https://www.pildorasinformaticas.es/

Curso Spring

Paso 1:

Vistos los videos:

Instalación e introducción: 1, 2, 3, 4 y 5
Inversión de control: 6, 7 y 8
(me falta poner el código de las clases)

Video 8:

Fichero XML

Con <bean>

se crea un generador de objetos
se crea un contenedor Spring que nos proporciona objetos

Cuatro pasos a seguir para utilizar un contenedor Sping

1- Cargar el archivo XML
(así se crea un contexto)
2- Pedir el bean
(se pide el objeto al contexto, al fichero XML)
3- Utilizar el bean
(se utiliza este objeto)
4- Cerrar el archivo XML
(cerrar el contexto)

Video 9:

Inyección de dependencias. Modularización

Modularización en POO

1- Este concepto nos indica que el todo está formado por partes
2- El todo es un conjunto de objetos independientes,
que se comunican entre sí para formar el todo
3- Los objetos a veces tienen entre ellos dependencias, es decir,
necesitan de otros objetos para funcionar correctamente.
4- Así un objeto necesita de muchos objetos, depende de
muchos objetos.

Inyección de dependencias. Funcionamiento

1- Cuando un objeto1 necesita otro objeto2 para funcionar
adecuadamente, en lugar de instanciarlo con new, se lo pide
a Spring.
Spring lo crea y después se lo inyecta, se lo proporciona.
2- Ventaja: este objeto2, que crea Spring, también se lo puede
inyectar a otro objeto3 que lo necesita.

Ejemplo concreto:

1- Ahora Jefe, Director y Secretario no sólo realizan un tarea.
También emiten informes, dependen de un informe.
Se deben crear objetos de tipo informes, pero no con new.
2- Spring inyect los informes y hay dos formas de hacerlo:
- Con constructor
- Con un setter
- (auto-wiring aún no se ve; sí en anotations)

Pasos para crear dependencias con un constructor

1- Crear la clase e interfaz para la dependencia (Informes)
2- Crear un constructor en la clase a la que se le va a inyectar la dependencia
3- Configurar la inyección de dependencia en el archivo XML

Video 10:

Pasos para crear dependencias con un setter

1- Crear la clase e interfaz para la dependencia (Informes)
2- Crear un método setter en la clase a la que se le va a inyectar la dependencia
3- Configurar la inyección de dependencia en el archivo XML

Para realizar esto se utiliza la clase SecretarioEmpleado.

Video 11:

Inyección de campos I

1-Cuando un objeto necesita ciertos campos, vamos a ver cómo inyectarlos
2- Supongamos que el SecretarioEmpleado necesita dos campos: email y nombreEmpresa

Pasos a seguir:

1- Primero crear estos campos (atributos) en SecretarioEmpleado
2- Crear lo métodos setter y getter de estos campos
(son importantes lo nombres de los setters)
3- En el archivo XML se crea esta inyección.
Nos fijamos en el bean que tenemos para el SecretarioEmpleado.
Le vamos a agregar dos propiedades.
4- No se debe utilizar la interfaz Empleados para acceder a los setters de estos campos, se debe utilizar la clase SecretarioEmpleado.

Hacemos lo mismo que en SecretarioEmpleado en la clase DirectorEmpleado

Video 12:

Inyección de campos II

1- Se utilizará un archivo externo de propiedades
2- Ventaja: la modificación de las propiedades será más cómodo
3- Objetivo: inyectar las propiedades desde un archivo externo
Este archivo será datosEmpresa.propiedades
(propiedad=valor)

En el archivo XML

1- Se carga el archivo de propiedades
2- Los valores de las propiedades se obtienen de dicho archivo.
(ej: ${email})

Video 13:

Singleton y Prototype

1- Son dos patrones de diseño

Singleton

1- Se utiliza para acceder a un único recurso global.
2- Sólo va a haber una instancia u objeto por clase
3- Nosotros pedimos beans al contenerdor de beans de Spring y nos devuelve una referencia que apunta a un único objeto.
(ej:un objeto apuntando a un único fichero abierto)
4- Spring utiliza Singleton por defecto

Prototype

1- Es lo contrario a Singleton, es decir, nos permite que una clase pueda tener varios objetos o instancias y estos objetos se crean por clonación.
2- Se utiliza un objeto como prototipo y después el resto se clonan.
(clonar siempre es más rápido que crear)
3- Spring puede utilizar este patrón, pero se lo hay que indicar.

Vemos 2 ejemplos, un de cada patrón

1- Hacemos una copia del fichero XML
(cambiándole el nombre: applicationContext2.xml)
Y ahora trabajamos con él, eliminando cosas innecesarias.
2- Quitamos lo del fichero externo y los bean de DirectorEmpleado.
3- Nos centramos en el bean de SecretarioEmpleado
(le quitamos a éste todas las property)

4- Creamos una clase aplicación nueva: UsoDemoSingletonPrototype
. Cargamos el xml
. Hacemos petición de beans al contenedor.
. Pedimos 2 beans de SecretarioEmpleado
. Cambiamos el nombre de ambos beans (María y Pedro)
(estos dos beans deben apuntar al mismo objeto SecretarioEmpleado)
. System.out.println() de ambos, María y Pedro, y ambos dan las misma referencia
5- Ahora vamos a trabajar con el patrón Prototype.
. Vamos el fichero xml y en el ben ponemos scope=”prototype”
. Ahora cada bean apunta a objetos diferentes.

Video 14:

1- Ciclo de vida de un bean
. Ejecutar tareas antes de crear el bean y
. Ejecutar tareas tras utilizar el bean y cerrarlo
. Así están los métodos init y destroy.
2- En ini
. Por ejemplo, podemos cargar dependencias de otros beanx
. Abrir un socket o una conexión a la base de datos
3- En destroy
. Por ejemplo, se pueden liberar recursos, cerrar conexiones, etc.
4- En la clase DirectorEmpleado vamos a hacer lo visto
. En el final de la clase creamos el métodoInit y el método destroy.
. Así ponemos el método Init (suele ser de tipo void) y lo llamamos como queramos, por ejemplo “métodoInicial()”, y le ponemos un mensaje, para verlo cuando se ejecute.
. Ahora ponemos el método destroy y lo llamamos “métodoFinal()”, de tipo void también.
5- En el fichero XML
. Le indicamos que utilice estos métodos en el bean.
. Por claridad, creamos un nuevo XML a partir de otro existente.
. Llamamos a este fichero “applicationContext2.xml”
. El bean lo limpiamos y le ponemos lo siguiente, con 2 modificadores:
<bean …..
init-method=”metodoInicial”
destroy-method=”metodoFinal”
6- Los probamos
. cuando generemos un informe con el bean de DirectorEmpleado
. Así, antes de generar el informe, ejecuta el método init y, tras generarlo,
. ejecuta el método destroy

…….
Video 74: https://youtu.be/qbFFrX-_bKs
Video 202 de java para instalar jdbc
https://youtu.be/TipyOAYGsdc