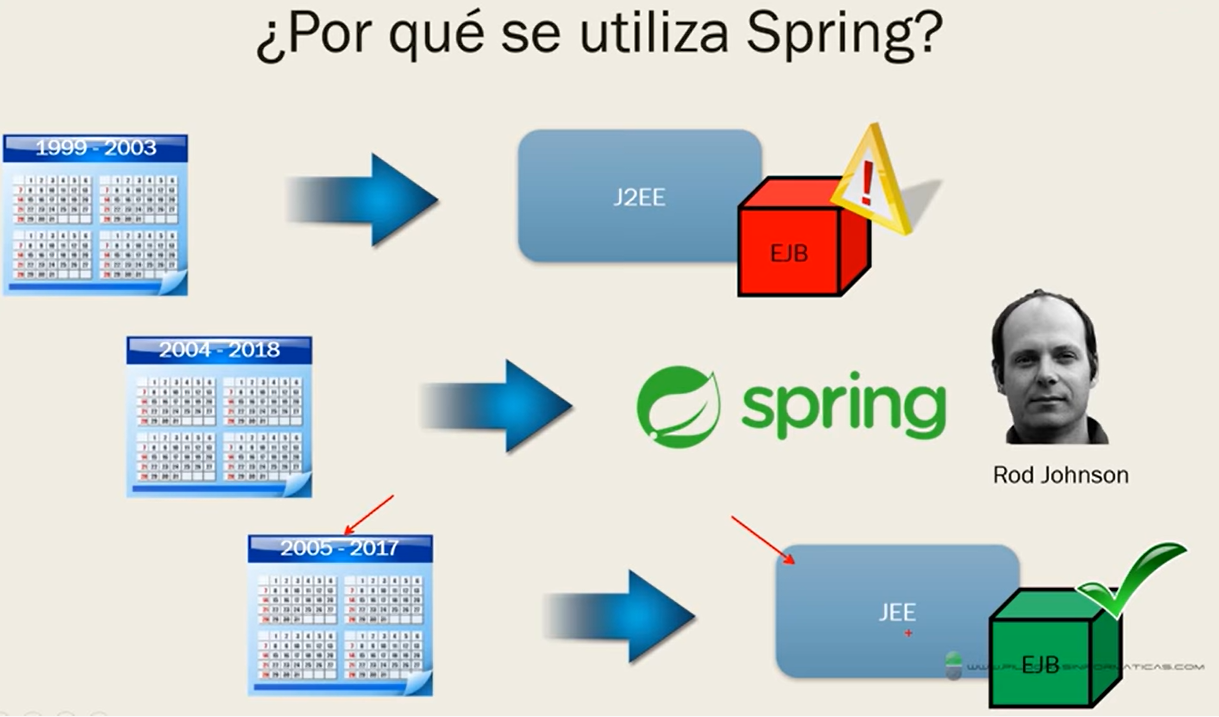






* En Spring las herramientas son bibliotecas de clases



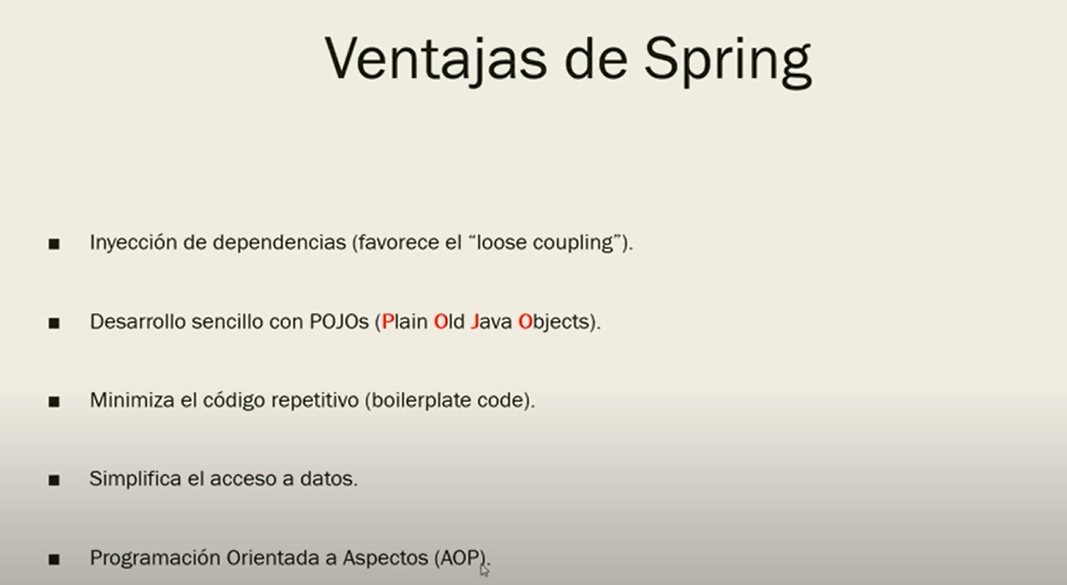


Los EJB eran complejos, lentos y consumían muchos recursos, por ese motivo surge Spring de la mano de Rob Johnson, aquí los programadores en masa migraron a Spring , luego los creadores de java vieron Spring y comenzaron a copiarse lanzando la versión JEE y empezaron a tener muchas cosas en común pero JEE llego tarde al mercado ya todos utilizaban Spring y habían migrado en masa ,hoy en día no existe tanta diferencia entre usar Spring y las últimas versiones de JEE

Si quieren manejar bien un framework primero deben aprender JE ,

El Core de Spring – principales ventajas





Ventajas de Spring

1) inyección de dependencias (favorece el “loose coupling”) , es cuando modificas una clase y debes introducir esa modificación en otras clases , permite la separación de los módulos de una App java para no repetir la modificación

2) desarrollo sencillo con pojos (plain old Java objects) ,por ej objeto persona (clase simple que no hereda ni extiende de nada )

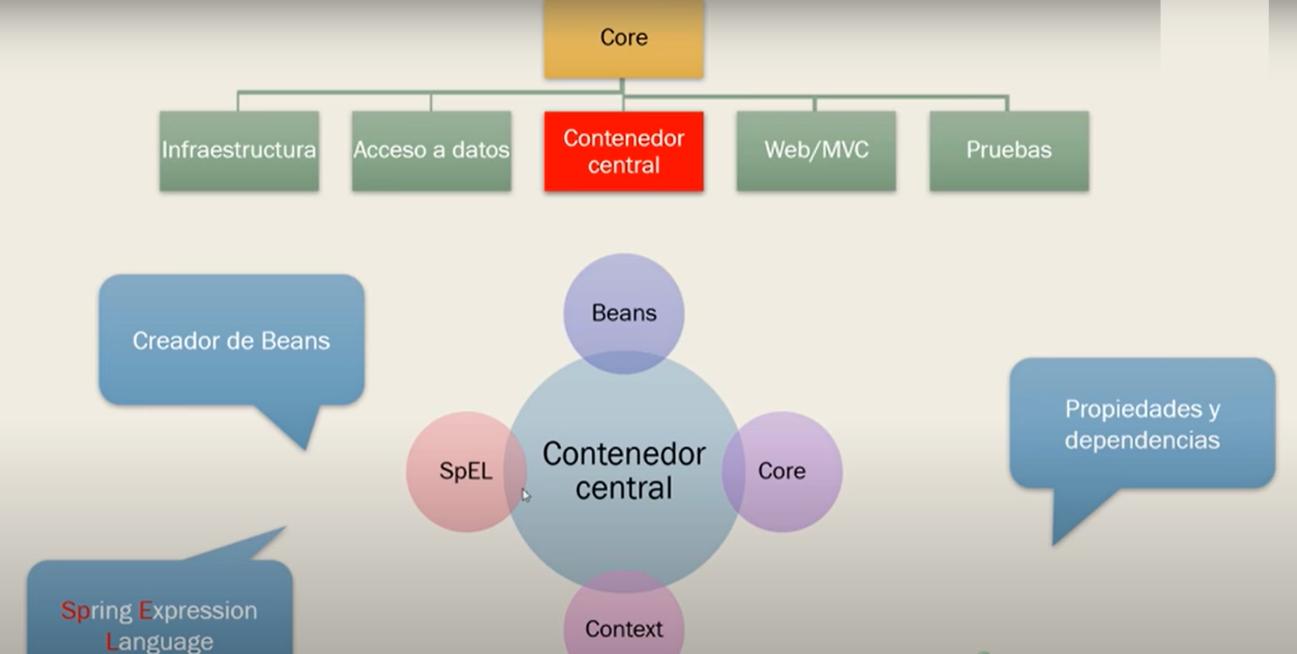
3) minimiza el código repetitivo que a veces copias y pegas en todos lados (boilerplate code) para que tu app funcione

4) simplifica el acceso a datos con el módulo de jdbc de spring para acceder a datos , en spring escribimos mucho menos codigo

5) AOP programación orientada a aspectos ,permite una modularizacion muy alta de nuestra app para hacer una separación muy clara de las diferentes tareas que debe hacer cada clase de nuestro programa

Módulos del core de spring

Se divide en 5 grandes módulos (infraestructura , accesos a datos , contenedor central , web/mvc , pruebas)



El principal es el central que se sub divide en 4 modulos mas , el core de spring esta divido en muchos modulos ahora solo queresmo tener una ideal global de todo

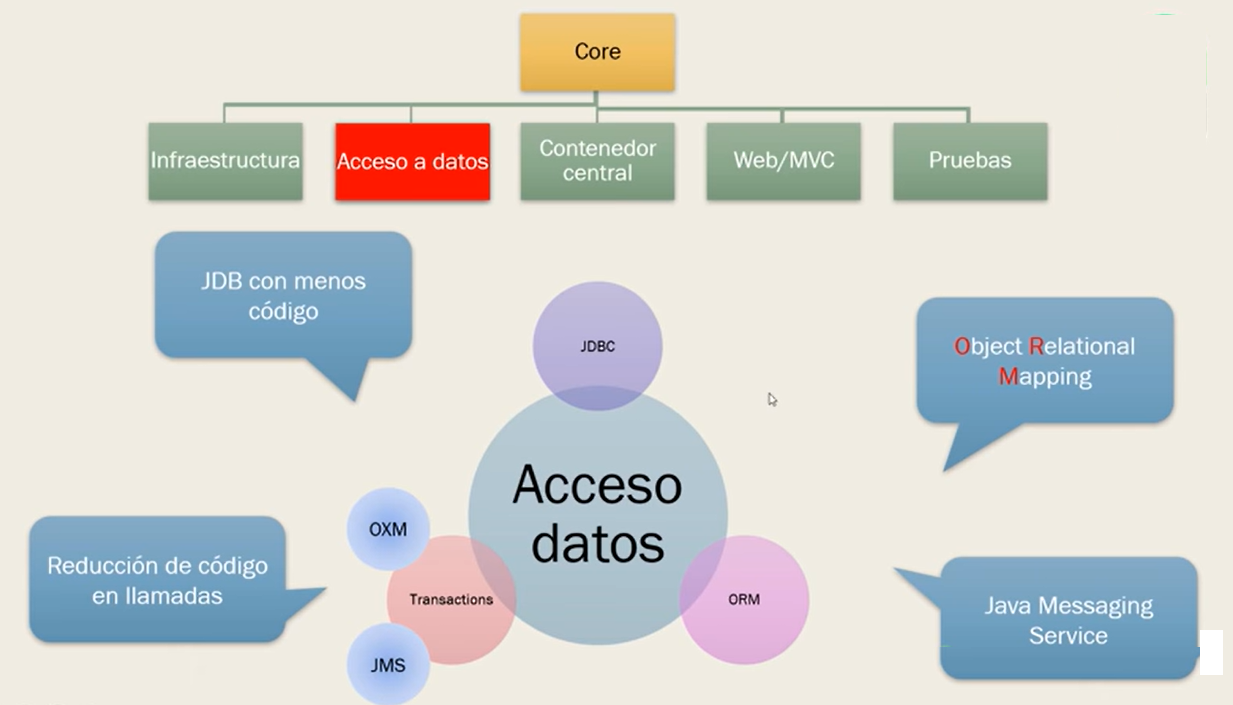
El contenedor central , crea beas , lee archivos config , maneja propiedad y dependencias , uso del context como almacen de los beans en memoria , el SPL (Spring Expression Lenguage) nos permite entre otras cosas manipular objetos en tiempo de ejecución

Infraestructura



Aca hacemos referencia a AOP , este módulo maneja todo lo que tiene que ver con logueo , transaciones , seguridad de la app,

Módulo de Acceso a datos



Este módulo forma parte del acceso a datos de spring y contiene 3 sub módulos mas

En módulo de transacción tiene dos sub modulos mas (oxm y jms ) que tiene mucha información y es muy importante

El módulo JDBS no permite acceder a la DB con clases auxiliarles con muchísimo menos código,

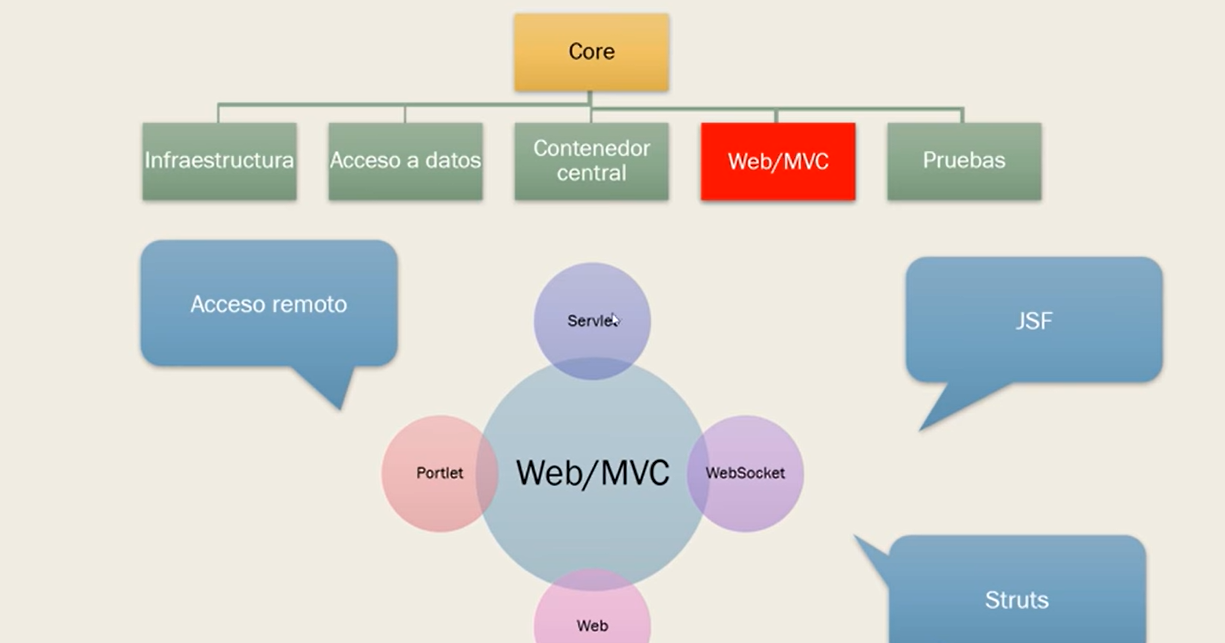
El módulo ORM no permite acceder a datos con un enfoque orientado a objetos , podemos tratas las bases de datos como objetos con sus propiedades y métodos , y una alta integración con hibernate

El módulo JMS permite manejar los mensajes java

El módulo de transacción no permite reducir mucho el codigo a la hora de de hacer llamadas a la DB

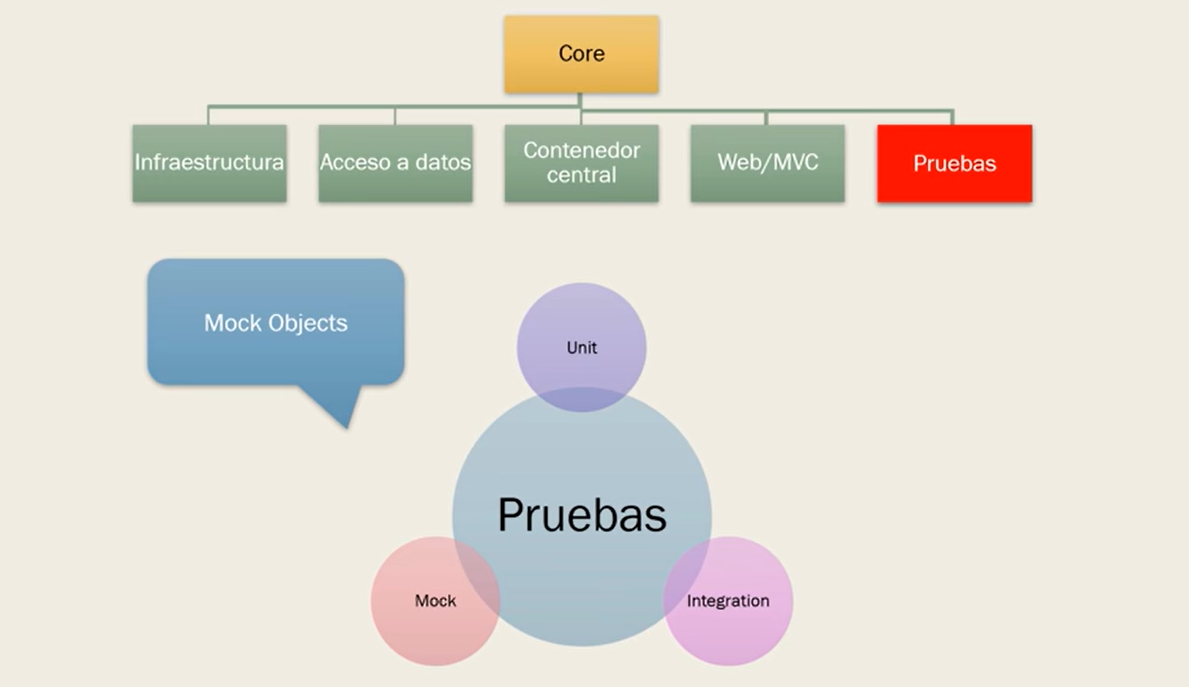
Módulo Web MVC

Estos 4 sub módulos nos permiten el acceso web remoto, la programación distribuida y integración con JSF , Strutc ,etc .



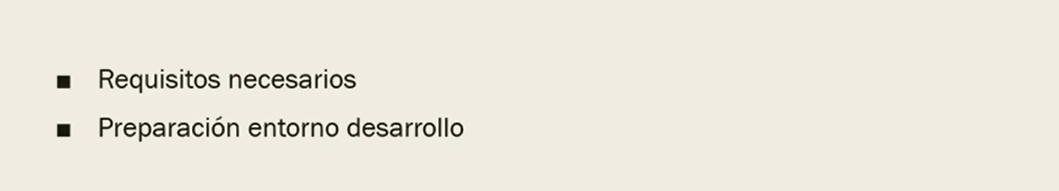
Módulo de Pruebas

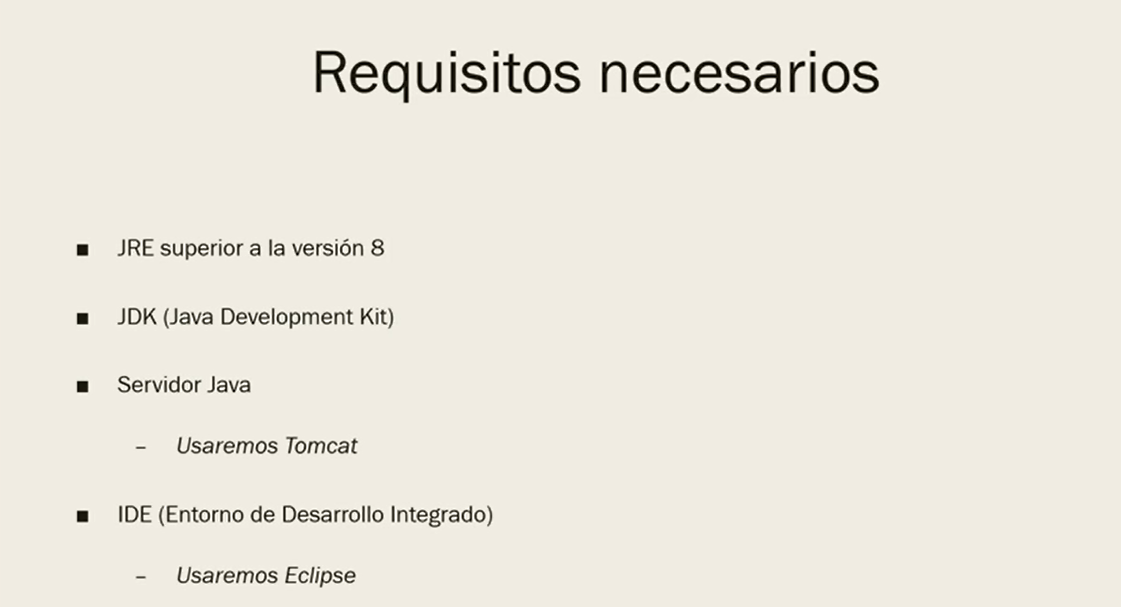
Este modulo de Test no permite realizar test durante el desarrollo de nuestra app , nos trae los objetos mock que podemos crear para hacer pruebas



Instalación Tomcat 9 y Entorno de desarrollo

IDE JEE

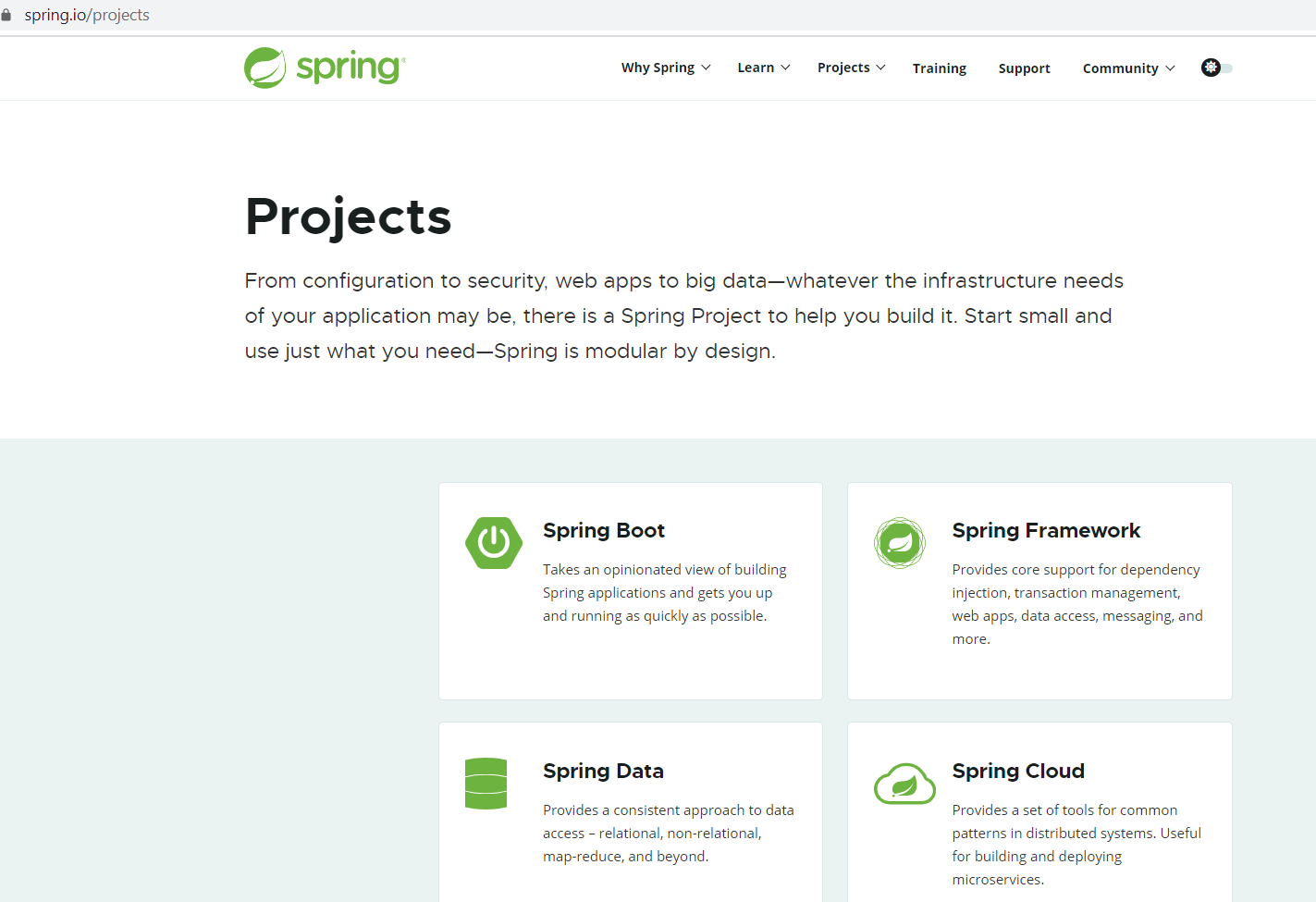




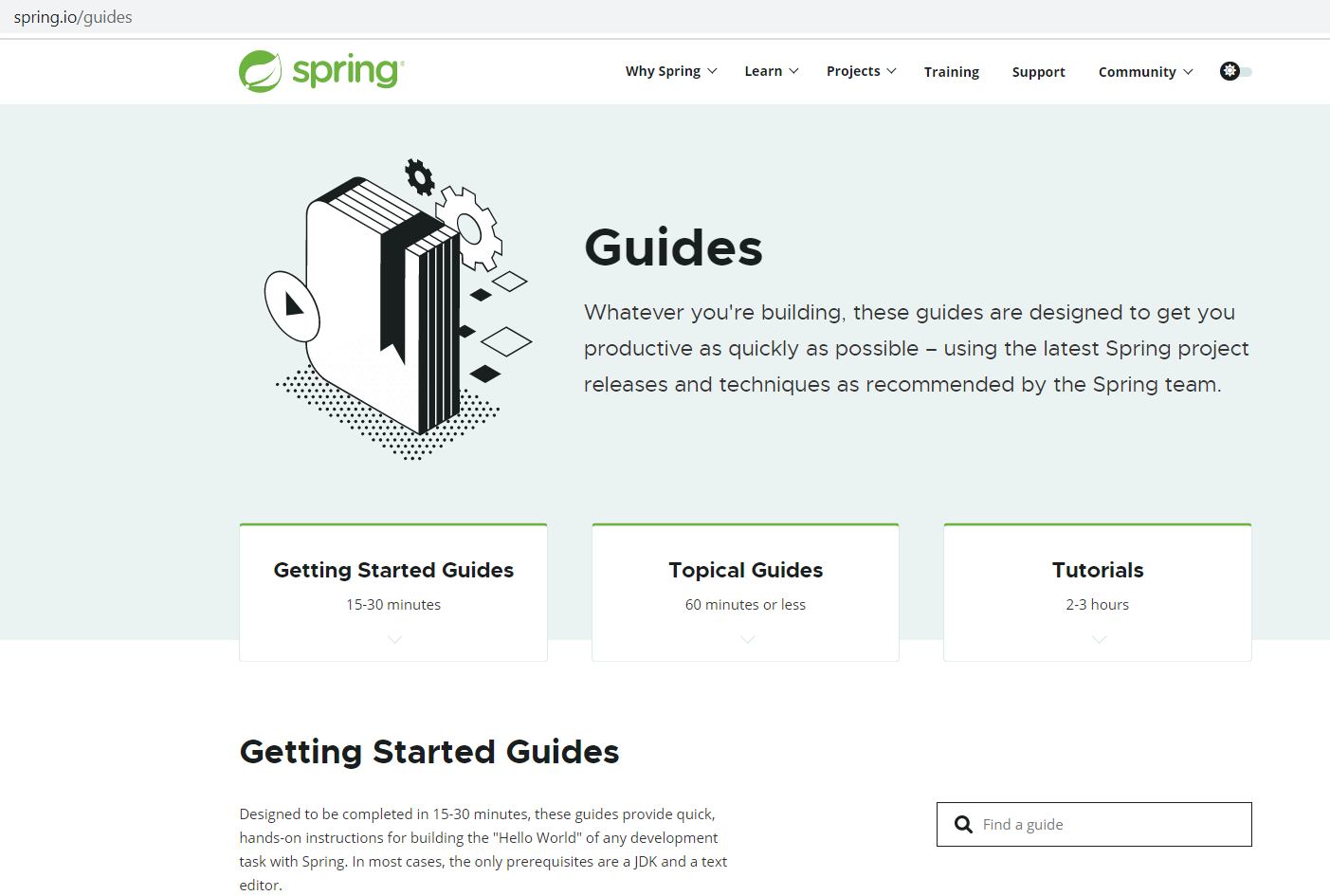
Dejaremos configurado java + eclipse + tomcat in eclipse

Debemos conocer el sitio spring.io

Aquí vamos a ver todos los proyectos que podemos crear utilizando este framework



Y también muchos tutoriales



# Descarga e instalación librerías Spring. Spring JAR files

1. Creamos un Java Project
2. Creamos una carpeta llamada libs
3. Buscamos los jar de spring en la url https://repo.spring.io/release/org/springframework/spring/
4. Los agregamos en la carpeta libs
5. Lo agregamos al build path

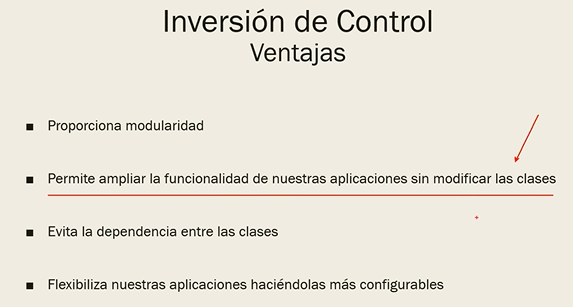
**Inversión Of Control - IoC**

Este paradigma lo invento Martin Fowler , se lo conoce como principio de Hollywood (por lo casting de actores )



La inversión externaliza la construcción y el manejo de la app , en nuestro caso se lo externalizamos

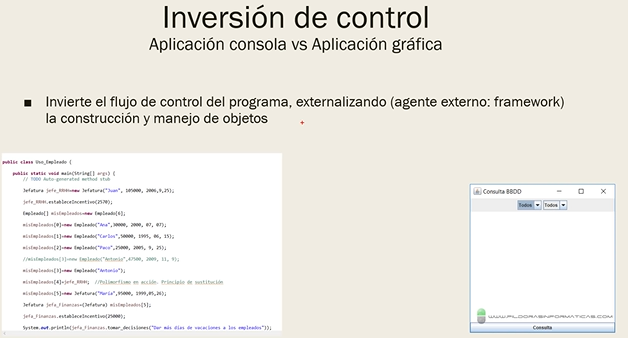
A Spring este manejara todo gestionar y controla los objetos y gracias a esto nos proporciona mas modularidad y ampliar la funcionalidad de nuestras clases sin tocar el código, evita la dependencia …



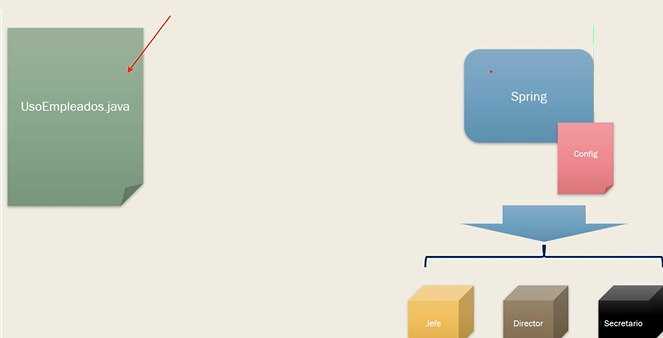
Por eje

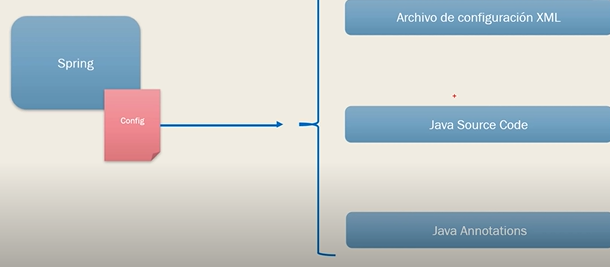
En un app de consola , el programa controla todo de forma clara , sabe cuándo crear objetos , llamar a un método , gestionar todo el programa .

Si embargo cuando hacemos una app grafica ya no es la app quien controla todo por que entran en juego todos lo evento , y la app no sabe exactamente cuándo se pulsara un botón , la app grafica sería un agente externo a nuestra app y es quien tiene el control , de esta forma se invierte el control de la app



Imaginemos tenemos una app java y esta necesita crear un objeto , la app hace una petición a spring pidiendo un objeto , entonces Spring a través un contenedor de objetos y un archivo de conf , este archivo se puede hacer de varias formas , este en quien en definitiva proporciona una instancia a la app java ,



El conf Se puede configurar de distintas formas (ver git hub proyecto )

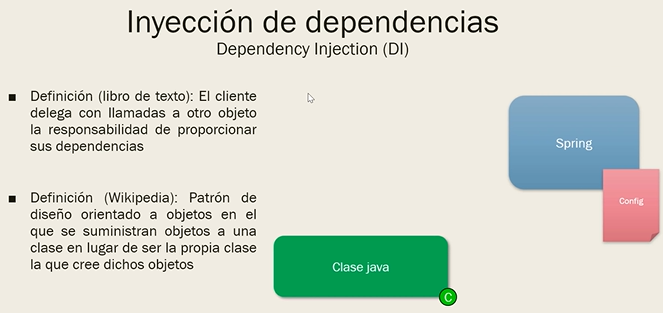
# Inyección de dependencias

Es una forma de diseñar nuestra app , que va de la mano del concepto de la modularizacion osea el programa debe estar creado por diferentes objetos independientes que se comuniquen y funcionen como una unidad , igualmente todos los objetos necesitan de otro para funcionar correctamente

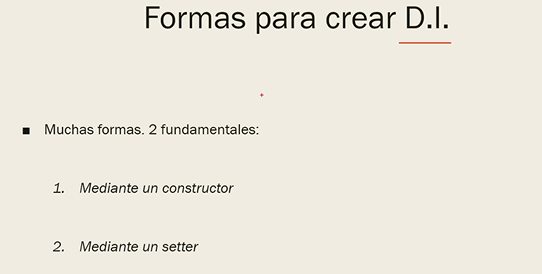


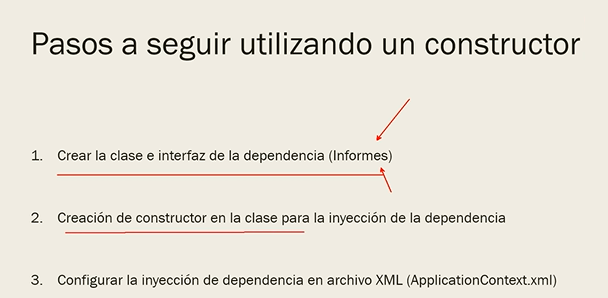
Spring es quien inyectara los objetos que necesita la clase para funcionar de manera adecuada, en

Vez de crear vos las clases con el NEW , la ventaja donde esta? Es que en el futuro si otra clase necesita esas misma dependencias se las podemos inyectar sin tener que intanciarlas ,es bastante útil cuando utilizamos acceso a ficheros externos , accesos a bases de datos



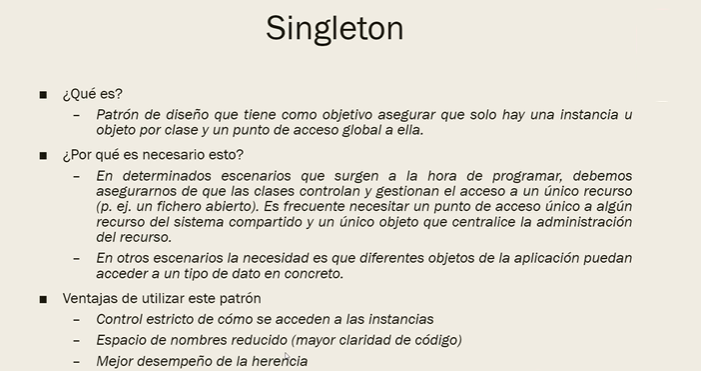
También existe el AutoWiring , que lo veremos más adelante cuando vean anottations





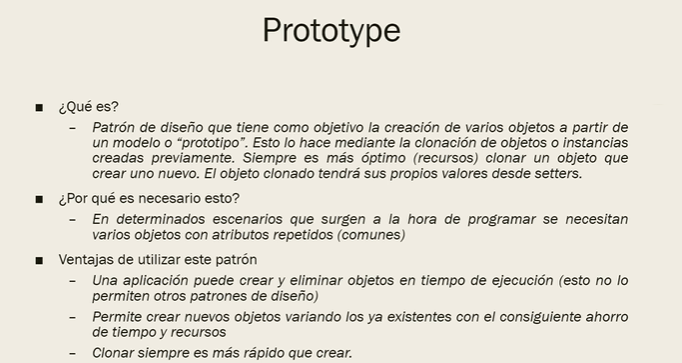
CLASE 2 - SINGLETON

Spring proporciona los beans mediante este patrón singleton

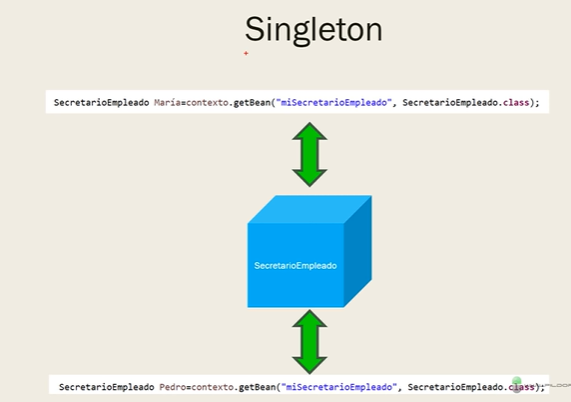


Patrón prototype

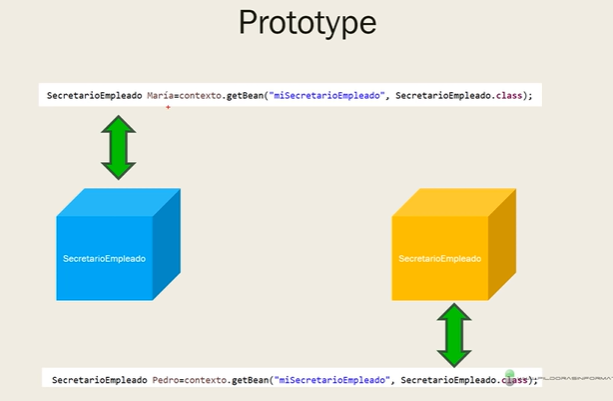
Es lo opuesto a singleton



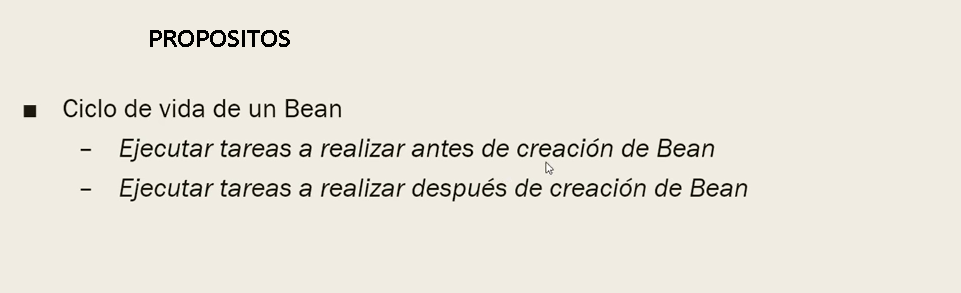
Si por ejemplo le pedimos a spring 2 beans manipulados con singleton (que es como lo manipula por defecto ) lo que ocurre es que recibiremos ambos beans apuntaran al mismo objeto de tipo secretarioEmpleado



Ahora la situación cambia si en vez de utilizar el patrón singleton utilizamos el patrón prototype , ya que cada vez que le solicitemos un bean a spring bajo este patrón de diseño prototype , cada beans apuntara a un objeto diferente como en este grafico



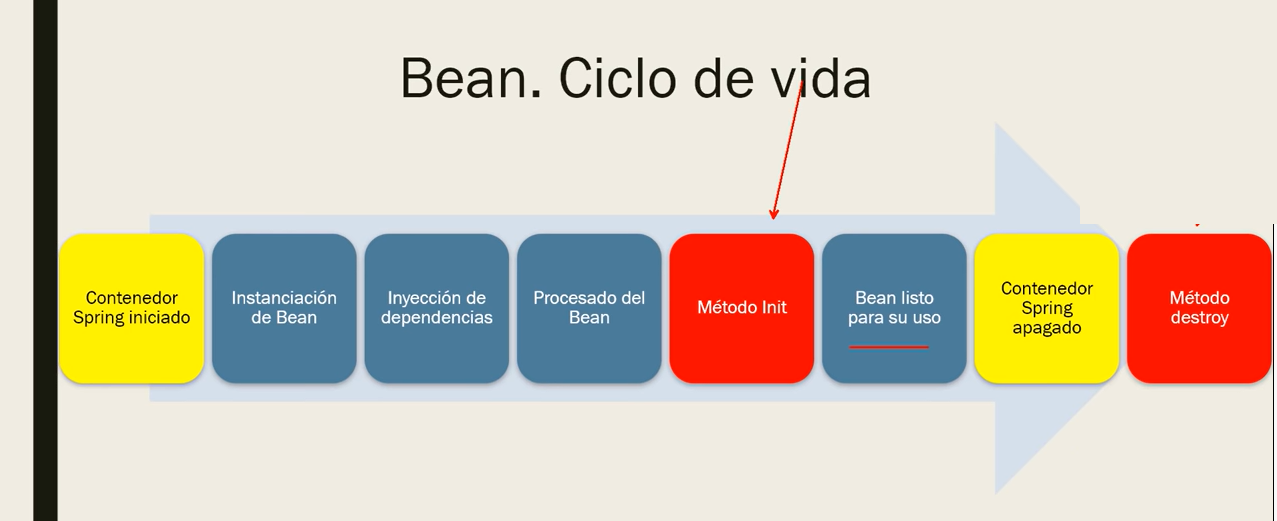
Ciclo de vida de un beans





1. arranca el contenedor de spring
2. se instancia el beans
3. se le inyectan las dependencias
4. internamente el container de spring procesa el bean
5. una vez procesado lo diponibiliza para que lo usemos
6. una vez que lo usamos y realizamos lo que necesitábamos en nuestra app el contenedor de spring se apaga

en muchas ocasiones necesitamos hacer tareas previas a que el bean esté disponible o posteriores al apagado del contenedor de spring , esto se consigue con dos métodos que son el init y el destroy

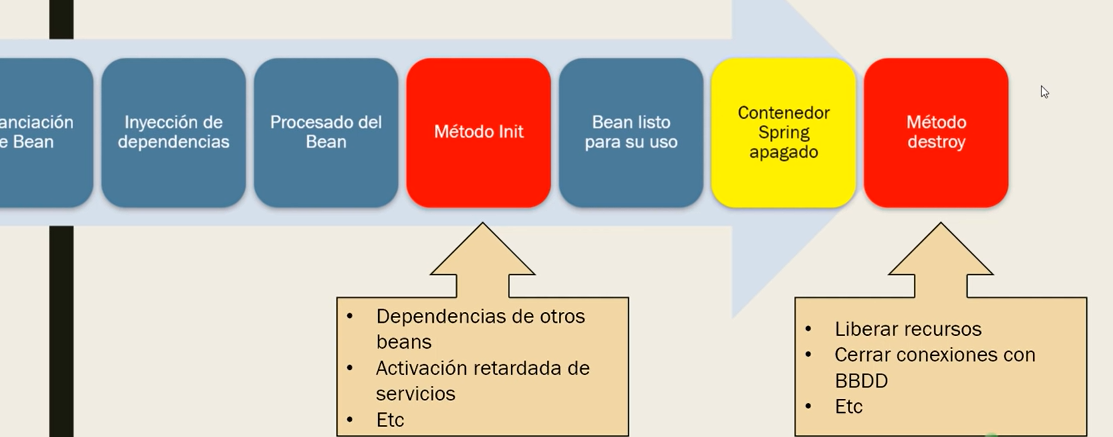


El método init podríamos usar para cargar dependencias de otros beans , ya que en ocasiones un beans necesita de otros beans o para la activasion de servicios por ej abrir un socket o una conexión a una db , necesitamos tener la conexión abierta antes que en bean este disponible

El propósito del destroy me sirve por ejemplo para cerrar la conexión a la base de datos o cerrar los sokect entre otros propósitos .

Es fundamental saber ejecutar tareas antes y después de utilizar el bean

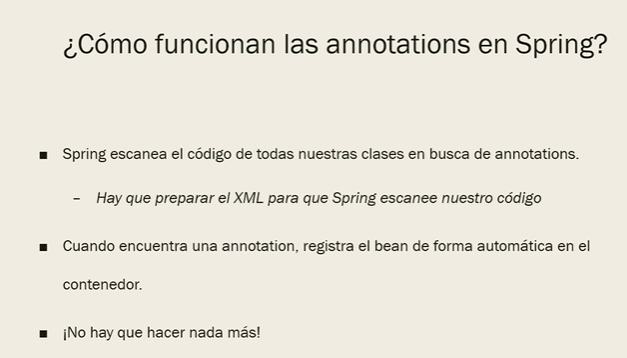
Ahora veamos un ejemplo con director empleado



**ANOTACIONES**

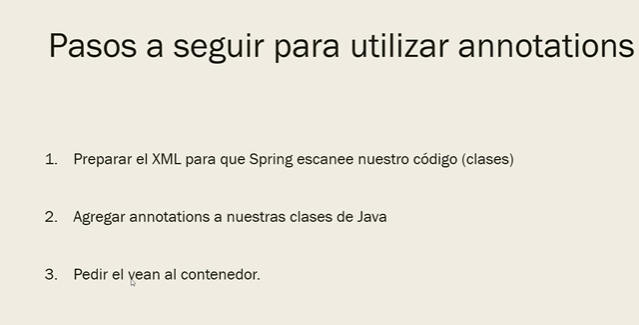
****

**Spring escanea recursivamente nuestras clases buscando xml**

****

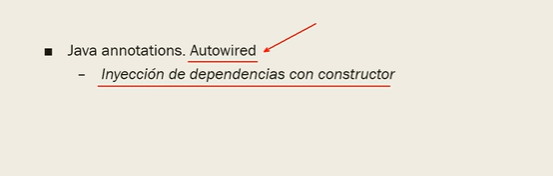
**Entonces con una simple anotación spring registrara el bean y cuando se lo pidamos spring nos lo va a proporcionar , no hara falta escribir código en el xml**

****

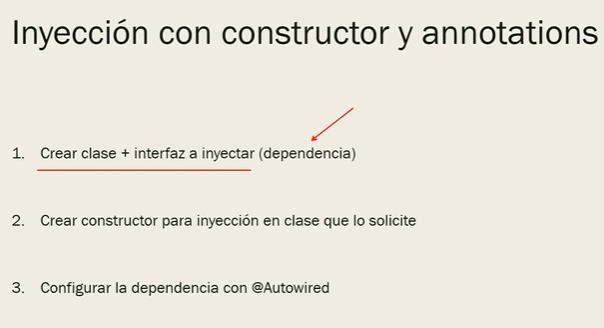
****

**Demo 1 : anotation @component , utilizada para registrar un bean**

**Demo 2 : @Autowired utilizada para inyecta dependencias**

****

**Paso para crear inyección de dependencias con constructor**

****

**Anotation @Scope : sirve para utilizar patrón singleton y prototype pero directo con las anottations**

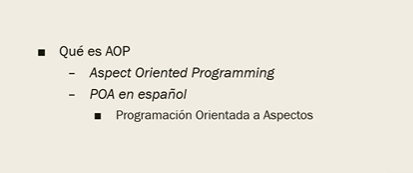
Anotation @PreDestroy y @PostConstruct me sirve para ejecutar código cuando se

Destruye un objeto y cuando se crea , aca podremos crear nuestro propios métodos y agregamos la anotaciones según corresponda

Predestroy seria cuando el objeto se destruye

Y posctConstructor antes que se crea

**AOP**

****

**Imaginen que tenemos un método que se encargar de inserta un cliente en un db , este método recibe dos parámetro ,crea la sesión y luego utiliza esta sesión para insertar el cliente en la db **

**Imaginen que la persona que va a insertar un cliente previamente tiene que pasar filtros de seguridad , por ejemplo el usurio debe estar logueado y el método insertar cliente debe comprobar que ese user tiene los permisos necesarios para insertar en la base de datos , entonces si queremos hacerlos deberíamos deberíamos chequear que esos permisos sean validos y tenga el role necesario para hacer esa acción**

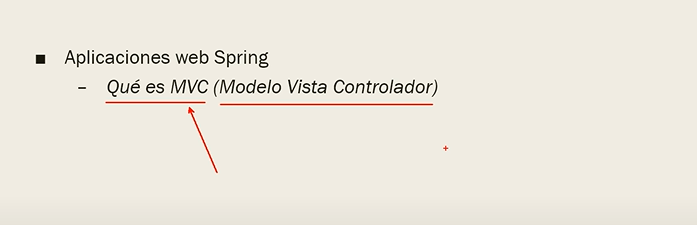
**Imaginamos que necesitamos hacer esto también para borrar y modificar otros objetos de la base , tendríamos mucho código repetido …**

[**https://www.youtube.com/watch?v=AjXPs9nVHow&list=PLU8oAlHdN5Blq85GIxtKjIXdfHPksV\_Hm&index=76**](https://www.youtube.com/watch?v=AjXPs9nVHow&list=PLU8oAlHdN5Blq85GIxtKjIXdfHPksV_Hm&index=76)

**AOP Ponitcut Expression :**

**Son patrones que podemos utilizar en las anotaciones que hacemos a la hora de utilizar programación orientada a aspectos a la hora de crear nuestro aspecto para que podamos afinar donde se realiza esa búsqueda o donde se aplica esa programación orientada a aspectos**

**Clase 4 spring MVC**

****

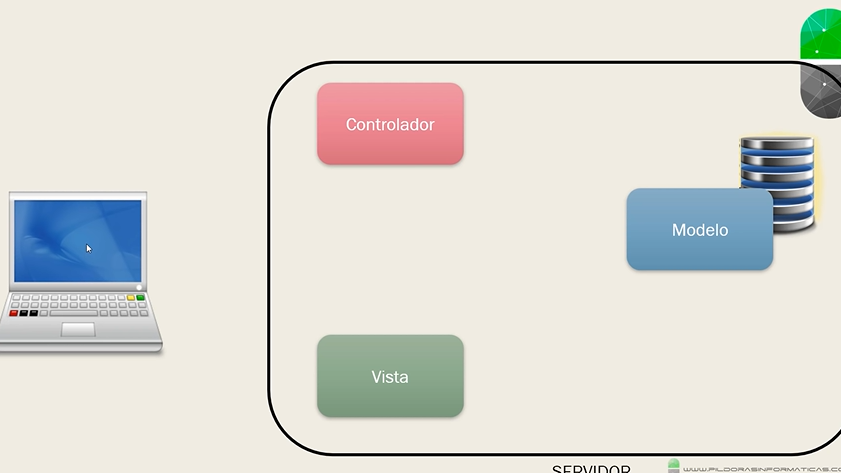
**Que es mvc?**

Un patrón que consiste en dividir nuestra app en 3 partes osea modularizarla en

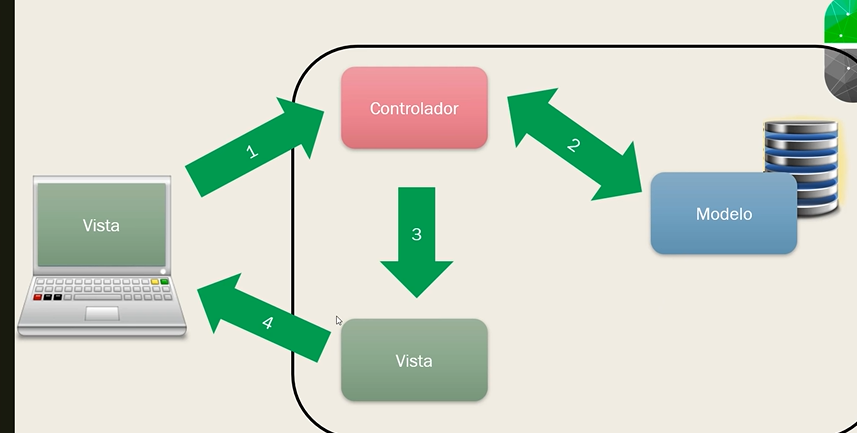
Contralor

Modelo

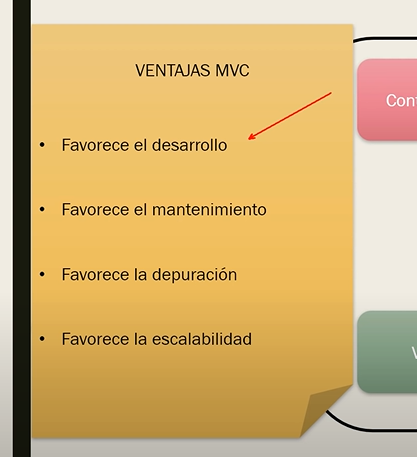
Vista



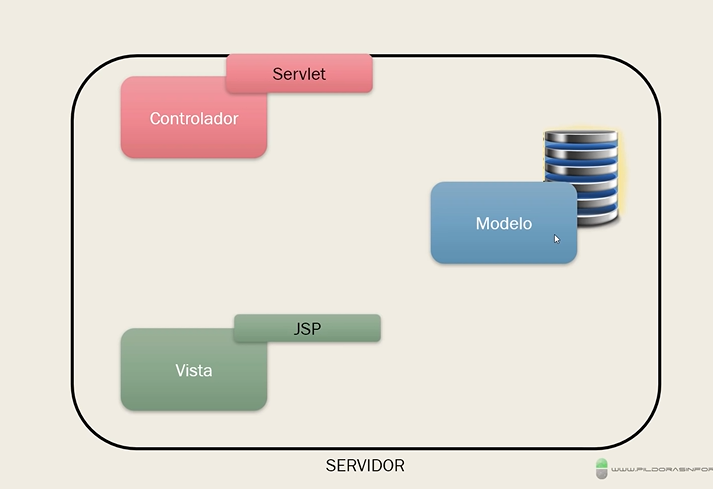
El user a través de un form hace una petición al controlador el controlador la procesa y consulta con el modelo y el modelo le brinda la información , luego una vez que el controlador tiene la información se la pasa a la vista que no es mas que una pagina web que se enviara al user



La modularizacion siempre favorece el mantenimiento , es mas fácil buscar errores y escalar la app , si en algún futuro necesitas hacer que la app crezca te será mas fácil

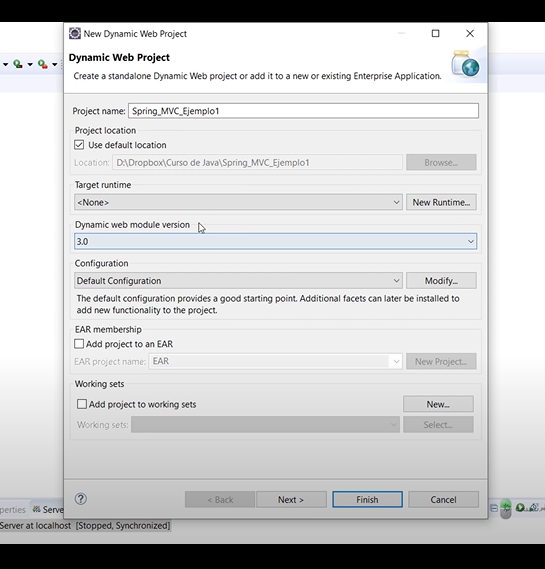


**Y como se hace esto con Spring?**



Fundamental tener bien configurado el tomcat!

Veamos como crear y configurar ahora nuestra app Spring mvc **en eclipse**

1. creamos un web dynamic Project (dejar los parámetros por default!) ,el target runtime en “none”
2. agregar el tomcat en eclipse
3. dentro de la carpeta lid del proyecto agregamos todos los jars de spring
4. y necesitaremos dos librerías más para trabajar con proyectos web por un lado (javax.servlet.jsp.jstl-1.2.1.jar y javax.servlet.jsp.jstl-api-1.2.1.jar)

link de descarga , www.pildorasinformaticas.es/archivos/spring/video27\_1.zip

1. también necesitaremos dos archivo .xml de configuración

(spring\_mvc\_servlet.xml y web.xml) lo podemos bajar de esta url : [www.pildorasinformaticas.es/archivos/spring/video27\_2.zip](http://www.pildorasinformaticas.es/archivos/spring/video27_2.zip)

Los vamos a pegar dentro de la carpeta WEB-INF (pero afuera de lib )

Estos dos archivos xml son casi siempre idénticos por lo que podríamos hacer un copy paste , esto también se puede hacer con código java sin el xml.

1. Una vez explicados los archivos xml pasamos a la creación de nuestro package
2. Creamos una clase controlador.java en el paquete del punto 6
3. Creamos la carpeta vista y agregamos un nuevo jsp que nos retorna nuestro ejemplo
4. url de test <http://localhost:8080/Clase4_Spring_MVC_Ejemplo1/>

ahora creamos nuestro primer formulario

1. crear el controlador y sus dos metodos
2. creamos el formulario holaalumnosFormulario.jsp
3. creamos el jsp HolaAlumnosSpring.jsp
4. crearmos un href en paginaEmpleo.jsp