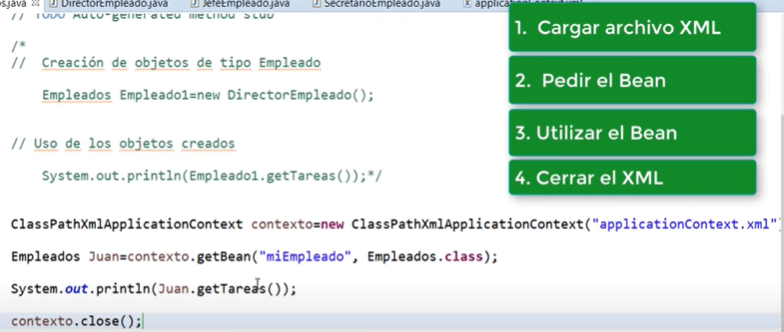
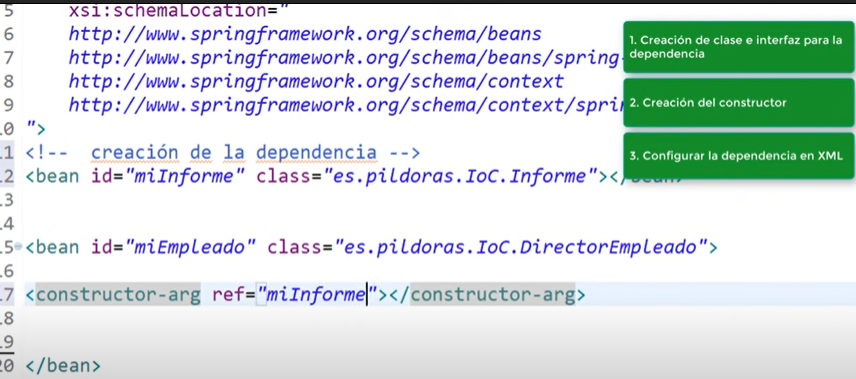
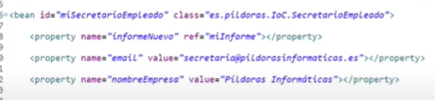
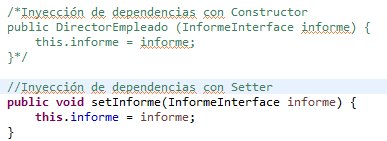
# CURSO SPRING

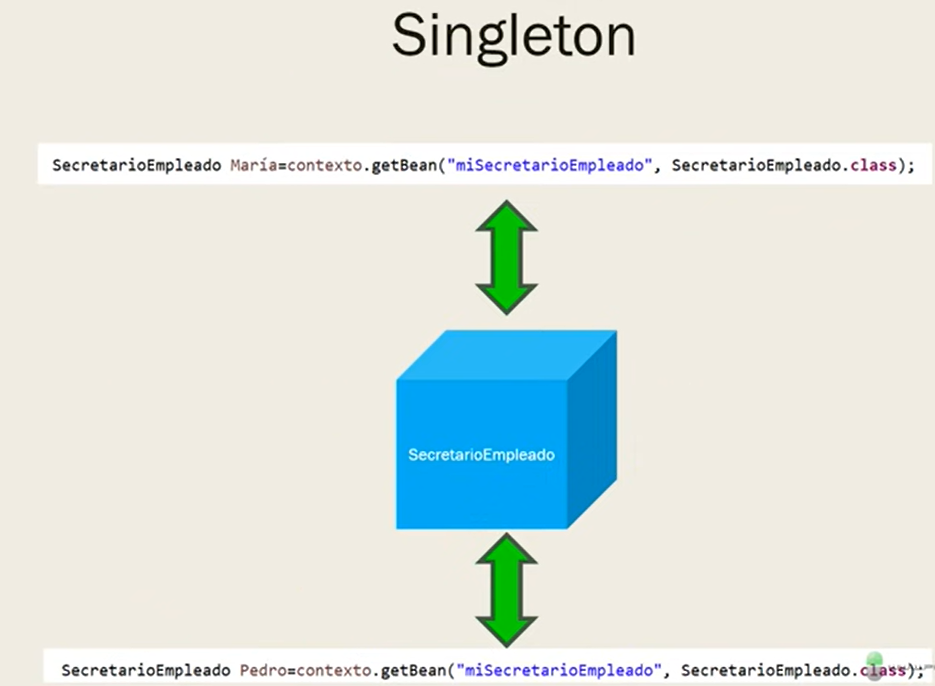
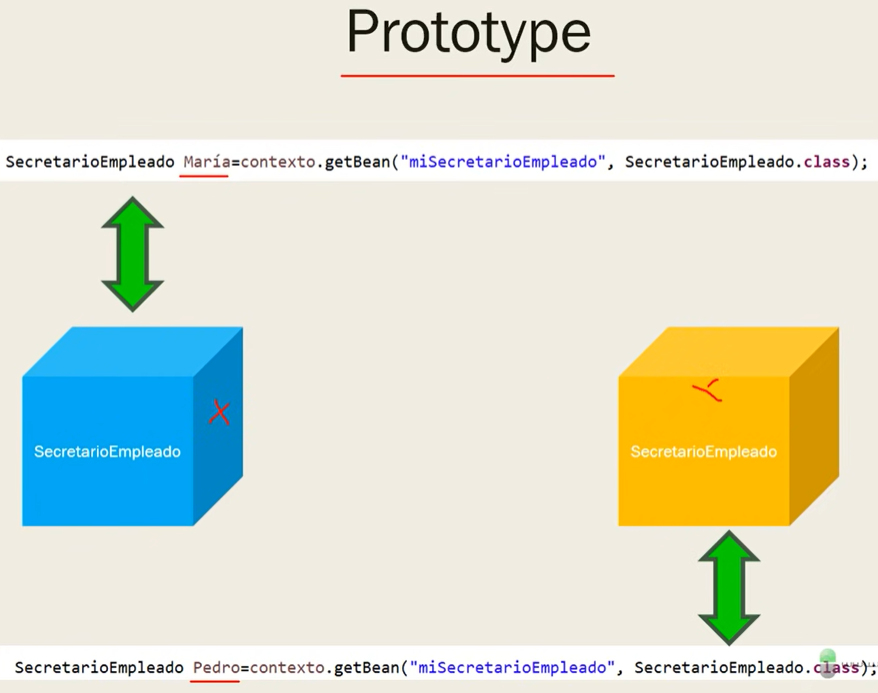
## Core Spring

* Ventajas de Spring
  + Inyección de dependencias (Favorece el “loose coupling” -Los componentes están débilmente asociados-)
  + Desarrollo sencillo de POJOs (Plain Old Java Objects -Clases simples de Java, que no heredan o implementan interfaz)
  + Minimiza el código repetitivo
  + Simplifica el acceso a datos
  + Programación Orientada a Aspectos (AOP – Favorece el modularidad)
* Estructura del core de Spring
  + Contenedor central
    - Creación de Beans, manejo de propiedades y dependencias y el Spring Expression Language (SpEL)
  + Módulo de infraestructura
    - Maneja todo lo referente a las transacciones, logueos, a la seguridad, a la AOP, etc.
  + Acceso a datos
    - En este módulo se maneja todo lo relacionado a las bases de datos: Como las transacciones, el JDBC y el ORM (Object Relational Mapping –Poder tratar a las bases de datos como objetos).
  + WEB/MVC
    - Permitir el acceso web remoto, utilizando los Servlet, Portlet, WebSocket, etc.
  + Pruebas
    - Permitir realizar pruebas; se utilizan los mock objects

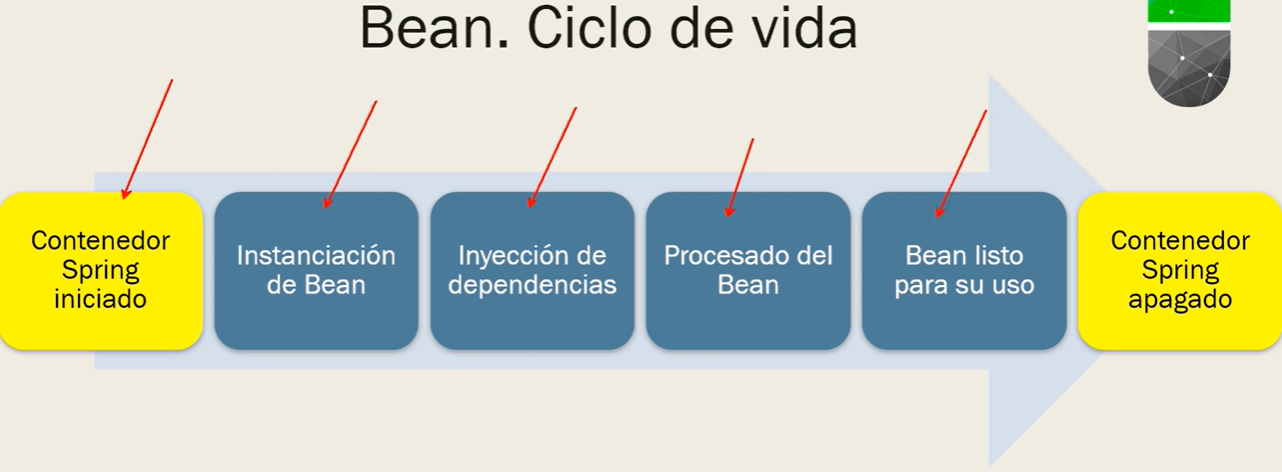
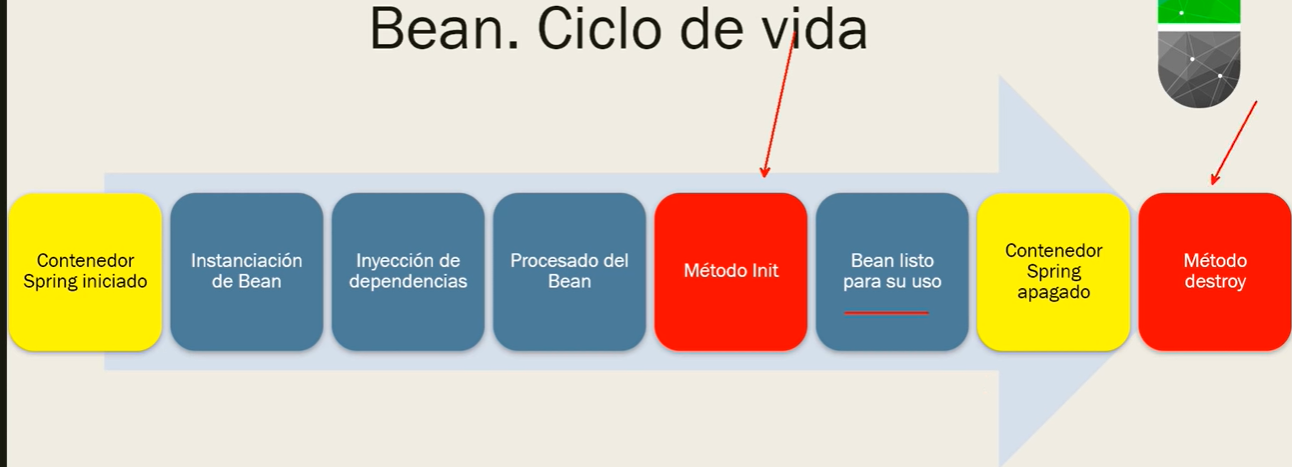
## Inversion of Control

* Invierte el flujo de control del programa, externalizando (Agente externo: Framework) la construcción y manejo de los objetos
  + Ventajas
    - * Proporciona modularidad
      * Permite ampliar la funcionalidad de nuestras aplicaciones sin modificar las clases
      * Evita la dependencia entre las clases
      * Flexibiliza las aplicaciones haciéndolas más configurables
  + Una **interfaz** en Java es una colección de métodos abstractos y propiedades constantes. En las interfaces se especifica qué se debe hacer, pero no su implementación.
  + Los **métodos abstractos** solo tienen una declaración de función y no tienen implementación de método. Esto significa que contiene solo un cuerpo vacío y no hay código dentro del método.
  + El archivo de configuración de Spring se puede crear de tres formas:
    - Archivo de configuración XML
    - Java Source Code
    - Java Annotations
  + 
* Inyección de dependencias
  + El concepto de dependencia deriva de la modularización. La inyección de dependencias en un patrón de diseño orientado a objetos en el que se suministran objetos a una clase en lugar de ser la propia clase la que cree dichos objetos
  + Formas para crear la inyección de independencias:
    - Mediante un constructor
      * Crear la clase e interfaz de la dependencia
      * Creación del constructor en la clase para la inyección de la dependencia
      * Configurar la inyección de dependencia en archivo XML
      * 
    - Mediante un setter
      * Crear la clase e interfaz de la dependencia
      * Creación del setter en la clase para la inyección
      * Configurar la inyección de dependencia en archivo XML
      * 
      * 

## Singleton y Prototype

* Singleton:
  + Es el patrón de diseño por defecto de Spring
  + Tiene como objetivo asegurar que solo haya una instancia u objeto por clase y un punto de acceso global a ella
  + Ventajas
    - Control estricto de cómo se acceden a las instancias
    - Espacio de nombre reducido (Mayor claridad de código)
    - Mejor desempeño de la herencia
  + 
* Prototype
  + Patron de diseño que tiene como objetivo la creación de varios objetos a partir de un modelo o “prototipo”. Esto lo hace mediante la clonación de objetos o instancias creadas previamente. El objeto clonado tendrá sus propios valores desde setters.
  + Ventajas:
    - Una aplicación puede crear y eliminar objetos en tiempo de ejecución
    - Permite crear nuevos objetos variando los ya existentes con el consiguiente ahorro de tiempos y recursos
    - Clonar siempre es más rápido que crear
  + 

## Ciclo de vida de un Bean

* Un bean es una clase Java cuyo propósito es encapsular la información, para hacer uso de él después
* 
* 
* 
* 

## Java Annotations

* ¿Qué son las Java Annotations
  + Son “etiquetas” que se añaden a las clases, métodos, campos, variables, etc., en un programa Java
* ¿Para qué sirven las Java Annotations?
  + Para añadir metadatos a nuestros programas (clases) en Java
  + ¿Qué son los metadatos?
    - Un conjunto de datos que describen el contenido y/o propósito de un objeto
* Poder describir la naturaleza de un objeto (metadatos) con annotations nos permite utilizarlas como sustituto de los archivos XML
* Las annotations son procesadas en tiempo de compilación o también en tiempo de ejecución
* ¿Cómo funcionan las annotations en Spring?
  + Spring escanea el código de todas nuestras clases en busca de annotations
    - Hay que preparar el XML para que Spring escanee nuestro código
  + Cuando encuentra una annotation, registra el bean de forma automática en el contenedor
* Ventajas de usar Java Annotations con Spring
  + Mayor facilidad en el momento de configurar beans
  + Simplifica enormemente el código del archivo de configuración XML
  + Especialmente útil cuando debemos configurar cientos de beans en un programa
* Pasos a seguir para utilizar annotations
  + Preparar el XML para que Spring escanee nuestro código (clases)
  + Agregar annotations a nuestras clases de Java
  + Pedir el bean al contenedor
* Inyección de dependencias con constructor (@Autowired)