**Spring Core**

Objetivo: Creación de beans con Spring de modos diferentes realizando la inyección de dependencias necesarias entre todos los objetos involucrados

***Ejercicio 1: Creación de Beans***

**Paso 1: Creación de Beans.**

Crear la clase Direccion con las siguientes propiedades dentro del paquete beans:

private String cp;

private String calle;

Crear métodos get/set de las propiedades declaradas anteriormente.

Sobrescribir el método toString de la siguiente forma:

|  |
| --- |
| public String toString() {  // TODO Auto-generated method stub  return ToStringBuilder.reflectionToString(this);  } |

La clase ToStringBuilder se encuentra en la librería commons-lang.jar

Crear la clase Persona con las siguientes propiedades dentro del paquete beans:

private Long id;

private String nombre;

private String apellidos;

Crear métodos get/set de las propiedades declaradas anteriormente.

Sobrescribir el método toString de la siguiente forma:

|  |
| --- |
| public String toString() {  // TODO Auto-generated method stub  return ToStringBuilder.reflectionToString(this);  } |

NOTA: Recuerda que las clases Persona y Direccion deben implementar la interfaz java.io.Serializable.

**Paso 2: Establecer la dependencia entre los beans anteriores**

1. En la clase Persona, establecer la siguiente propiedad:

private Direccion direccion;

Crear métodos get/set de la propiedad anterior

**Paso 3: Crear la clase Lanzador dentro del paquete main, que contenga un método main con el siguiente código:**

public static void main(String[] args) {

Direccion d = new Direccion();

d.setCalle(“MI calle”);

Persona p = new Persona();

p.setNombre(“Luis”);

p.setDireccion(d);

System.out.println(p);

}

Observar que aparecen todas las propiedades de la persona correctamente por la consola

**Paso 4: Realizar el mismo ejemplo pero desde Spring.**

Añadir las librerías de Spring Context al proyecto y crear un fichero de configuración vacío. Nombrarlo beans.xml y asegúrate de que este en la carpeta src

Inlcuir el siguiente código de configuración:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id=*"persona"* class=*"example.Persona"* scope=*"prototype"*>

<property name=*"id"* value=*"1"* />

<property name=*"apellidos"* value=*"Perez"* />

<property name=*"nombre"* value=*"Paco"* />

<property name=*"direccion"* ref=*"dir"* />

</bean>

<bean id=*"dir"* class=*"example.Direccion"*>

<property name=*"cp"* value=*"29000"* />

<property name=*"calle"* value=*"Gran via"* />

</bean>

</beans>

Configurar cada uno de los beans de forma que el resultado sea similar al anterior.

Crear la clase LanzadorSpring dentro del paquete main, con el siguiente código:

|  |
| --- |
| **public** **static** **void** main(String args[]){  ApplicationContext appContext = **new** ClassPathXmlApplicationContext(**new** String[]{"beans.xml "});  Persona person=(Persona)appContext.getBean("persona");  System.*out*.println(person);  } |

Nota: ctx.getBean(“persona”), debe sustituir “persona” por el nombre del bean configurado en beans.xml.

**Paso 5: Eliminar el atributo scope=”prototype” dentro de la bean persona y observar que sucede si creamos 2 beans persona:**

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

ApplicationContext appContext = **new** ClassPathXmlApplicationContext(**new** String[]{"beans.xml "});

Persona person=(Persona)appContext.getBean("persona");

Persona p2 = (Persona) appContext.getBean("persona");

System.*out*.println(person);

System.*out*.println(p2);

}

¿Qué sucede con la referencia del objeto de cada bean? ¿Es la misma? ¿Cambia?.

Probar el ejercicio volviendo a cambiar el scope=”prototype”. Observar las referencias de objetos de ambos beans.

***Ejercicio 2:***

Crear una nueva clase Persona con la siguiente estructura:

private String nombre;

private int edad;

private float altura;

private Persona pareja;

private List<Persona> amigos;

private List<String> colores;

private Map<String,String> emails;

Realizar la inyección de dependencias de TODAS LAS PROPIEDADES en un nuevo archivo de configuración al que llamaremos (beans2.xml).

El método main no variará con respecto al último punto del ejercicio 1.

***Ejercicio 3: Creación de Beans desde factoría.***

Crear una nueva clase PersonaFactoria dentro del paquete beans, con el siguiente código

|  |
| --- |
| **package** ejemplo;  **import** org.apache.commons.lang.builder.ToStringBuilder;  **public** **class** PersonaFactoria {  **private** **static** PersonaFactoria *instancia*= **null**;  **private** String nombre;  **private** PersonaFactoria(){  System.*out*.println("Construyendo por Constructor");  }  **static**{  *instancia* = **new** PersonaFactoria();  }  **public** **static** PersonaFactoria getInstance(){  System.*out*.println("Construyendo via getInstance");  **return** *instancia*;  }  **public** **void** setNombre(String nombre) {  **this**.nombre = nombre;  }  **public** String getNombre() {  **return** nombre;  }  @Override  **public** String toString() {  **return** ToStringBuilder.*reflectionToString*(**this**);  }  } |

El archivo de configuración queda de la siguiente forma:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.1.xsd"*>

<bean id="personaFactoria" class="beans.PersonaFactoria" factory-method="getInstance"/>

</beans>

Mediante esta configuración es la factoría la encargada de crear el bean y no el contexto de spring. Es un ejemplo más de la versatilidad de Spring de adaptarse a nuestras clases.

***Ejercicio 4: Autowiring***

Realización de autowiring.

Crear una bean Persona dentro del paquete beans con las siguientes propiedades:

String nombre;

int edad;

float altura;

Direccion direccion;

Crear una bean Direccion dentro del paquete beans con las siguientes propiedades:

1. String cp;

Realizar el autowiring:

1. Por nombre
2. Por tipo
3. Por constructor